NISTIR 5897

Representation of Axes for Geometric Fitting

Theodore H. Hopp

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards and Technology
Manufacturing Engineering Laboratory
Manufacturing Systems Integration Division
Gaithersburg, MD 20899-0001

NIST

NISTIR 5897

Representation of Axes for Geometric Fitting

Theodore H. Hopp

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards and Technology
Manufacturing Engineering Laboratory
Manufacturing Systems Integration Division
Gaithersburg, MD 20899-0001

September, 1996

Representation of Axes for Geometric Fitting

Theodore H. Hopp

Manufacturing Systems Integration Division
National Institute of Standards and Technology
Gaithersburg, MD 20899-0001

ABSTRACT

We review methods for representing geometric axes in three dimensions and discuss why these methods have undesirable properties for geometric fitting applications. We then describe a new representation that seems better suited for use in problems of fitting geometric elements to data points. This representation was developed in the course of our work on testing of data analysis software in coordinate measurement systems. The new representation has allowed us to lower significantly the estimated uncertainty of our test results. Details and properties of the representation are discussed.

Keywords: computational metrology; geometric modeling; measurement uncertainty

INTRODUCTION

A common issue in many applications is how to represent axes in three dimensions.[†] One of the most common representations is a point on the axis and a direction vector. We will call this the *naive representation*, and it has serious disadvantages for some applications. We have found the naive representation particularly problematic in our work on fitting geometric surfaces to point data. At issue is the stability of the results of fitting.

Geometric fitting is different from many other geometric modeling applications. In a typical modeling application, one starts with a given geometry in some representation and is concerned with assessing geometric properties (evaluation, point classification, area, etc.). In such applications, stability means that two geometries with similar representations (in some sense of similarity) should have similar properties. However, stability does not preclude geometries with radically different representations from having the same properties. (In fact, this situation is highly desirable for geometric design, in which it is important to have a rich set of tools for defining geometry.) In contrast, geometric fitting is the inverse problem. One starts with a given property (the point set to be fit) and is concerned with computing the representation. In geometric fitting applications, stability means that two geometries with similar properties (e.g., similar point sets to be fit) should have similar representations. That is, a small change in the point set should not radically change the representation. Furthermore, stability does not preclude geometries with radically different properties from having the same representation. (That is, the same geometry may fit two very different point sets.)

It is not surprising, therefore, that the naive representation, as well as others developed for typical geometric modeling applications, causes problems when used for fitting. Several approaches have traditionally been used in geometric fitting to avoid the problems of the naive

[†]We use the term *axis* to mean an oriented line. This is in keeping with much of the literature of geometric fitting, but at odds with that of spherical data analysis[1]. In the latter, the term is used to mean the direction (without location) of an unoriented line.

representation. These approaches in turn have their own shortcomings.

As will be discussed below, one effect of these problems is to increase the uncertainty of measured values obtained with measurement systems in which fitting is part of the measurement process. This is an economically important issue, as increased measurement uncertainty forces tighter production tolerances (thus raising production costs) and reduces process yields.

We have developed a new axis representation method to improve geometric fitting routines used at the U.S. National Institute of Standards and Technology (NIST) in our work on algorithm testing for coordinate measurement systems[2]. The method can also be used to represent orientations (without location). Use of this representation has had the effect of reducing the uncertainty of our assessment of software performance, by an order of magnitude or more in some cases.

This report is organized as follows. In the next section, we review the naive representation and some existing methods for improving on it. Following that, we describe our new representation. We then discuss some issues in using our representation for fitting problems, followed by a report of our experiences in using this representation in our algorithm testing work. Finally, we discuss some advantages and disadvantages of our representation. An appendix following the references shows the derivation of certain formulae needed for our implementation.

EXISTING AXIS REPRESENTATIONS

This section discusses the naive representation, the problems associated with it, and various other representation methods used to overcome these problems. It also discusses the weaknesses of these other methods.

The Naive Representation

The naive representation, shown in Figure 1, consists of six numbers: the coordinates (x, y, z) of a point $\boldsymbol{a}$ on the axis and the components (u, v, w) of a vector $\boldsymbol{n}$ establishing the axis direction. The axis is the locus of points determined parametrically in t, $-\infty<t<\infty$, by:

$$\boldsymbol{p}(t) = \boldsymbol{a} + t\boldsymbol{n}. \tag{1}$$

In general, we cannot assume that $\|\boldsymbol{n}\| = 1$, since some fitting routines may not maintain the scaling of $\boldsymbol{n}$.

Most fitting applications are concerned with the orthogonal distance from a point to the axis, shown as d in Figure 1. For a point $\boldsymbol{q}$, this is given by:

$$d = \min_{t} \|\boldsymbol{p}(t) - \boldsymbol{q}\|, \tag{2}$$

where $\| \|$ denotes the Euclidean vector norm. Another distance frequently of interest is the projected distance between $\boldsymbol{q}$ and $\boldsymbol{a}$ along the axis, shown as l in Figure 1. This is given by:

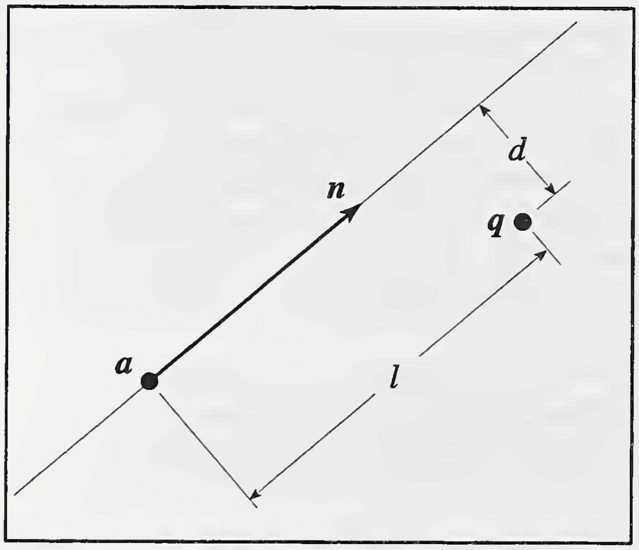

Figure 1 The naive representation.

$$l = \frac{(q - a) \cdot n}{\|n\|}. \tag{3}$$

In three dimensions, d can be written using the vector cross product:

$$d = \frac{\|(q - a) \times n\|}{\|n\|}. \tag{4}$$

Sometimes one wishes to avoid the vector cross product. In that event, d can be expressed using the Pythagorean Theorem[†]:

$$d = \sqrt{\|(q - a)\|^2 - l^2}. \tag{5}$$

Practical experience indicates that (4) is less subject to numerical roundoff error than is (5), particularly when point q is close to the axis. Equation (5), however, is more useful for deriving expressions for the derivatives of d.

Problems of the Naive Representation

The underlying difficulty with the naive representation is that a given axis can be represented by many different combinations of parameter values. This is the result of a surplus of parameters: an axis in three dimensions has four degrees of freedom, while the naive representation has six parameters. This multiplicity of representations has a geometric interpretation: any point on an axis will locate that axis, and any non-zero scaling of the direction vector will orient it.

That the naive representation is not unique can create severe problems in geometric fitting. Three problems seem most serious:

- *Fitting routines may fail.* Fitting routines can spend all their time searching for the "best" point on the axis or magnitude for the direction vector, when there really are no significant geometric differences between them.
- *Confidence limits may be poor.* Because the naive representation is not unique, assessing the sensitivity of the result to perturbations of the data being fit is hard. Most sensitivity analysis methods rely on the representation being of "full rank"—that it has no redundancies[3,4]. These methods fail for the naive representation.
- *Computed fits may be inaccurate.* Picking an axis point far from the data being fit, or an out-of-proportion orientation magnitude, can create large numerical roundoff errors due to poor scaling[5]. In extreme cases, the calculated results may contain no correct significant digits[6].

We emphasize that the first two problems are inherent weaknesses of the naive representation.

[†]This formula works nicely in any dimension, while the vector cross product is a three-dimensional operator.

Some optimization routines have built-in protections that reduce the effects of the third problem.

We now examine various methods used to avoid these problems. These are of two types: those adding constraints to the original problem and those transforming the representation to reduce the number of parameters. (Most of the methods are quite common; we have only provided references to notable uses of the methods in geometric fitting.) As we go along, we will note the shortcomings of these alternatives.

Introducing Constraints

The two extra degrees of freedom of the naive representation can be eliminated by introducing constraints. The two most common constraints are:

- Force $\boldsymbol{n}$ to be a unit vector;
- Force $\boldsymbol{a}$ to be the point on the axis closest to a given, fixed point.

For some fitting geometries (e.g., lines and planes), these constraints can be used to dramatically simplify the formulation of the fitting problem, to the point that iterative methods are no longer needed. Usually, however, iterative methods must be used and the constraints must be either incorporated into the optimization method (see, e.g., Bertsekas[7]) or enforced outside the method at key points in the iterations. In the former case, the need to deal with constraints considerably complicates the optimization routines. In the latter case, the convergence properties of the optimization method may be affected. Also, sensitivity analyses cannot be done easily.

Eliminating Parameters

Rather than introducing constraints directly in the problem, one can use them to eliminate parameters. We describe several methods for doing this.

One method of using constraints to eliminate parameters involves iteratively transforming the data so that the fitted axis is close to a given axis, typically the z coordinate axis. It then becomes possible to fix the value of one or two parameters. For instance, one can introduce the constraints z=0 and w=1. (This method was reported by Murthy[8].) In essence, then, the fitting problem is reduced to that of finding four numbers: the x and y coordinates of the axis intersections with the z=0 and z=1 planes.

Note that this method can fail if the data are not rotated. The geometric fit may be parallel (or nearly so) to the x-y plane. Potentially, numerical scaling problems may exist as well if the axis is far from the origin. Thus, it is imperative that the data be rotated and translated between iterations to bring the fitted axis close to the z coordinate axis.

Having to stop the iterations to reorient the data is a minor nuisance; many optimization routines have built-in support for such processing. The major shortcoming of this method is computational expense, especially when many points are being fitted. There also remain potential scaling problems if the transformed data extend far beyond the range [0, 1] in z.

Another way to eliminate parameters is to observe that, if $\|\boldsymbol{n}\| = 1$, $\boldsymbol{n}$ represents a point on the unit sphere. This allows us to write $\boldsymbol{n}$ using spherical angles. We will use θ for the declination of $\boldsymbol{n}$ from the +z axis and ϕ for the azimuth (angle from the +x axis) of the projection of $\boldsymbol{n}$ in the *x-y* plane. We have the mapping:

$$n = \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} \cos\phi\sin\theta \\ \sin\phi\sin\theta \\ \cos\theta \end{pmatrix} \tag{6}$$

and the inverse mapping:

$$\begin{pmatrix} \phi \\ \theta \end{pmatrix} = \begin{pmatrix} \tan^{-1}(v/u) \\ \cos^{-1}(w) \end{pmatrix} \tag{7}$$

where $0 \le \phi < 2\pi$ (the quadrant being determined from the signs of u and v) and $0 \le \theta < \pi$. These mappings allow us to use ϕ and θ instead of u, v, and w, thus eliminating one parameter. Distances from points to the axis can still be calculated with (4) or (5), by using the mapping (6).

This technique works well most of the time, but it still has problems when $\sin\theta$ becomes small (that is, when $\boldsymbol{n}$ approaches the z axis). It is well-known that no parameterization of the sphere is free of singularities, and the above parameterization is no exception. As $\sin\theta$ approaches zero, ϕ becomes computationally unstable, and when $\sin\theta$ is zero, ϕ is completely indeterminate.

Even with spherical angles, we still have one extra parameter—for location. The rotation method (described above) allows us to eliminate a parameter if we constrain $\boldsymbol{a}$ to be the axis point closest to a given point.

If $\boldsymbol{a}$ is the axis point closest to a given point $\boldsymbol{a}_0$ then $(\boldsymbol{a} - \boldsymbol{a}_0)\cdot\boldsymbol{n} = 0$. Assume that we iteratively transform the data so we are guaranteed that $w \neq 0$. Then we can solve for z in the other parameters of the naive representation and the coordinates of $\boldsymbol{a}_0$. Combined with the spherical angle method, this gives a minimal representation. However, we still have the potential rank deficiency of the spherical angle method, combined with the expense of constantly rotating the data set to ensure that $w \neq 0$.

A BETTER REPRESENTATION

All of the methods described in the last section have weaknesses that reduce their utility for geometric fitting. In this section, we describe a method that seems better suited to fitting applications. Our representation locates the axis with two parameters and orients it with two parameters. We will discuss the orientation parameters first, since they are simpler to deal with.

Better Spherical Angles

The orientation parameters are a simple change to the spherical angle method of the last section. We observed that the problem with spherical angles is that ϕ becomes indeterminate when $\sin\theta$ approaches zero. We can avoid this problem by using an *adaptive parameterization*, as follows. If $\sin\theta$ drops below a certain threshold (say, 0.5) then we switch from the conventional spherical angles as given in (6) and (7) to angles ξ and η that use the $+x$ axis as the pole:

$$n = \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} \cos\eta \\ \cos\xi\sin\eta \\ \sin\xi\sin\eta \end{pmatrix} \quad (8)$$

and the inverse mapping:

$$\begin{pmatrix} \xi \\ \eta \end{pmatrix} = \begin{pmatrix} \tan^{-1}(w/v) \\ \cos^{-1}(u) \end{pmatrix} \quad (9)$$

with ranges analogous to those of (7). If $\sin\eta$ becomes small, we merely switch back to conventional spherical angles. (There is no advantage in introducing a third axis direction, since we are merely seeking a direction orthogonal to the current axis.) We will call the conventional representation *the ϕ-θ mode* and the alternative representation *the ξ-η mode*.

The only subtlety in this representation is that the thresholds for $\sin\theta$ and $\sin\eta$ must be well-separated, to avoid frequent changes of representation mode near the cross-over points. The thresholds mentioned above, $\sin\theta<0.5$ and $\sin\eta<0.5$, correspond to a declination of 30° from the respective poles. This neatly divides the range between the x and z axes into thirds and provides a suitable "buffer zone" within which either representation is acceptable.

This method is reminiscent of the method of rotating the data (described in Section 2.4), but it can be handled in the representation by a simple mode flag. As with the method of rotating the data set, this method requires that the fitting routine be interrupted, here whenever the mode changes. However, instead of expensive rotations, all that is required is to change the order in which coordinates are indexed.

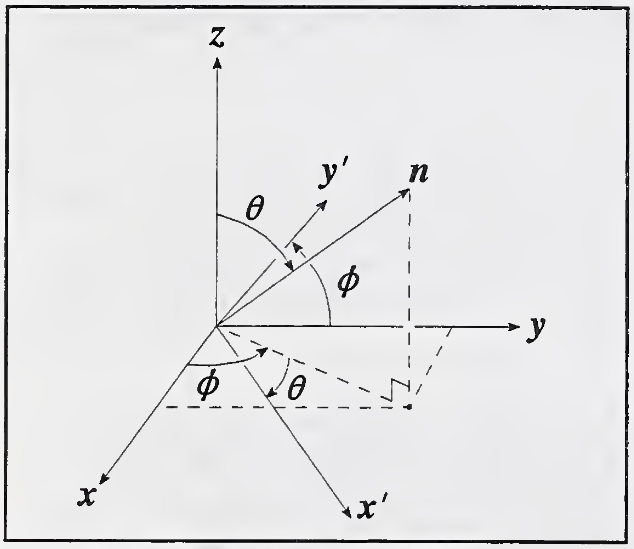

Figure 2 Illustration of local axes for the ϕ-θ mode. Note that y' is in the x-y plane. This construction greatly simplifies the expressions for derivatives (see the Appendix).

Locating the Axis

To reduce the number of location parameters, we start as before by constraining $\boldsymbol{a}$ to be the axis point closest to a fixed point $\boldsymbol{a}_0$. We then proceed differently. We observe that $\boldsymbol{a}$ lies in the plane through $\boldsymbol{a}_0$ having normal $\boldsymbol{n}$. Thus, we can locate $\boldsymbol{a}$ with two parameters by establishing a local coordinate system in that plane with origin at $\boldsymbol{a}_0$.

We will assume that $\boldsymbol{a}_0 = \boldsymbol{0}$. (There is no loss of generality, since we can always translate the problem by $-\boldsymbol{a}_0$ before fitting.) The method we use to establish the local axes depends on the mode of the orientation parameterization. We first present the method for the ϕ-θ mode.

We will designate the positive direction of the local axes for the ϕ-θ mode by $\boldsymbol{x}'$ and $\boldsymbol{y}'$, corresponding to the local x and y axes, respectively. To establish these directions, we first rotate the original coordinate system around the $+y$ axis by θ and then around the (original) $+z$ axis by ϕ. This rotates the $+z$ axis to be parallel to $\boldsymbol{n}$. The transformed $+x$ and $+y$ axes become parallel to, respectively, $\boldsymbol{x}'$ and $\boldsymbol{y}'$, as shown in Figure 2. It can be shown[9] that the transformed axes can be represented as:

$$\begin{pmatrix} \boldsymbol{x}' & \boldsymbol{y}' & \boldsymbol{n} \end{pmatrix} = \begin{pmatrix} \cos\phi\cos\theta & -\sin\phi & \cos\phi\sin\theta \\ \sin\phi\cos\theta & \cos\phi & \sin\phi\sin\theta \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}. \tag{10}$$

The location parameters for the representation are the $\boldsymbol{x}'$ and $\boldsymbol{y}'$ coordinates of $\boldsymbol{a}$ in the transformed coordinate system. We will call these e and f, respectively. We then have:

$$\boldsymbol{a} = e\boldsymbol{x}' + f\boldsymbol{y}'. \tag{11}$$

The inverse mapping is given by (7) and:

$$\begin{pmatrix} e \\ f \end{pmatrix} = \begin{pmatrix} \boldsymbol{a}\cdot\boldsymbol{x}' \\ \boldsymbol{a}\cdot\boldsymbol{y}' \end{pmatrix}. \tag{12}$$

For the ξ-η mode, we proceed similarly. We will rotate the $+x$ axis to be parallel to $\boldsymbol{n}$. We will designate positive direction for the local axes for the ξ-η mode as $\boldsymbol{x}''$ and $\boldsymbol{y}''$, corresponding to the rotated y and z axes, respectively. The local coordinates of $\boldsymbol{a}$ for the ξ-η mode will be called g and h. The appropriate transformation is:

$$\begin{pmatrix} \boldsymbol{n} & \boldsymbol{x}'' & \boldsymbol{y}'' \end{pmatrix} = \begin{pmatrix} \cos\eta & -\sin\eta & 0 \\ \cos\xi\sin\eta & \cos\xi\cos\eta & -\sin\xi \\ \sin\xi\sin\eta & \sin\xi\cos\eta & \cos\xi \end{pmatrix}. \tag{13}$$

We then have:

$$\boldsymbol{a} = g\boldsymbol{x}'' + h\boldsymbol{y}''. \tag{14}$$

The inverse mapping is given by (9) and:

$$\begin{pmatrix} g \\ h \end{pmatrix} = \begin{pmatrix} \boldsymbol{a}\cdot\boldsymbol{x}'' \\ \boldsymbol{a}\cdot\boldsymbol{y}'' \end{pmatrix}. \tag{15}$$

FITTING CONSIDERATIONS

Computing a fit using our representation can be quite complex. Generally, for an arbitrary point $\boldsymbol{q}$, two distances (shown as d and l in Figure 1) and their gradients (derivatives with respect to the fitting parameters) must be calculated by a fitting routine. Fitting routines that build up gradient estimates numerically must discard those estimates whenever the representation mode is switched. (The gradients could perhaps be transformed, but the gradient estimates are generally not accessible to users of packaged optimization routines.) Thus, fitting can be done quicker by routines that make use of explicitly computed derivatives of d and l. For reference, expressions for these derivatives are derived in the Appendix.

Optimization routines that accumulate gradient information numerically based on values of the objective function being minimized may be unreliable if the representation mode is changed during a fit and the routine is not restarted. One way to avoid this danger is to start by computing an approximate fit using one of the standard methods described earlier and then using that result as a starting guess for a final fit with our representation, using the representation mode for which the initial fit is farthest from the switching threshold. So long as the first fit is reasonably good, it will be unlikely that the representation mode will have to switch during the calculation of the second fit. The cost of the extra fitting is quite small—since the initial fit is (presumably) close to the optimum, the second fit using our representation will need to do very little work to arrive at a solution. The benefit, however, is that our representation assures good numerical stability and tighter bounds on the sensitivity of the result.

In our experiments, described below, the representation mode did not have to change during the fitting process. Since the initial guess for each fit was reasonably good, the modes chosen before starting the iterations remained acceptable throughout the fitting. This behavior seems quite reasonable for this class of iterative algorithms. Other procedures, such as simulated annealing, may have behavior requiring more mode switching; however, we have no experience to guide us in these matters.

EXPERIMENTAL RESULTS

We implemented a cylinder-fitting routine using our new representation and compared it to a similar routine that uses the naive representation. Experiments show that our representation can reduce the effects of numerical roundoff errors by an order of magnitude and produce fits for which tighter confidence intervals can be estimated. This section describes our implementations and the results we obtained.

The Fitting Algorithm

Both fitting routines make use of a standard Levenberg-Marquardt minimization algorithm, an example of a "trust region" search procedure. In broad terms, the algorithm proceeds in two phases. The first phase computes a direction (in the space of fit parameter values) in which to search. The second phase searches in that direction within the limits of the trust region radius for a point with a smaller sum of squared residuals. These two phases alternate until some convergence criterion is met (typically, that the step taken in the second phase is very small).

The algorithm is outlined in somewhat more detail below. Our implementation is based on a common formulation of the Levenberg-Marquardt algorithm due to Nash[10]. First, we must explain some terminology used in the description. In our experiments, a cylinder is represented by its axis (in either the naive or the new representation) and its radius. These parameters are collected together into a vector $\boldsymbol{p}$. When using the naive representation, the constraints $\|\boldsymbol{n}\|=1$ and $\boldsymbol{a}\cdot\boldsymbol{n}=0$ are enforced at key points in the algorithm. When using our new representation, the mode is checked and, if necessary, changed at the same key points. These processes are both identified in the description as "normalizing" the parameters.†

In our algorithm, the representation mode is changed in only two places: when new gradient information is about to be calculated and when the routine is about to converge. The mode is changed if the declination angle is with 30° of 0° or 180°. The change is done by transforming the parameter vector in place. The two places at which the mode can change were chosen for an important property: at those points, no prior gradient information will be further used.

The algorithm also makes use of functions to evaluate residuals and their partial derivatives with respect to the fitting parameters. In the description, the residuals are denoted by a vector-valued function, $f(\boldsymbol{p})$. The i^{th} component of $f(\boldsymbol{p})$ is the i^{th} residual. The partial derivatives of the residuals are represented as a matrix-valued function $\boldsymbol{F}(\boldsymbol{p})$; the element at row r and column c is the derivative of the r^{th} residual with respect to fitting parameter c. The matrix $\boldsymbol{I}$ is the identity matrix (of size understood by context) and the notation "diag(...)" refers to the diagonal matrix with diagonal elements indicated by the values in parentheses. In the description, a "T" superscript denotes matrix transpose, and vectors are treated as single-column matrices.

We can now present the algorithm used in the experiments.

Modified Levenberg-Marquardt Algorithm *The Levenberg-Marquardt algorithm due to Nash is modified in Steps 2 and 5 to normalize the fit parameter vector* $\boldsymbol{p}$, *as described above. The representation mode is changed only during normalization, and then only when the declination is within 30 ° of the pole.*

Step 1 [Initialization]. Set $\boldsymbol{p}$ to an initial guess for the fit. Set u=0.0001. *(The components of vector* $\boldsymbol{p}$ *vary according to the representation. The parameter u arises from the Lagrange multiplier for the trust region constraint; see, e.g., Moré*[11]*.)*

Step 2 [Determine a search direction]. Reduce u by a fixed factor, normalize $\boldsymbol{p}$, set $\boldsymbol{C}=\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}$, $\boldsymbol{v}=\boldsymbol{F}^{\mathrm{T}}\boldsymbol{f}$, $\boldsymbol{d}=\boldsymbol{p}$, and $s=\boldsymbol{f}\cdot\boldsymbol{f}$. *(F and f are evaluated at* $\boldsymbol{p}$*. s is the sum of squared residuals.)*

Step 3 [Compute a step along the line]. Increase u by a fixed factor, set $\boldsymbol{A} = \boldsymbol{C} + u(\boldsymbol{I} + \text{diag}(c_{11}, c_{22}, ..., c_{nn}))$, and solve the system $\boldsymbol{Ax}=-\boldsymbol{v}$ for $\boldsymbol{x}$. *(The system is solved using the Choleski decomposition of* $\boldsymbol{A}$*.* $\boldsymbol{x}$ *is the step to be taken.)*

Step 4 [Find result of step]. Set $\boldsymbol{p}=\boldsymbol{d}+\boldsymbol{x}$ and $t=\boldsymbol{f}\cdot\boldsymbol{f}$. *(f is evaluated at* $\boldsymbol{p}$*, which is the point after the step. t is the sum of squared residuals after the step.)*

Step 5 [Test results]. Check for convergence; if converged, normalize $\boldsymbol{p}$ and report $\boldsymbol{p}$ as the fit. Otherwise, if $t \geq s$ go to Step 3 *(the step was not an improvement)*; else go to Step 2 *(find another search direction).*

This algorithm is reasonably efficient and very compact in terms of memory usage.

The main issue with which we are concerned is the difficulty of solving the matrix equation

†If we were to implement a Levenberg-Marquardt algorithm that accumulated derivative information numerically, normalization would require that the algorithm be terminated and, if necessary, restarted at the key points.

in Step 3. The numerical accuracy of the fit and the sensitivity of the fit to perturbations in the data (which is reflected in the confidence limits) depend on the condition number of matrix A at the last iteration of the algorithm.† With the naive representation, the matrix C can be singular (because of the redundancies in the parameters), so the condition number of A can be very large, especially when u is small. We investigated these effects through a series of fitting experiments.

Experimental Results

Our experiments consisted of generating data sets representing "noisy" cylinder data, fitting the data using both the naive and new axis representations, and comparing the results using the methods of the NIST Algorithm Testing and Evaluation Program for Coordinate Measurement Systems[13]. Fitting was done in 64-bit IEEE real arithmetic for which the unit roundoff was about 2.2×10^{-16}. The fitting algorithm was run to full machine precision.

We generated 92 data sets for 14 different cylinders of 1 m radius each. The data were all generated by sampling the cylinders at 10 cross sections, with 12 equally spaced points at each cross section. We then perturbed each data point in a direction uniformly distributed on the sphere by an amount uniformly distributed between 0 and 50 mm. This was done five times for each cylinder. Data sets for 12 cylinders were left unperturbed. The cylinder had lengths of 1, 10, and 100 m. Six cylinders were sampled along an 80° arc at each cross section, 2 along 120° of arc, and 6 around the full circumference. Six cylinders were aligned with the z axis, 4 with the x axis, and 4 at 45° between. This exercised both modes of the new representation.

The fits generated with the two representations were then compared to one another. The comparisons were done by picking first one fit and then the other as the "reference." The pair-wise comparisons consisted of evaluating the largest separation of the assessed axis from the reference axis over the range of the data, the angular tilt between the axes, and the difference in radii. The differences were, as expected, extremely small; the root-mean-square values for all data sets were about 1.5×10^{-10} rad (3×10^{-5} arc second) tilt and about 4×10^{-10} m separation and radius difference. These are regardless of which fit was used as the reference. (Note that the differences, while small, are six orders of magnitude larger than the unit roundoff.)

Of principal interest, however, are the effects of the representation on the uncertainties of the comparison. The comparison uncertainties are a function of the uncertainties associated with the fits that were used as the reference. The reference fit uncertainties, in turn, are estimated from bounds on the numerical errors in solving the matrix equation in Step 3 of the algorithm. For details, see Hopp and Levenson[13].

When the fits computed using the naive representation were used as the reference, the combined standard uncertainties of the comparison were:

3.5×10^{-9} m for separation

1.4×10^{-4} arc second for tilt

5.4×10^{-10} m for radius difference.

When the fits computed using the new representation were used as the reference, the combined

†It should be noted that estimating the uncertainty of the fit by the matrix condition number alone can be unduly pessimistic[12]. This is, however, common practice and we used it consistently in our experiments.

standard uncertainties of the comparison were:

1.8×10^{-10} m for separation

2.7×10^{-5} arc second for tilt

1.6×10^{-10} m for radius difference.

This represents nearly an order of magnitude reduction in the uncertainty of the comparison. The same analysis done for the twelve unperturbed data sets (representing zero-residual problems) shows much more dramatic effects. For these data sets, the observed differences were zero, but the combined standard uncertainties of the differences were as follows:

Uncertainty	Representation	
	naive	new
separation (m):	1.0×10^{-8}	9.0×10^{-13}
tilt (rad):	4.1×10^{-7}	3.6×10^{-11}
radius (m):	2.0×10^{-6}	1.8×10^{-13}

These represent a reduction of four orders of magnitude in the uncertainties of the comparisons.

The differences in comparison uncertainties can be attributed entirely to a reduction in the uncertainties of the fits when our new representation is used. The results suggest that our new representation is much more stable numerically. The same mechanism (matrix condition) that affects the numerical uncertainty of the fit is used to determine confidence intervals for the fitted geometry. Thus, we can also claim that our new representation will reduce the estimated sensitivity of fits to perturbations in the data.

DISCUSSION

We have developed a representation for axes that has the following features:

- The axis is oriented by two parameters. The parameters are either the usual ϕ and θ spherical angles (respectively, the azimuthal angle from x and the declination from z) or alternative angles ξ and η (respectively, the azimuthal angle from y and the declination from x).
- The axis is located by two parameters defining the point of intersection of the axis with a plane determined by the axis orientation and a given, fixed point. The two parameters are the coordinates of the intersection point in a two-dimensional coordinate system defined in the plane.
- The representation is minimal and has no singularities. It requires a reparameterization to switch between two ways to represent the axis orientation. Thresholds can be set to separate the points at which the representation switches back and forth.
- Fits computed using this representation are less sensitive to perturbations in the data and to roundoff error than are fits computed using more common representations.

Our representation can be used when the axis location is of no interest—that is, problems involving orientation only—and for problems in which the point locating the axis is naturally determined in other ways. For instance, one might represent a cone by its apex point, axis direction, and cone angle. Representing the direction as described herein will avoid problems inherent in more traditional methods.

Similarly, a problem may have translational symmetry along an axis but no rotational symmetry (for instance, an extrusion). In such cases, the part of our method dealing with locating the axis may be used independently of the orientation representation (although the representation mode—ϕ-θ or ξ-η—must be maintained).

An axis has both rotational and translational (i.e., cylindrical) symmetry, so it might appear that our full representation is only useful for cylinders. This is not so. Some problems that might not appear to require the full generality of our representation may benefit from it. For a cone, for instance, the data may be far from the apex, so use of the apex in the representation can introduce roundoff problems in the computations. A numerically superior representation is to locate the axis by the axis point closest to the data centroid and locate the cone by the orthogonal distance from that point to the cone surface[14]. Thus, our representation may be fully applicable even when the problem is not cylindrically symmetric.

This representation should generalize to arbitrary dimension. Conceptually, the generalization is direct; however, finding a single, general method may be tricky—one would like to avoid many special cases. The case of two dimensions is straightforward: a line can be represented by a rotation followed by a displacement of the x axis. We note that the two-dimensional representation requires no mode switching. In higher dimensions, one would have to develop analogues to the spherical angle representations ((**6**) and (**8**)) and corresponding rotation matrices ((**10**) and (**13**)). We expect the main difficulty would be in identifying the singularities of the mappings and discovering appropriate mapping modes for avoiding them.

Finally, we wish to point out that the representation presented here is not a panacea. We might prefer it for geometric fitting problems, but it would not be our first choice for some other applications. Computer graphics, for instance, requires rapid evaluation of Cartesian coordinates on the modeled geometry. Our representation, with its use of transcendental functions, would not be a wise choice. As another example, a mechanical design engineer would like to be able to independently manipulate groups of parameters representing conceptually different attributes—such as location and orientation. Our representation fails in that regard, since the location parameters are functions of orientation.

However, the very features that make our representation weak for some applications are what make it highly suited for geometric fitting.

ACKNOWLEDGMENTS

I would like to thank Michael Pratt and Marsha Hopp for reading earlier versions of this paper and suggesting many useful clarifications to the presentation.

REFERENCES

1 Fisher, N. I., Lewis, T., and Embleton, B. J. J., *Statistical Analysis of Spherical Data*,

Cambridge University Press, New York, NY 1987.

2 Diaz, C., *Concept for an Algorithm Testing and Evaluation Program at NIST*, NISTIR 5366, National Institute of Standards and Technology, Gaithersburg, MD, January, 1994.

3 Boggs, P. T., and Donaldson, J. R., *The Computation and Use of the Asymptotic Covariance Matrix for Measurement Error Models*, NISTIR 89-4102, National Institute of Standards and Technology, Gaithersburg, MD, 1989.

4 Fuller, W. A., *Measurement Error Models*, John Wiley, New York, NY, 1987.

5 Wilkinson, J. H., *Rounding Errors in Algebraic Processes*, Prentice Hall, Englewood Cliffs, NJ, 1963.

6 Goldberg, D., "What Every Computer Scientist Should Know About Floating-Point Arithmetic," *Computing Surveys*, **23**(1):5-48, March 1991.

7 Bertsekas, D. P., *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York, NY, 1982.

8 Murthy, T. S. R., "A Comparison of Different Algorithms for Cylindricity Evaluation," *Int. J. Mach. Tool Design and Res.*, **22**(4):283-292, 1982.

9 Faux, I. D., and Pratt, M. J., *Computational Geometry for Design and Manufacture*, John Wiley & Sons, New York, NY, 1979.

10 Nash, J.C., *Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation*, Adam Hilger, Ltd., Bristol, England, 1979.

11 Moré, J. J., "The Levenberg-Marquardt Algorithm: Implementation and Theory," in *Numerical Analysis*, G. A. Watson, ed., Lecture Notes in Mathematics 630, Springer-Verlag, Berlin, 1977, pp. 105-116.

12 Winkler, J. R., *Singular Value Decomposition and Polynomial Basis Conversion*, Internal Technical Memorandum BRU/CAE/94:5, Department of Computer Science, Brunel University, London, November, 1994.

13 Hopp, T. H., and Levenson, M. S., "Performance Measures for Geometric Fitting in the NIST Algorithm Testing and Evaluation Program for Coordinate Measurement Systems," *NIST J. Res.*, to appear.

14 Forbes, A. B., *Least-Squares Best-fit Geometric Elements*, NPL Report DITC 140/89, National Physical Laboratory, Teddington, England, April, 1989.

APPENDIX

We here derive expressions for the derivatives of l and d for the two representation modes. We begin with the ϕ-θ mode. By inspection, the axes defined by **(10)** have the following derivatives:

$$\begin{array}{lll} \dfrac{\partial \mathbf{x}'}{\partial \phi} = \mathbf{y}'\cos\theta & \dfrac{\partial \mathbf{y}'}{\partial \phi} = -\mathbf{x}'\cos\theta - \mathbf{n}\sin\theta & \dfrac{\partial \mathbf{n}}{\partial \phi} = \mathbf{y}'\sin\theta \\ \dfrac{\partial \mathbf{x}'}{\partial \theta} = -\mathbf{n} & \dfrac{\partial \mathbf{y}'}{\partial \theta} = \mathbf{0} & \dfrac{\partial \mathbf{n}}{\partial \theta} = \mathbf{x}'. \end{array} \qquad \mathbf{(16)}$$

From **(11)** and **(16)**, we also have:

$$\frac{\partial a}{\partial \phi} = (ey' - fx')\cos\theta - fn\sin\theta \qquad \frac{\partial a}{\partial \theta} = -en. \tag{17}$$

We now consider l, the projected distance between q and a along the axis. Recall that we are assuming $a_0 = 0$, so $a \cdot n = 0$. Also, note that, from **(10)**, perforce $\|n\| = 1$. From the definition of l (Equation **(3)**) we have:

$$\frac{\partial l}{\partial \phi} = \frac{\partial}{\partial \phi} q \cdot n = q \cdot \frac{\partial n}{\partial \phi} = q \cdot y' \sin\theta. \tag{18}$$

Similarly,

$$\frac{\partial l}{\partial \theta} = \frac{\partial}{\partial \theta} q \cdot n = q \cdot \frac{\partial n}{\partial \theta} = q \cdot x'. \tag{19}$$

Since l is independent of e and f, those partial derivatives are zero.

We now turn to d. If d=0, then q is on the axis. It is not hard to see that in a small neighborhood of the axis parameters, d is proportional to the absolute value of each representation parameter. Thus, the derivatives at that point are undefined. We have found that defining all the partial derivatives of d to be zero whenever d=0 works well in practice.

Assuming then that $d \neq 0$, we will use the identity

$$\frac{\partial d}{\partial \phi} = \frac{1}{2d} \frac{\partial d^2}{\partial \phi} \tag{20}$$

and similarly for θ, e, and f. Differentiating **(5)** with respect to ϕ, we get:

$$\begin{aligned} \frac{\partial d}{\partial \phi} &= \frac{1}{2d} \frac{\partial}{\partial \phi}\left[(q-a)\cdot(q-a) - l^2\right] = \frac{1}{d}\left[-(q-a)\cdot\frac{\partial a}{\partial \phi} - l\frac{\partial l}{\partial \phi}\right] \\ &= \frac{-(q-a)\cdot(ey'-fx')\cos\theta + f(q-a)\cdot n\sin\theta - lq\cdot y'\sin\theta}{d} \\ &= \frac{(q-a)\cdot(fx'-ey')\cos\theta - l(q\cdot y' - f)\sin\theta}{d} \\ &= \frac{(q-a)\cdot((fx'-ey')\cos\theta - ly'\sin\theta)}{d}, \end{aligned} \tag{21}$$

the last step following from $a \cdot y' = f$. Similarly, use of **(11)** gives:

$$\frac{\partial d}{\partial \theta} = \frac{l[e - q\cdot x']}{d} = \frac{-l(q-a)\cdot x'}{d}, \tag{22}$$

$$\frac{\partial d}{\partial e} = \frac{1}{2d}\frac{\partial(q-a)\cdot(q-a)}{\partial e} = \frac{1}{d}(q-a)\cdot\frac{\partial(q-a)}{\partial e} = \frac{-(q-a)\cdot x'}{d}, \qquad (23)$$

and

$$\frac{\partial d}{\partial f} = \frac{-(q-a)\cdot y'}{d}. \qquad (24)$$

For the ξ-η mode, we have the following derivatives:

$$\begin{aligned} &\frac{\partial x''}{\partial \xi} = y''\cos\eta \quad && \frac{\partial y''}{\partial \xi} = -x''\cos\eta - n\sin\eta \quad && \frac{\partial n}{\partial \xi} = y''\sin\eta \\ &\frac{\partial x''}{\partial \eta} = -n && \frac{\partial y''}{\partial \eta} = 0 && \frac{\partial n}{\partial \eta} = x''. \end{aligned} \qquad (25)$$

The partial derivatives of l and d for the ξ-η mode then follow directly using the same chain of operations as for the ϕ-θ mode. For l, we obtain:

$$\frac{\partial l}{\partial \xi} = q\cdot y''\sin\eta, \qquad (26)$$

$$\frac{\partial l}{\partial \eta} = q\cdot x'', \qquad (27)$$

and

$$\frac{\partial l}{\partial g} = \frac{\partial l}{\partial h} = 0. \qquad (28)$$

Regarding d in ξ-η mode, when d=0, all its derivatives are defined to be zero. When $d\neq 0$, we obtain, as before:

$$\frac{\partial d}{\partial \xi} = \frac{(q-a)\cdot((hx''-gy'')\cos\eta - ly''\sin\eta)}{d}, \qquad (29)$$

$$\frac{\partial d}{\partial \eta} = \frac{-l(q-a)\cdot x''}{d}, \qquad (30)$$

$$\frac{\partial d}{\partial g} = \frac{-(q-a)\cdot x''}{d}, \qquad (31)$$

and

$$\frac{\partial d}{\partial h} = \frac{-(q-a)\cdot y''}{d}. \qquad (32)$$