1.0 Introduction

Many objects in both the man-made world and nature exhibit some degree of symmetry. Formally, an object exhibits symmetry whenever it can be divided into two or more parts which can be permuted by the application of certain isometries which leave the original object unchanged [17]. Two symmetry operations for planar figures are reflection and rotation. The letter M is a reflexive symmetry while the letter S is a rotational symmetry. The letter H is both laterally (reflexive) and rotationally symmetric. If a planar figure is arbitrarily rotated in three dimensions and projected orthographically onto the image plane, the projected figure is generally no longer reflexive symmetric. Under orthographics projection, a right angle can project onto a right, acute, or obtuse angle in the image plane. Consequently, the reflexive symmetric contour in figure 1a projects onto the *skew symmetric* contour shown in figure 1b, where *skew symmetry* is defined in [15] as a symmetry not necessarily perpendicular to the axis, but at a fixed angle to it. A skewed symmetry defines two directions, a skewed-symmetry axis and a skewed-transverse axis, as shown in figure 1b. In the real world, skewed contours occur frequently. Figure 2a is an edge image of a PUMA robot arm from the Columbia Vision/Robotics Lab, while figure 2b is a skewed contour hand-segmented from the same image with recovered skew axes shown (using the moment-based method described in section 3). Similarly, figure 3a shows a SUN4 SPARC station in the lab, figure 3b shows a hand-segmented skewed contour from the figure and figure 3c shows the *deskewed* contour with recovered axes of skew.

This paper concerns itself with finding the axes of skew symmetry given a contour that is the orthographic projection of a reflective symmetric planar curve. Often the solution will not be unique and several such solutions will exist, in which case an algorithm finding *all* such solutions is desired. This paper does not consider other classes of symmetry axes that have been defined in the context of ribbons (see [26]) and generalized cylinders (see [25]). An algorithm that finds axes of skewed symmetry is highly desirable since it is an important step in recovering the coordinate system of the projected figure. If the planar figure is part of an object having a 3D orthogonal coordinate system, as shown in figure 4, then a skewed axis pair make up 2 of the 3 projected axes of the object-centered coordinate system. If the 3rd image axis can be recovered, as in the case of straight homogeneous generalized cylinders [22], the transformation between viewer-and object- centered coordinate systems can be computed (up to a sign factor) [1]. Thus, determining the orientation of skewed axes is important to the shape recovery process.

The term skew symmetry was first defined by Kanade in [15]. That paper was primarily concerned with the problem of gradient ambiguity, that is, given the skew symmetric projection of a planar reflexive symmetry, what is the gradient value of the plane in which the symmetric figure lies? In that paper, Kanade shows that there is a hyperbola in the gradient space on which that gradient must lie. Various heuristics have been proposed to uniquely determine the gradient value [3],[13],[15].

In addition to gradient ambiguity, there is generally a second class of amiguity in the skewed figure. This ambiguity is called *skew ambiguity* since it is the ambiguity involved in determining which of several pairs of skew axes is the correct one. This paper does not discuss methods for resolving gradient ambiguity as this problem is highly context-sensitive and, in general, cannot be solved uniquely from a single skewed contour. Although heuristic methods (a.k.a., constraints) have been used in the past to solve this problem, the necessary constraints are often readily available from the image itself by considering adjacent contours. Constraining the gradient ambiguity by percolating constraints from adjacent contours is discussed in [28]. The context-sensitive nature of resolving gradient ambiguity for skew-symmetric axes is studied in [10]. This paper is concerned with the problem of finding axes of skew. This problem usually has a very finite set of solutions so our interest is in recovering all valid pairs of skew axes, rather than in finding a unique solution to the problem. Determining a *unique* skew pair solution from this set of solutions usually involves considering constraints from adjacent contours, a topic beyond the scope of this paper.

Section 2 of this paper discusses the motivation for detecting skew symmetric axes, some preliminaries relating image skew to orthographic projection, and previous related work. Next, a global moment-based method is presented. A motivation for the method is provided, and examples of the method under noise and occlusion are given. The problem of bias can be quite extreme when computing with higher order moments and this is also studied (section 3). In section 4, a local method is presented for recovering the skew axes that provides a tangential constraint between corresponding skew pairs. Results are shown for both synthetic and real images. Detecting skew contours is a prerequisite for finding the axes of skew. A method for detecting skew contours using a projective quasi-invariant is discussed in section 5.

The two methods for finding axes of skew introduced in this paper are part of a system named SYMAN, which is an acronym for SYMmetry ANalyzer. Elements of the current system are discussed in section 6. Finally (section 7), there is a discussion of conclusions and future work.

2. Motivation and Previous Work

The motivation for this work, finding skew symmetric axes, is twofold. On the one hand, finding the underlying axes of a skewed symmetric shape is a function that humans seem very adept at [28]. A theoretical understanding of how such axes might be determined is thus desirable. On the other hand, many of the shapes that one comes across in the real world exhibit some degree of symmetry and understanding how such symmetries can be determined from their projected images can be used as part of a general shape recovery algorithm. In addition, the algorithm described in this paper is parallelizable and requires only $O(log\ n)$ time on a pyramid machine, making it quite feasible to run in real-time [7],[14].

There has been considerable work in the area of resolving gradient ambiguity once the skew symmetric axes have been detected. Kanade [15] proposed that the correct gradient was the one on the hyperbola that minimized the slant. In [3], it is argued that the minimal slant heuristic is not consistent with human visual perception. Similar arguments are made in [28] and [9]. Brady and Yuille [3] proposed that the gradient that extremized shape compactness was the one that better correlated with human performance. The extremum principle of Brady and Yuille for finding shape from contours interprets a skew symmetric figure as a tilted symmetric figure. This is equivalent to both selecting a pair of skew axes (resolving skew ambiguity) and selecting a particular planar orientation (resolving gradient ambiguity). Of all the possible original figures, the most compact figure is chosen, where compactness is defined as the ratio of area over perimeter squared. When there are a finite number of axes of skewed symmetry it seems from the analysis in [3] that only one such solution is found by extremizing compactness, although in [9] the case is made that several sets of axes of skewed symmetry may be detectable due to the presence of several local extrema of compactness. An ellipse is always interpreted as the projection of a circle even though this is not always consistent with human perception, as noted in [30]. The method of extremizing compactness also does not correctly interpret a sheared box correctly since each parallelogram is interpreted as a sheared box even though it results in a globally inconsistent interpretation. Other heuristics for resolving gradient ambiguity have been proposed [13]. More recent results indicate that accurate estimation of surface gradient requires constraints from adjacent surfaces [10]. In any event, previous work on resolving gradient ambiguity either assumed the skewed axes were known [15] or solved for the gradient directly without ever explicitly detecting the skew axes [3].

Finding axes of skewed symmetry has been studied by several researchers [9],[21],[27],[32]. In [9], a moment-based method is used. Freidberg makes use of the second order moments of the skewed symmetry contour. From these moments, a constraint which he terms the *Fundamental Symmetry Constraint* is derived. His method involves searching along two one-dimensional parameter spaces, rotation space and skew space, subject to the Fundamental Symmetry Constraint. At each point selected in rotation-skew parameter space a heuristic measure of symmetry is applied. Extrema of the symmetry measure are selected as candidate solutions. This method is computationally expensive to compute since a search is required through rotation and skew space. In addition, it is easy to detect false solutions, as illustrated in [9] (see figure 14c).

In [21], a local method is used. A necessary condition is derived for pairs of points that are the projections of planar symmetric points in the scene. This local method exhaustively compares pairs of contour points to determine if the necessary condition is satisfied. It is similar to the approach used in [2] to detect smooth local symmetries. This method has the advantage that it does not suffer from occlusion. On the other hand, this method requires computing image curvatures and is necessarily sensitive to image noise. In addition, the necessary constraint relies on a non-zero curvature at both skew points, as a ratio of curvatures is taken. In the case of zero-curvature contours, that come up frequently in the real-world, (i.e., terminal, desks, polygons, etc.) this method fails to correctly detect the axes of skew symmetry (see [12]). Also, since the method is local, in principle a pairwise comparison is required among every pair of contour points, requiring an $O(n^2)$ algorithm, (although the method of projections [20] is suggested to reduce the algorithm's complexity). An implementation of the algorithm on some real edge images is given in [21].

A Hough-based method for finding the skewed axes is proposed in [32], but no results are given.

In [27] a method is proposed that consists of considering the skewed contour over projection space. For each projection, corresponding pairs of points are matched and their midpoint is taken. The correct projection angle is the one whose midpoints provide the best-fitting line. Their implementation does not actually use the points themselves but rather B-splines that were fit to the contour. Aside from the time complexity involved in a local method that must consider all possible angles of projection, many, if not most, of the skew symmetries in real images involve some degree of skew ambiguity, i.e., more than one set of skew axes is possible. It is not clear how the method in [27] deals with such skew ambiguity. There is a robusteness to the method, though, in that it does not require computation of any derivative information, tangents or curvatures, in order to find the skew axes.

2.1 3D Rotation and Oblique Coordinate Transformation

Skew symmetric figures are planar figures in the image space that are generated from planar symmetric figures by one of two processes :

(1) rotation in three dimensions followed by orthographic projection onto the image plane;

(2) oblique coordinate transformation followed by rotation in the image plane.

Translation invariance is assumed in this paper. This is acheived by measuring all coordinates relative to the centroid (i.e., center of mass) of the figure. The centroid is invariant under affine transformations, such as skew and rotation. In addition, the centroid of the figure has the property that all axes of symmetry must pass through it.

Under projection, information about the underlying symmetric shape and its orientation are lost. Thus, skew symmetric figures are ambiguous since there are many figure-transformation pairs capable of generating a given skew symmetric figure. This loss of information, as noted in [9], is the result of two distinct types of ambiguity : *gradient* ambiguity and *shear* ambiguity.

Gradient ambiguity exists in the process of projection onto the image plane because there exists an infinite number of rotations that project the x and y axes orthographically to the same pair of skew symmetric axes. The solutions for the plane orientation are constrained to lie along the locus of a hyperbola in the gradient space [15]. Shear ambiguity, as described in [9], is defined as the uncertainty of the projection of the real symmetry axis due to the skewed figure having multiple axes of skewed symmetry. Certain figures exhibit symmetric ambiguity. A circle, for example, has an infinite number of symmetry axes, while a square has four. Under skewed symmetry unambiguous symmetric figures, such as an isosceles triangle, can become shear ambiguous, see figure 5. Figures of finite ambiguity, such as an ellipse having two axes of symmetry, can become infinitely shear ambiguous under shear symmetry.

To skew a symmetric image in the plane, a skew coordinate transformation is required. The skew transformation is given by $pT = q$, where

$$T = \begin{bmatrix} \cos\alpha & \sin\alpha \\ \cos\alpha \, cot\ \beta - \sin\alpha & \sin\alpha \, cot\ \beta + \cos\alpha \end{bmatrix} \tag{2.1}$$

Each pair of skew axes that is found corresponds to a skew coordinate transformation from a symmetric figure into the given image skew symmetric figure. Where multiple axes of skewed symmetry exist, it is desirable to find all such solutions.

2.2 A Symmetry Hierarchy

In this section, we define a symmetry hierarchy that is useful in analyzing and detecting skew symmetries. In particular, since we intend to use analytic constraints to solve for the skew axes directly, it is important to be able to infer something about the degree of the reflexive symmetry in the original figure from its projection. Is there a single underlying axis of symmetry; are there two such axes? Can some sort of hierarchy be established so that figures with a greater degree of symmetry project to contours with a greater degree of skew symmetry?

In developing such a hierarchy, it is possible to focus either on the *real* symmetric figure or on its projected skewed symmetric contour.† For example, it is easy to define a circle as being at the top of such a symmetry hierarchy. It has an infinite number of symmetry axes, and its projection (an ellipse) has an infinite number of skew symmetric axes. Next, however, consider an ellipse versus a regular octagon. The octagon is more symmetric than the ellipse if one is considering axes of symmetry; it has 12 such axes as opposed to 2 axes of symmetry for an ellipse. On the other hand, when considering the projections of these figures, a skewed ellipse (still an ellipse [18]) has an infinite number of skew symmetric axes while the octagon has only 12 axes of skew. Obviously, the relationship between a figure's degree of symmetry and the degree of skew symmetry of its projection is nontrivial.

Before discussing degree of symmetry any further, we present some definitions. Without loss of generality, we assume the curve C is aligned with respect to the y axis, i.e., its axis of symmetry is coincident with the y axis.

Definitions: Let C be some finite set of curves in the xy plane. If for every point $(x,y) \in C$ there exits a point $(-x,y) \in C$ then C is *reflexive symmetric*. If C is reflexive symmetric and for every point $(x,y) \in C$ there exists a point $(x,-y) \in C$ then C is *doubly reflexive symmetric*. If for every point $(x,y) \in C$ there exists a point $(-x,-y) \in C$ then C is π- rotationally symmetric. Hereafter, we will often refer to π-rotational symmetry as simply rotational symmetry.

From these definitions, some propositions can be derived:

Proposition 2.1: If a curve C is doubly reflexive symmetric then it is also π-rotationally symmetric.

Proof: Since C is doubly reflexive symmetric, for every point $(x, y) \in C$ there exist points $(-x, y) \in C$ and $(x, -y) \in C$. But if $(x, -y) \in C$ and C is reflexive symmetric, then $(-x, -y) \in C$.

Proposition 2.2: The property of π-rotational symmetry is invariant to the affine transformations of skew, rotation, and scaling.

Proof: Let T be a transformation matrix that skews, rotates, and rescales a point p into a point q. We can write this as $p\ T = q$. But then $(-p)\,T = -(p\ T) = -q$.

From Proposition 2.2 and the relationship between oblique coordinate transformation and 3D rotation followed by orthographic projection, we have the following corollary:

Corollary: If a planar 3D scene curve C is doubly reflexive symmetric then its 2D orthographic image projection C_I is π-rotationally symmetric.

Of course, the converse of Proposition 2.1 is not true. A curve can be π-rotationally symmetric but not doubly reflexive symmetric. An example of this is the sine function, which is rotationally symmetric about the origin but is not even singly reflexive symmetric, let alone doubly reflexive symmetric. The curve in figure 1a is doubly reflexive symmetric, while its projection (i.e., skewed contour), shown in figure 1b, is no longer doubly reflexive symmetric nor even singly reflexive symmetric. It is skew symmetric, however, and also π-rotationally symmetric. Generally, due to the compact nature of real-world objects, rotationally symmetric contours in the image plane are the projections of doubly reflexive symmetric shapes in the scene. This motivates a heuristic that we intend to use in analyzing symmetry. The heuristic, called the *rotational symmetry heuristic*, is stated as follows:

> "If a contour in the image is π-rotationally symmetric and can be interpreted as a doubly reflexive symmetric curve in the scene, then it is assumed to be the projection of a doubly reflexive 3D planar curve in the scene."

This heuristic would not include rotationally symmetric image contours that do not allow for such an

† In this paper, the term *figure* generally refers to the planar 3D curve in the scene while the term *contour* refers to the projected curve in the image plane.

interpretation (as the projection of a doubly reflexive 3D curve). Thus, the sine function, which cannot be interpreted as the projection of a double reflexive symmetry, is not included in the heuristic. But if the curve can be interpreted as the projection of a doubly reflexive symmetry in the scene, then it must have at least two axes of skew symmetry in the image (since a real symmetry axis in the scene *always* projects to a skew symmetric axis in the image). But then this heuristic is analogous to the Kanade heuristic [15], albeit in a more restricted domain, The skew symmetry heuristic of Kanade states that a skewed symmetry in the image plane is the projection of a real symmetry in the scene. We have restricted our focus to a rotationally symmetric, doubly (at least) skew symmetric contour. This type of curve is important because it comes up often in real images and because it has a moment-based signature that can be used to detect it (see section 5).

In practice, since doubly reflexive symmetric planar figures occur often, their projective contours deserve further study. Consider the family of superellipses defined by the equation

$$\underline{x}(\theta) = \begin{bmatrix} a_1 \cos^{\varepsilon}\theta \\ a_2 \sin^{\varepsilon}\theta \end{bmatrix} \qquad 0 \leq \theta < 2\pi$$

Examples of superelliptic curves are shown in figure 6, where $a_1 = 2$ and $a_2 = 1$n and ε takes on the values of 0.5, 1.0, 2.0, and 3.0. Skewed versions of these curves are shown in figure 7, where the angle of skew is 60 degrees with respect to the horizontal axis. It can be seen that only the skewed ellipse is still an ellipse. The superellipses shown in the figure, with the exception of the regular ellipse, have 4 pairs of skew-symmetric axes. Of the infinite family of superelliptic curves, it appears that the pure ellipse ($\varepsilon = 1$) is the only curve that retains the property of double reflexive symmetry under skew. The fact that a skewed ellipse results in another rotated ellipse is directly related to the fact that ellipses in the scene project orthographically to ellipses in the image plane. This is also true for hyperbolas [18]. It is easy to show, however, that the symmetry axes in the image ellipse are not the projections of the symmetry axes in the scene ellipse [10]. Ellipses (including circles) are the only simple *closed* curves known to the authors that retain the property of reflexive symmetry under skew.

Based on the preceeding observations, a symmetry hierarchy can be constructed. A circle has the highest degree of symmetry since it is infinitely symmetric and its projection is infinitely skew symmetric. An ellipse is further down the hierarchy since it is only doubly reflexive symmetric, with a projection of infinite skew symmetry. The pure ellipse is above regular doubly reflexive symmetric curves. Included in this class of regular double reflexive symmetry are members of the superelliptic family, other than the ellipse, whose projective curves are only 4th degree skew symmetric (see figure 7). Below this class in the hierarchy is the class of curves that are singly reflexive symmetric, but whose projections are multiply skew symmetric. All non-equilateral triangles are in this class. The lowest level in the hierarchy is the class of curves whose symmetry and projective skew symmetry are both of degree one.

Having constructed a symmetry hierarchy and some degree of correlation between the underlying real symmetry and the projected skew symmetry, we would like to be able to draw conclusions about the degree of real symmetry from the degree of projected skew symmetry. In general, certain conclusions about the underlying symmetry of the figure in the scene can be made. This topic, however, is deserving of further study. Methods of detecting skew symmetry in an image, a natural precursor to finding axes of skew, are discussed in section 5.

Any analytic solution to finding pairs of skew symmetry, i.e., skew symmetry axes and their corresponding transverse axes, must be concerned with the degree of the underlying skew symmetry. Clearly, if there are 5 pairs of skew axes, we cannot recover them all with a 4th order polynomial. As noted in [9], due to the potentially unbounded shear ambiguity, "general analytic solutions for axes of skew symmetry are unlikely to exist." Fortunately, however, with the exception of a few curves whose properties under skew are well-understood (e.g., ellipses), most real-world skew symmetric contours are of extremely limited degree. In particular, aside from circles, almost all real symmetries are of degree 1 or 2. Under skew transformation, these skew symmetries are generally of degree 4 or less. Ellipses are an exception, as they have an infinite degree of skew symmetry. Because ellipses project to ellipses, they can be detected in the image in a straightforward manner [32],[29], even under partail occlusion.

The analytic moment-based approach presented in section 3 presumes that the contour's skew symmetry is of degree 4 or less, unless it detects a pure symmetry under rotation (step 2 of the algorithm in section 3.2). Ellipses are detected at this step of the algorithm (step 2) and, consequently, do not violate the algorithm's underlying assumption that the degree of skew ambiguity does not exceed four.

2.3 Global Versus Local Methods

There appear to be two approaches to solving for axes of skew. Each of these approaches has its respective advantages and disadvantages. One approach is to look for a global technique that looks at the whole contour at once and, from certain properties of the contour, finds its axes of skew. This approach has the advantage of being efficient as it is not necessary to consider each pair of skew points individually. Conversely, any such method relies on the fact that the skew contour is fully in view and would suffer from the affects of occlusion. Both the method presented in [9] and that of the next section have the aforementioned advantages and disadvantages.

A local method, one that compares pairs of skew points, has the advantage that it does not rely on any general (global) properties of skewed contours. Thus, it is not likely to fail due to partial occlusion. On the other hand, pointwise matching of skew pairs is required, a very time-consuming process. In addition, there are many images where humans seem to have little difficulty in determing the direction of skew yet a pointwise matching among skew pairs is virtually impossible. An example of this is shown in figure 8. Here a local point matching among skew pairs is difficult due to the complex, non-convex nature of the contour. The moment-based technique of the next section, on the other hand, is able to find the axes of skew for such contours (see figure 18).

A robust skew symmetry recovery system could benefit from using both local and global methods. This is in fact how our symmetry analyzer SYMAN is constructed. Elements of the SYMAN system are described in section 6.

3.0 A Global Moment-based method

Any attempt at a moment-based solution to the problem of finding skew symmetric axes must proceed with caution. Clearly, moments do not retain all the information concerning the underlying contour. That having been said, the skew symmetric contours we wish to solve for are generally of extremely limited degree such that the information needed to find the axes of skew can, in fact, be encoded in the moments of the contour.

The work most closely related to our moment-based approach is that of Freidberg in [9]. He attempted to use moments to find the axes of skew. A solution consists of a pair of values (α_i, β_i) which define a trasformation matrix T_i, given by equation (2.1). As stated earlier, given a skewed contour C there is often a set of such (α, β) solutions. When more than one such solution exists, we are not interested in selecting a unique solution, but rather in finding all such solutions. Freidberg limited his use of moments to that of second order. This only allowed him one constraint, which he calls the *Fundamental Symmetry Constaint*. Since there are two parameters to be recovered, that of rotation and skew, this constraint did not yield a solution for the axes of skew but rather constrained the search space. Other related work in moment-based solutions to affine transformations has involved domains where there exist *apriori* knowledge of either the shape of the original figure or the parameters of the imaging process. Using methods from tensor calculus, Cyganski [8] shows how to solve for the affine transformation relating the original figure to the image contour when the 3rd and 5th order moments are known.

Assume we are analyzing a skewed image contour C_I that is derived from a planar symmetric figure. We wish to find a skew-rotation transformation T_j that takes symmetric figure C_j into skew symmetric image contour C_I. The transformation matrix can in fact be decomposed into a skew followed by a rotation. We have found this problem easier to solve through decomposition. The transformation matrix T can be rewritten as the set of transformation matrices S, a pure skew transformation, followed by R, a pure rotation. These matrices are given by

$$S = \begin{bmatrix} 1 & 0 \\ cot\ \beta & 1 \end{bmatrix} \tag{3.0.1}$$

and

$$R = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \tag{3.0.2}$$

In the next section, we decompose the problem of recovering axes of skew to two subproblems, recovering the rotation (in the image plane) of a pure skew curve and then finding the skew of a reflexive symmetric curve under pure skew (with no rotational component).

3.1 Decomposing the Problem

Rather than being given skewed and rotated image contour C_I and solving for the transformation matrix T (which is equivalent to solving for the skew symmetric axes), we wish instead to solve first for the rotation component α of rotation matrix R. Having solved for matrix R, we will then attempt to solve for skew matrix S. Consider contour C_I (figure 9a), a skewed and rotated version of some symmetric figure C_k (figure 9c). Let the set of moments for C_I be given by {P}, where the $(i,j)^{th}$ moment of C_I is p_{ij}. Next, let the contour C_{S_k} (figure 9b) be a derotated version of C_I, such that it is a pure skew of the underlying symmetric figure C_k and let α_k be the angle of derotation. Let the set of moments for C_{S_k} be given by {N}, where the $(i,j)^{th}$ moment of contour C_{S_k} is given by $n_{i,j}$. Let figure C_k be a deskewed version of C_{S_k}, such that it is a reflexive symmetric figure and let $cot\ \beta_k$ be the deskewing parameter. Let the set of moments for contour C_k be given by {M}, with the $(i,j)^{th}$ moment given by m_{ij}.

What relationship, if any, exists between these sets of moments? From the definition of reflexive symmetry given in section 2.2 it follows that, since C_k is a symmetric contour, all moments in {M} of the form m_{ij}, where i is odd, are equal to 0. Now consider the moments {N} of a pure skew. What can be said about the relationship among this set of moments? When a point (x,y) on contour C_j undergoes a pure skew transformation, given by a skew matrix S_j (equation (3.0.1)), it is mapped to a point $q=(x',y')=(x+y\,k,y)$, where $k=cot\ \beta_j$. The moment n_{11} of pure-skew contour C_{S_k} is given by $n_{11}=\sum(x'y')=\sum(x+yk)y=\sum xy+k\sum y^2=m_{11}+k\ m_{02}=k\ m_{02}$, since $m_{11}=0$. Next, consider moment n_{02}. Since $y'=y$, it follows that $n_{02}=m_{02}$. Thus, we have a ratio of moments given by

$$\frac{n_{11}}{n_{02}}=k. \tag{3.1.1}$$

Similarly, moment n_{12} is given by $n_{12}=\sum x'y'^2=m_{12}+k\ m_{03}=k\ m_{03}$. Again, since under pure skew $y=y'$, $n_{03}=m_{03}$. The moment ratio is given by

$$\frac{n_{12}}{n_{03}}=k. \tag{3.1.2}$$

Setting the left hand side of equations (3.1.1) and (3.1.2) equal and cross multiplying, we obtain a moment-based constraint

$$n_{11}\,n_{03} = n_{02}\,n_{12}. \tag{3.1.3}$$

It is easy to see that analogous constraints can be derived using other order moments in {N}. In general, we have a necessary constraint for pure skew symmetric figures (no rotational component) given by

$$n_{1,u-1}\,n_{0,v} = n_{1,v-1}\,n_{0,u} \tag{3.1.4}$$

Just as $m_{odd,j}=0$ was found to be a necessary constraint for moments of a symmetric figure, equation (3.1.4) is a necessary constraint among moments of a pure skew figure.

Is there a necessary constraint among moments of a general skew figure (skew and rotation)? If such a constraint existed it could be used to *detect* a skewed contour before solving for the parameters of skew and rotation. In the general case, we have not found any such constraint. For the special case of a doubly reflexive symmetry under skew and rotation, which transforms to a π- rotational skew, such a moment-based constraint does exist and can be used to detect skew contours in the image (section 5). In any event, necessary moment-based constraints exist for reflexive symmetric and pure-skew figures.

We are given the image contour C_I, which is believed to be the skewed and rotated contour of some symmetric figure C_j. Given constraint equation (3.1.3), how can we recover the rotational component α_j

?

Given image contour C_I, with corresponding set of moments {P}, we want to find the corresponding pure-skew contour C_{S_k}, with corresponding set of moments {N}, satisfying equation (3.1.3). Let $r = (x, y)$ be a point on image contour C_I and $s = (x', y')$ be a point on the derotated, pure-skew contour C_{S_k}. The inverse rotation matrix R^{-1} is just R^T given by

$$R^{-1} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \tag{3.1.5}$$

Since $r\,R^{-1} = s$, it follows that

$$s = (x', y') = (x, y)R^{-1} = (C_\alpha x + S_\alpha y, -S_\alpha x + C_\alpha y), \tag{3.1.6}$$

where $C_\alpha = \cos\alpha$ and $S_\alpha = \sin\alpha$. The moments of pure-skew contour C_{S_k} are given by {N}. The moments of image contour C_I are given by {P} and are directly computable from image contour C_I. From equation (3.1.6), we can write moment n_{11} as

$$\begin{aligned} n_{11} &= \sum x' y' = -S_\alpha C_\alpha \sum x^2 + (C^2{}_\alpha - S^2{}_\alpha) \sum x\,y + C_\alpha S_\alpha \sum y^2 \\ &= -S_\alpha C_\alpha p_{20} + (C^2{}_\alpha - S^2{}_\alpha)\, p_{11} + C_\alpha S_\alpha p_{02}. \end{aligned} \tag{3.1.7}$$

Since moment p_{ij} is computable directly from image contour C_I, the only variable in equation (3.1.7) is α. Thus, n_{11} can be written as $n_{11}(\alpha; p_{20}, p_{11}, p_{02})$, i.e., n_{11} is a function of α given the 2nd order image moments p_{20}, p_{11}, p_{02}. In fact, any moment n_{ij} of contour C_{S_k} can be written as a function of α and the $i+j$ th order moments of {P}. Thus, equation (3.1.3) can be written as an equation in α and the 2nd and 3rd order moments of {P}. From equation (3.1.3) and the preceeding discussion, we construct a function F given by

$$\begin{aligned} &F(\alpha; p_{20}, p_{11}, p_{02}, p_{30}, p_{21}, p_{12}, p_{03}) \\ &= n_{11}(\alpha; p_{20}, p_{11}, p_{02})\, n_{03}(\alpha; p_{30}, p_{21}, p_{12}, p_{03}) - n_{02}(\alpha; p_{20}, p_{11}, p_{02})\, n_{12}(\alpha; p_{30}, p_{21}, p_{12}, p_{03}) \\ &= 0 \end{aligned} \tag{3.1.8}$$

Thus, any solution α_k must satisfy equation (3.1.8). A nice property of this formulation is that the skew parameter β does not appear in equation (3.1.8). The moment constraint of equation (3.1.3), on which equation (3.1.8) is based, is only sufficient, not necessary, so a solution α_k to equation (3.1.8) may not result in contour C_{S_k} being a pure skew.

Doubly reflexive figures have the property that under orthography (alternatively, skew and rotation) their projective contours are π-rotationally symmetric, as shown in Propositions 2.1 and 2.2 . But a π-rotationally symmetric contour C has the property that, for every point $(x, y) \in C$, there exists a point $(-x, -y) \in C$. It is easy to see that for π-rotationally symmetric contours, for any $(i, j)^{th}$ moment where either i is odd and j is even or i is even and j is odd, $p_{ij} = 0$. But all 3rd order moments satisfy this property! Thus, for a π-rotationally symmetric contour (i.e., the projection of a double symmetric figure), equation (3.1.8) is always equal to zero regardless of the value of α. For π-rotationally symmetric image contour C_I, then, it is necessary to use 4th order moments (which are not, in general, equal to zero). A function G, similar to function F, can be derived using the general constraint of equation (3.1.4). The function $G(\cdot)$ is given by

$$\begin{aligned} &G(\alpha; p_{20}, p_{11}, p_{02}, p_{40}, p_{31}, p_{22}, p_{13}, p_{04}) \\ &= n_{11}(\alpha; \cdot)\, n_{04}(\alpha; \cdot) - n_{02}(\alpha; \cdot)\, n_{13}(\alpha; \cdot) \\ &= 0 \end{aligned} \tag{3.1.9}$$

Again, the skew term β is absent from $G(\cdot)$. Solving for either function $F(\cdot)$ or function $G(\cdot)$, depending on whether the image contour C_I is rotationally symmetric or not (which can be determined by computing the 3rd order image moments), allows SYMAN to solve for the rotational component of the skew transformation. Alternatively, the recovered value of α_k is the direction of the transverse axis. Using elimination theory, equations (3.1.8) and (3.1.9) can, respectively, be reduced to a cubic and quartic equation in $\cos\alpha$, for which a closed form solution exists. In practice, we solve for α using numerical root-finding techniques.

Once the zeroes of function $F(\cdot)$ (alternatively $G(\cdot)$) have been solved for, the skew parameter β needs to be recovered. This is considerably easier than solving for rotation component α. For a given value of recovered rotation parameter α, all the moments in {N} can be computed. But from equation (3.1.1), the value of $cot\ \beta$ is given by

$$cot\ \beta = \frac{n_{11}}{n_{02}}.$$

Having solved for the skew and rotation parameters, it is necessary to check if the recovered parameters (α, β) constitute a valid solution, i.e., will the inverse transformation T^{-1} when applied to contour C_I yield a symmetric contour? Since the constraints used in solving for the skew parameters are necessary, but not sufficient, such verification of a solution is necessary. This can be done by checking other moments of the recovered contour (i.e., the contour that results from an inverse skew transformation applied to the image contour). If the solution is valid, then all moments of the form $m_{odd,j}$ should equal zero. If the image contour was rotationally symmetric then, additionally, all moments of the form $m_{i,odd}$ should equal zero.

So far, we have only considered contours that are skewed and rotated versions of a reflexive symmetric planar figure. What if the image contour is either a pure skew or a pure rotation? Pure skews present no problem. The function $F(\cdot)$ will simply return a value of $\alpha = 0$. What about pure rotations ? In fact pure-rotation contours come up frequently in real images in the form of ellipses. It is almost invariably a skewed and rotated ellipse in the scene, but since an ellipse under skew is just a rotated ellipse, the skew component of the transformation is not in evidence. The constraint equation (3.1.2) is numerically unstable as the n_{03} moment tends to zero since it is dominated by aliasing and discretization error, rather than by its real value. If the denominator in equation (3.1.2) is exactly 0, we have a divide by zero condition. So an image contour that is a reflexive symmetry under pure rotation needs to be tested for *before* solving for the general case. Similar to the method described above for obtaining function $F(\cdot)$, a function $H(\cdot)$ can be derived that is a function of image moments {P} and rotational parameter α, and is equal to zero whenever α is a solution. Since we are interested in recovering axes of symmetry up to degree 4, it is sufficient to derive $n_{13}(\cdot)$, a fourth order moment of the derotated contour, as a function of image moments {P} and rotation angle α. The function $H(\cdot)$ is then set equal to n_{13} (alternatively n_{31}), which is zero when the derotated contour is symmetric.

3.2 The Global Moment-based Algorithm

The algorithm, presented in pseudo-code, is given below:

1. Compute moments (up to fourth order) of the image contour and normalize the moments with respect to the contour centroid.

2. Solve for the rotation parameter α assuming the contour is a pure rotation. Having solved for α, test if the contour is really a pure rotation.

3. Test if the contour is π-rotationally symmetric.

4. If (contour is π-rotationally symmetric)
   use 4th degree function $G(\cdot)$;
   else
   use 3rd degree function $F(\cdot)$;

5. Check if the solutions are valid.

To avoid fractional exponent terms, elimination theory techniques are used in deriving $F(\cdot)$ and $G(\cdot)$ (see [23],[5],[6]). Function F is a cubic univariate polynomial in x, where $x = \cos^2(\alpha)$. The coefficient terms in F are products of 2nd and 3rd order image contour moments. Function G is a quartic univariate polynomial in x, where, once again, $x = \cos^2(\alpha)$. The coefficient terms in G are products of 2nd and 4th order image contour moments. Since these two functions are univariate and either quartic or cubic in degree, they can be solved for in closed form. In practice, SYMAN uses a numerical iterative root-finding method, taken from Numerical Recipes [24].

The general method described above can be extended to higher degrees of skew ambiguity by using higher order moments. For example, if we are interested in recovering the skew axes for a skewed n-gon with degree of skew ambiguity equal to n (actually there are 2n solutions, but n of them are duals of the first n), the 4th order moments in equation (3.1.9) are replaced by nth order moments. A synthetic example of a skewed 9-gon and one of nine recovered solutions for the axes of skew (all of the recovered solution were approximately correct) is shown in figure 10. For real images, the higher the degree of moments used, the less reliable the recovered solution is likely to be. As a result, to achieve the greatest accuracy, the degree of image moments used, and the resulting degree of the polynomial that needs to be solved, should equal the degree of skew ambiguity in the image contour.

In practice, the functions $F(\cdot)$ and $G(\cdot)$ are nontrivial to derive. The C code for functions $F(\cdot)$ and $G(\cdot)$ is given in Appendix A.

Examples of the algorithm, both real and synthetic, are provided in the next section.

3.3 Experimental results

Consider the skewed and rotated superellipse of figure 11a. When the algorithm was run on fig 11a, a rotational symmetry was detected (i.e., 3rd order moments were approximately equal to 0). The fourth order equation, equation (3.1.9), was solved for resulting in four pairs of (α, β) solutions. Of course, of the 4 solutions only 2 are unique; the other two are duals of the first two, with the skew and transverse axes reversed. There are two ways to display solutions. One is to draw the recovered axes of skew on top of the skewed image contour; the other is to actually derotate and deskew the image contour. We will alternate between these two methods.†† Two solutions for the skewed contour of figure 11a are shown (in deskewed, derotated format) in figure 11b and figure 11c.

The contour of a synthetic skewed vase is shown in figure 12a. The vase does not test positive as a rotational symmetry (step 3 of the algorithm) so the third order equation, equation (3.1.8), is solved for. Of the three solutions returned, only one of them when tested is a valid solution. The skew and transverse axes for the correct solution are shown in figure 12b.

A triangle under skew and rotation is shown in figure 13a. It is not rotationally symmetric, so equation (3.1.8) is solved for. All three solutions of the equation represent valid solutions, as a triangle always has 3 sets of skew axes. The recovered axes of skew (shown as derotated, deskewed contours) are shown in figure 13b, figure 13c, and figure 13d.

Consider the skewed picture, taken from Mohan [16] (originally from Marr), shown in figure 14a. There are 4 solutions recovered by the algorithm (the 2nd two solutions are duals of the first two), two of which are shown in figures 14b and 14c. Another picture, also taken from [16], is shown in figure 15a. Note here also that the contour is discontinuous, the tangents are locally in random directions, and that the curvature, wherever it is defined, is zero. The symmetry of the figure is only perceptible at a larger scale. Here SYMAN detects a pure rotations and returns 2 solutions (step 2 of the algorithm). Both of these solutions, displayed as derotated contours, are shown in figures 15b and 15c.

Next, the algorithm was run on the dice image shown in figure 16. This image is due to Glenn Healey. The edge contours shown were detected with a Canny-like edge detector then labeled using a connected-component algorithm. Some of these connected components were then hand selected to run the algorthm on. It is important to note, however, that other than detecting the edges and selecting which components to run the algorithm on, no further edge processing was required. The recovered skew axes are

†† The derotated and deskewed contour should not be confused with the real 3D scene contour. There is still gradient ambiguity here that cannot be resolved without additional context or heuristics.

shown overlaid on top of the original intensity image. None of these skew contours had more than one solution for the axis of skew, thus, there is no skew ambiguity here. The results for this die image compare favorably with results for the same image in [21]. The algorithm given above, however, does not need to compute the curvature and tangent values of points along the contour while the algorithm given in [21] does. Also, the algorithm in [21] is local and so must consider the contour over projection space, with a time complexity of $O(n^2)$, while the algorithm here is $O(n)$, with potential time bound on pyramid hardware of $O(logn)$. Additionally, the algorithm here has no trouble recovering skew axes on the letter "A" on the die, while the results in [21] only show recovered skew axes for the suits (heart, club, ...), but not for the letter "A". This is because the necessary constraint in [21] relies on the curvature being non-zero, and cannot work on straight contours.

Consider the edge contour of a real leaf image shown in figure 17a. The algorithm does not detect a pure symmetry under rotation (step 2) nor does it detect a π-rotational symmetry (step 3), so a skew ambiguity of degree three or less is assumed (i.e., function F is solved for). Of the three solutions returned by the algorithm, one is shown (the one that is closest to the correct solution) in figure 17b.

An edge contour of a real image taken of a symmetric ink blot was already shown, as an example of where local methods do not perform well, in figure 8. A solution recovered by the algorithm is shown in figure 18.

Does the method suffer from occlusion? Clearly, any moment-based method will have trouble dealing with occluded contours. This is one of the strong reasons for local techniques. However, this algorithm does seem rather robust under partial occlusion. The results of a poorly detected superelliptic edge are shown in figure 19a. Two unique solutions (the other two are duals) are shown in figures 19b and 19c.

3.4 Bias in the method

Whenever one tries to obtain a solution by minimzing the value of a measure, as we have done in step 4 of the algorithm by finding the zeroes of the function $F(\cdot)$ (or alternatively $G(\cdot)$), there is a potential problem of bias [11]. It can be particularly acute if the measure relies on moment data that is subject to discretzation errors. The algorithm of the previous section decoupled solving for rotation from solving for skew. The rotational component is recovered by finding the angle α that transforms the image contour into a pure skew. The function $G(\cdot)$, for example, attempts to find a solution such that the ratio of $\frac{n_{11}}{n_{02}} = \frac{n_{13}}{n_{04}}$. Once the rotational component has been recovered, the algorithm then finds the skew component $cot\ \beta$ by computing either of the above ratios. If there were no errors introduced due to discretization errors this method would probably work fine. In practice, by making use of only certain moments and not others, biases are introduced into the recovered solution. In particular, the recovered (derotated and deskewed) contour will tend to have close to zero values for moments like m_{11} and m_{13}, which were used in the recovery measure. This is done at the expense of other moment values that tend to become large.

A method of minimizing the effects of bias on the recovered solution is to find an approximate solution using only certain moments, as described in the previous section. Once near a solution, we can employ a measure that uses different moments, not included in the original function. The original function $F(\cdot)$ (or alternatively $G(\cdot)$) generally gives us a solution within a few degrees of the desired solution, and is desirable in that it is simple to compute and can be solved for in closed form. A more general function is harder solve for but, once we are near the solution, using the approximation given to us by $F(\cdot)$, a gradient descent method can be used on the more general function in order to obtain a more accurate solution.

A particular bias problem resulted from function $G(\cdot)$, where 4th order moments are used. This was almost entirely eliminated by *polishing* the solution with a gradient descent method on a second function $K(\cdot)$, that was less bias-prone than $G(\cdot)$ (but harder to solve for). To obtain the function $K(\cdot)$, we note that the bias in $G(\cdot)$ has the tendency to get n_{13} as close to zero as possible at the expense of n_{31} becoming quite large. This is quite understandable considering the way in which $G(\cdot)$ is constructed. To solve this problem, the n_{13} and n_{31} moments (moments of the derotated image contour, i.e., a pure skew) were each derived as functions of moments in {P} (image moments) and the rotation parameter α (in the same way as described in the previous section). Then a function $J(\cdot)$ is derived where

$$J(\alpha;\{P\}) = n_{13}^2(\alpha;\{P\}) + n_{31}^2(\alpha;\{P\}) \qquad (3.2.1)$$

We want to minimize $J(\cdot)$, since minimzation of $J(\cdot)$ requires that the n_{31} moment also be small. Taking the derivative of $J(\alpha)$ with respect to α and setting it equal to zero, we obtain a function $K(\alpha)$ given by

$$K(\alpha;\{P\}) = \frac{\partial J(\alpha)}{\partial \alpha} = 2\,n_{13}(\alpha)\frac{\partial n_{13}(\alpha)}{\partial \alpha} + 2\,n_{31}(\alpha)\frac{\partial n_{31}(\alpha)}{\partial \alpha} = 0 \qquad (3.2.2)$$

Function $K(\cdot)$ has much better localization properties than the original function $G(\cdot)$, and can be effectively utilized, using gradient descent, to *polish* the solution given by $G(\cdot)$.

As an example, consider again the image shown in figure 14a. In figure 20a the derotated and deskewed contour is shown without polishing (using only function $G(\cdot)$). A polished version, applying gradient descent to function $K(\cdot)$ using the solution returned by $G(\cdot)$ as an initial starting value, is shown in figure 14b.

The polshing routine is fast and seems well-justified considering the degree of imnprovement in the localization of the recovered skew axes. The C code for minimal-bias function $K(\cdot)$ is given in Appendix B.

3.5 Computational Efficiency

The method-based algorithm described in section 3.2 primarily consists of computing moment values. Solving the resulting polynomial is not the timing bottleneck. As such, the algorithm is currently $O(n)$ for n contour points. It could easily be sped up to require only $O(log\ n)$ by computing the image moments on pyramid hardware. The moments can be computed on a real pyramid machine (see [7][14]) and passed on to a non-parallel machine that multiplies moments to obtain polynomial coefficients and then solves the resulting polynomial.

To give a feel to the reader for the computational efficiency of the algorithm (without pyramid hardware), we present actual timing results for the die image shown in figure 16. These timing results (for the die) can be compared to timing results given in [21] for the same die image (and presumably a similar set of contour edges). Real time performance is practically instantaneous. For each of the die edges superposed on the die in figure 16 response time was ceratinly within one second, where SYMAN was running on a SUN 4 SPARC station. Using the UNIX C shell *time* command, we can be more precise about timing. System times for the 4 edge images shown in the figure were given as .16 seconds, .33 seconds, .28 seconds, and .21 seconds, while user times (elpased time during the command) were given, respectively, as .28 seconds, .62 seconds, .50 seconds, and .44 seconds. In [21], the total computing time (on average) for detecting axes of skew for each of these die edges was given as "around 30 seconds". Strictly from a computational standpoint, then, the moment-based algorithm appears to be much faster than the algorithm given in [21]. In terms of speed, we would expect it to outperform by an order of magnitude (at least) any local method. This is due to the fact that *any* local method must either compare every contour point to every other contour point ($O(n^2)$) or use a projection-based method ($O(k\ n)$) [20], both of which are extremely time consuming. A more detailed analysis of the computational efficiency of several skewed-axis finding algorithms, including the moment-based analytic approach just presented, is given in [12].

4.0 A Local Tangent-based Method

In this section, we present a local tangent-based method for finding axes of skew. As argued in the introduction, there are strong reasons for a robust skew analyzer to incorporate both local and global axis-finding techniques. There are several local methods, discussed in section 2, for finding axes of skew. The motivation for yet another such method is because the other two local methods tried by the authors (see [12]) had various drawbacks that this method does not. The local method in Ponce [21] works well if curvature can be reliably computed from the image contour. Since computing curvature involves taking second derivatives, it is sensitive to noise and discretization error. In addition, the algorithm described in [21] will not work on linear contours ($k = 0$), which occur frequently in real images. The method described in [27] does not require computing any derivatives on the image contour, as opposed to our tangent method that requires computing a first derivative or Ponce's that involves second derivatives. The problem is that it is very sensitive to outliers, namely, mismatched pairs of points. In our method, as in ponce's, every pair of points at a given projection votes for a solution. As such, outliers do not pose a problem in recovering the correct solution. The line-based method in [27], i.e., selecting the projection angle yielding the *straightest* axis of skew symmetry, is very sensitive to outliers. In [12], where a version of the algorithm described in [27] is implemented, we attempt to deal with the problem by using median filtering, discarding points with absolute deviation larger than the median, and recomputing the line parameters. This robust statistics approach helps mitigate the effects of outliers, but does not completely solve the problem. Thus, the need for yet another local method. The tangent-based method described in this section is quite robust under noise and extremely insensitive to outliers. Examples of the algorithm are presented in section 4.1.

We first state some proprositions, then sketch out the method.

Proposition 4.1: For a symmetric planar curve C, intersecting tangents of symmetric points on C intersect on the axis of symmetry.

Proof: Without loss of generality, the symmetry axis is aligned with the y axis ($x = 0$). Since $y(x) = y(-x)$ by definition, we differentiate both sides with respect to x to obtain $y'(x) = -y'(-x)$. The slopes have opposite signs, one passing through the point (x, y), the other through the point $(-x, y)$, thus intersecting on the symmetry axis y .

Proposition 4.2: For a skew symmetric planar curve C, if the tangents of a skew symmetric pair of points intersect (are not parallel), then they intersect on the axis of skew symmetry.

Proof: Without loss of generality, assume the symmetry axis, before skewing, is aligned with the y axis ($x = 0$). Consider the symmetric pair of points $(-x, y)$ and (x, y). We assume that the curve is locally continuous and differentiable. The skew transformation, defined in equation (3.0.1), where the angle of skew is β, transforms the pair of points into a skew symmetric pair, respectively given by $(-x + cot\ \beta y, y)$ and $(x + cot\ \beta y, y)$. Their respective slopes are $\tan\theta_- = \frac{\partial y}{-\partial x + cot\ \beta\, \partial y}$ and $\tan\theta_+ = \frac{\partial y}{\partial x + cot\ \beta\, \partial y}$, where θ_- and θ_+ are the angles the corresponding tangents make with the x axis. Then we have

$$cot\ \theta_- = \frac{-\partial x + cot\ \beta\, \partial y}{\partial y}\ ,\quad cot\ \theta_+ = \frac{\partial x + cot\ \beta\, \partial y}{\partial y} \tag{4.1}$$

From which it immediately follows that

$$cot\ \beta = \frac{cot\ \theta_- + cot\ \theta_+}{2}\quad . \tag{4.2}$$

The tangent lines, having slopes of $\tan\theta_-$ and $\tan\theta_-$, and passing, respectively, through points $(-x + cot\ \beta y, y)$ and $(x + cot\ \beta y, y)$, intersect at a point (x_0, y_0) given by

$$x_0 = cot\ \beta(y - x\,\frac{\partial y}{\partial x})\ ,\ y_0 = y - x\,\frac{\partial y}{\partial x}. \tag{4.3}$$

Note that x_0 and y_0 are well-defined exactly where $\frac{\partial y}{\partial x}$ is well-defined. But since $\frac{\partial y}{\partial x}$ is the slope of the unskewed symmetry point (x, y), equation (4.3) simply states that the tangents at a skew pair of points intersect exactly where the tangents at the corresponding symmetry points are not parallel to the axis of symmetry. By definition, the axis of skew intersects the midpoint of each skew pair. So the axis of skew passes through the point $(y\ cot\ \beta, y)$. To prove the proposition, it remains to show that the line passing through the point $(y\ cot\ \beta, y)$ and having slope $\tan\beta$ (where $cot\ \beta$ is given in equation (4.2)), passes through the point (x_0, y_0) (equation (4.3)). This point is easily verified by the reader, and the proposition is proved.

Another way to prove proposition 4.2 is by showing that the skewed tangent of a point on a locally differentiable curve is identical to the tangent of the skewed curve.

From equation (4.2), however, we have a necessary condition on a pair of skew symmetric points. If a pair of points is in fact a skew symmetric pair, then the sum of the cotangents at the skew points is twice the cotangent of skew angle β. Of course, β is not known *apriori* so we cannot determine, given two contour points, whether they are a skew symmetric pair. We have a constraint, though, among pairs of skew symmetric pairs; namely, 2 pairs of skew symmetric points must return the same value when their cotangent values are added together. This is to be compared with the stronger necessary constraint among skew symmetric pairs of points found in Ponce [21]. Ponce has a necessary condition between a skew symmetric pair given by

$$\frac{k_+}{k_-} = -\left[\frac{\sin\theta_+}{\sin\theta_-} \right]^3 , \tag{4.4}$$

where $\theta\pm$ are as defined above, and $k\pm$ is the curvature at the point (before skewing) $(\pm x, y)$. This condition, however, requires knowledge of second derivative information at each point, i.e., curvature. On the other hand, the method described in [27] can be considered a necessary condition among triples of skew symmetric pairs, namely, the midpoints of these three pairs of points must lie on a straight line. This method requires no derivative computation at points on the contour. Thus, there is a tradeoff between a stronger necessary condition (among fewer pairs of skew points) and the need for more derivative information at each contour point.

4.1 The Local Tangent-based Algorithm

The general method used to implement the tangent-based alogrithm is now described. Pairs of contour points are compared. Every pair of contour points has an angle α_i that the line connecting the pair makes with the horizontal axis. If this pair of points corresponds to a skew pair, α_i is just the rotational component of the skew transformation. The pair of points is then added to a list of points associated with the bin α_i (we used a separate bin for each angular degree in the projection space). Within each bin, each pair of points determines a skew angle β (using equation (4.2)). For each bin, the skew parameter space is discretized into bins and the votes for β are histogramed. The projection angle α that yields the narrowest distribution (smallest σ) with respect to the skew angle β is selected as the rotation angle, with the peak of its β distribution selected as the corresponding angle of skew. This is equivalent to the method of projections described in [20] and also used in the skew finding algorithm described in [21].

The algorithm, in pseudo-code form, is given below. The input consists of a list of points, given as (x,y) values, together with the computed tangent value at each point.

1. For each point : compare to every other point in the list and add the pair to the projection bin corresponding to the slope between the two points.

2. At each projection bin : compute a beta value for every pair of points and histogram the beta values.

3. At each projection bin : compute the distribution with respect to the peak in the beta histogram (where the peak is taken as the mean (and the mode) of the distribution.

4. Order solutions as the projection angles (i.e., bins) and corresponding peak beta values having the smallest standard deviation σ.

5. Verify correctness of solution(s).

In practice, there is often skew ambiguity in the contour and more than one solution is desired. This is accompished by using a variant of the above-described method whereby, for each projection bin, a σ value is computed after the solution (peak in β) has been bracketed to exclude outliers. Each projection bin now has an associated σ value. These σ values are ordered, with the projection bin having the smallest value of σ being the most likely solution for rotation angle α. Thus, any number of solutions can be attained by changing the threshold for what is considered an acceptable value in σ. The notion of thresholds is not specific to our local tangent-based algorithm. Any local method must set a threshold in order to recover more than just the most likely solution. Many, if not most, of the skew contours that occur in real images have some degree of skew ambiguity so this is a problem that needs to be dealt with by any robust skew finding algorithm. We have been able to fine tune our algorithm so that σ did not need to be changed in order to find *all* the correct solutions for rotation angle α and skew angle β over a large set of test images.

In the next section, we show some experimental results on real and synthetic images. The performance of the algorithm is also compared to some existing methods. A much more thorough comparison of exisitng skew finding techniques can be found in [12].

4.2 Experimental Results

Figure 21a is the contour of a dodecagon under skew and rotation. Because its degree of skew symmetry is greater than four, *all* solutions for α and β cannot be recovered using the moment-based method of section 3. Using the tangent-based method described in the previous section, where the contour is generated synthetically and the tangent at a point is computed by taking finite differences in a local neighborhood around the point, all correct solutions of skew axes are found. The projection space spans $[-\frac{\pi}{2}, \frac{\pi}{2}]$, so a solution of α for the rotation angle also represents the rotational solution $\alpha + \pi$ (i.e., the recovered symmetric figure is correct up to a mirror reversal). Two of the twelve recovered solutions are shown in figure 21b and 21c.

An elongated superelliptic contour under skew and rotation is shown is figure 11a. Its recovered skew axes, using the moment-based algorithm, were shown in figures 11b and 11c. The tangent-based method recovers 4 solutions, 2 of which are unique (the other two are duals of the first two). The recovered skew axes are shown in figure 22a and 22b. There is a problem this local method has that almost results in a correct solution not being found. This is the result of false matches that take place, i.e., pairs of incorrectly matched contour points. There is a rotational solution at $\alpha = 50$ degrees. What should occur at that projection angle is that the points on the far ends of the elongated sides of the superellipse should be matched together. What happens instead is that points on the same side of the contour get matched together since the ends of the superellipse approach a straight line and at $\alpha = 50$ degrees these points all lie along the line going through the contour centroid and making an angle of 50 degrees with the horizontal axis. It is easier to see this by viewing a scatter plot of the matched contour points, as illustrated in [12]. In any event, false matches are a problem with any local method such as [21] and [27], and are more prevalent for complex, non-convex contours. The problem of point mismatching for local skew-finding algorithms is discussed in greater detail in [12]. This is one of the advantages that a global, moment-based method has over the local methods; no pointwise correspondence is required so there is no problem of pointwise mismatches. The algorithm presented in section 3 has no problem finding the correct solutions for the skew symmetry axes, as was shown in figures 11b and 11c.

5. Detecting Skew Symmetries

A prerequisite to finding axes of skew *given* a skew symmetric curve is, of course, detecting the skewed contour in the first place. There have been very few papers dealing with this topic. Clearly, any system that wants to analyze skew symmetric contours must first be able to find them. Nevatia [19] pairs symmetric contours by considering the nearest symmetries. Brooks [4] used prediction models and constrained viewpoint assumptions to help in ribbon detection. Rao [25] generalized the work in [19] to images with marking and broken contours. A survey of such detection methods and some of their deficiencies is given in Mohan [16]. In [16], a new symmetry axis is defined. His symmetry axis is curve-based, rather than being point (edge) based, and is thus much easier to compute. The defintion for Mohan's symmetry axis is very general, i.e., any two curves can be paired together, so that there is no strongly attached necessary condition (as in [21] or the tangent criterion of section 4). It would seem that a robust symmetry detector should also be able to *exclude* certain pairs of curves from being grouped as a (skew) symmetric contour pair.

Among the deficiencies mentioned in [16] with existing methods are :

1. Only pure symmetries, and not skew symmetries, are detected.

2. In the projected scene, the orientation of the image axis may be a function a viewpoint and may not correspond to the projected axes of the object.

3. Detection of these axes are based on matching points on the contours. This has a computational complexity of $O(n^2)$ so that it is highly inefficient. Even with the method of projections, the time bound is $O(k\ n)$, where k is the number of bins the projection space has been divided into.

In this section, a method is described to efficiently detect skew symmetric contours. We will specifically address how the method deals with the deficiencies listed in [16] and cited above. As noted in section 2, many symmetric shapes in the real world are not simply symmetric, but rather they are doubly symmetric; that is, the x and y axes both serve as axes of symmetry (see section 2.2). But, as noted in section 2, a doubly symmetric curve is also π-rotationally symmetric: for every point (x, y) on curve C, there is a point $(-x, -y)$ on curve C. It was also shown in section 2.2 (Corollary) that a rotationally symmetric curve projects orthographically to a rotationally symmetric curve. Furthermore, it was shown that a heuristic of *rotational symmetry in the plane corresponds to double symmetry in the scene* (wherever such interpretation is possible) corresponds to Kanade's skew symmetry heuristic. In section 3.1 the point was made that, for a rotationally symmetric curve, all 3rd order moments are equal to zero. But this can be used to detect skew symmetric contours. If the 3rd order moments are all identically zero (or sufficiently small), our algorithm hypothesizes that the contour is actually the projection of a doubly reflexive symmetric 3D curve.

Consider the superelliptic curve shown in figure 11a. In computing its moments, SYMAN determines that compared to other order moments, its 3rd order moments are very close to zero. It then presumes that the contour is rotationally symmetric and the projection of a double symmetry.

The intensity image of a Ferrari is shown in figure 23a; the detected edges, using a Canny-like edge detector, are shown in figure 23b. Figure 23c shows a contour that, when given as input to SYMAN, was perceived to be a rotationally symmetric contour since its third order moments were relatively small (compared to its second order moments). Our algorithm automatically checks a given contour to see if its third order moments are unusually small. If that turns out to be the case, an underlying double symmetry is conjectured. This conjecture can be verified by solving for the rotation and skew components (the moments have presumably already been computed) and then verifying that all moments of the recovered contour of the form $x^{odd}\, y^z$ or $x^z\, y^{odd}$ are approximately equal to zero (which must be true if the real curve was doubly reflexive symmetric). The skew axes recovered for the edge image of figure 23c, using the moment-based algorithm of section 3, are also shown.

What about singly symmetric figures that project onto singly skew symmetric contours? For this case, the method of detecting the skew symmetry actually involves solving for the axis of skew. Since the moment-based method of finding skew symmetric axes is used, even this detection algorithm is fast. Once a proposed pair of skewed axes has been found, the solution is tested to determine if, for the recovered (and supposedly symmetric) contour, all moments of the form $x^{odd}\, y^z$ zero out. Thus, a fast moment-based technique exists to determine whether the contour is the projection of a double symmetry. To determine whether the contour is the projection of a single symmetry, a set of skew-pair solutions must be generated and tested for, but even this can be done efficiently using the moment-based axis-finding method of section 3.2.

The 1st and 3rd deficiencies listed above for current methods of symmetry detection are not applicable to this method. It is capable of detecting skew symmetry, not just symmetry, and it is fast as there is no need for either pointwise matching or a traversal over projection space (see section 3.5). But what about item number 2? In general, a skew symmetric axis is a candidate solution for the projection of the underlying axis of symmetry. It is the job of a symmetry analyzer to find such axes of skew. Whether a particular axis of skew corresponds to the actual symmetry axis is usually a context-sensitive decision that cannot be decided at the contour level [10]. Adjacent contours, each with a set of plausible skew symmetric axes, can be used to find a unique solution.

Image ellipses provide axes of symmetry that do *not* correspond, under an assumption of general viewpoint, to the axes of the underlying scene ellipse. Therefore, the *recovered* axes of an image ellipse need to be handled with caution. While they do not necessarily relate to the underlying symmetry axes, they may provide important information. If the image ellipse is the projection of a circle, then the minor axis is aligned with the projected normal of the circle. In any event, we test for a symmetry under pure rotation (step 2 of the algorithm). Thus, since ellipses orthographically project to other rotated ellipses,

ellipses are detected at this point. The contours detected at this stage of the algorithm are presumed to be ellipses since, under general viewpoint, symmetric curves do *not* project onto symmetric contours (with ellipses being an exception). To a large extent, this approach also handles deficiency 2 above, since tricky contours, whose projected axes are deceiving (such as ellipses), are detected early and handled separately.

6. SYMAN

In this section, we briefly describe the symmetry analyzer SYMAN. The motivation behind SYMAN was the fact the the universe is, by and large, a very regular, symmetric place. As such, a robust recognition system has the need for a system (like humans) that understands a lot about symmetries and their projections.

SYMAN consists of of different components. In particular, it is capable of running 4 different algorithms that recover axes of skew. Two of these algorithms were described in this paper. The other two are described in [21] and [27]. The reason for wanting to implement all of these methods is because each of the methods appears to have certain strengths and the trick seems to be in knowing when to use each technique. For example, the moment-based method does not work well under occlusion or when the degree of skew symmetry is greater than 4. The method of [21] does not work well when curvature values are unreliable or the contour is discontinuous. The line-based method of [27] is very sensitive to outliers. By having several methods to choose from, and knowing when to use them, SYMAN can be made more robust.

Aside from being able to recover axes of skew, SYMAN also tries to *understand* the contour. It tests for whether the contour is a real symmetry under rotation (no skew component). It checks if the contour is a rotational symmetry. When a solution is recovered, it tests whether it is valid by seeing if certain moments in the recovered contour are (approximately) equal to zero.

One of the more important features of SYMAN is the realization that there are some things a symmertry analyzer should not resolve. Attached to each contour is a structure. If SYMAN determines that the contour is rotationally symmetric, it will find 4 pairs of skew axes. Rather than attempting to decide which of these is correct (as several other algorithms would do), SYMAN enters these solutions, as well as the conjecture that the contour is doubly symmetric, to the contour's data structure. Since, in this case, there really are 4 such pairs of skew axes, it makes sense not to try and resolve the *skew ambiguity* on this level, but rather to pass it on to a higher level that is more *context-sensitive* and can probably arrive at a unique set of skew axes using constraints from adjacent contours (see [10]).

Realizing that moments are fast but subject to bias, SYMAN has a bias correcting module. Given a solution, which it assumes is *close* to being correct, it polishes the solution to greater accuracy using a measure that is more insensitive to noise and discretization effects. This bias-correcting module substantially improves the accuracy of recovered solutions (see section 3.5). SYMAN is intended as a module in a larger shape recovery system that tries to recover shape from monocular intensity images by assuming an underlying model of a regular, symmetric universe (see [10]).

7. Conclusions and Future Work

This paper presented two new techniques for finding axes of skew. The first was a analytic approach that *solved* for the axes of skew from computed image moments. This global-based method is very fast. It does not require any processing of the edge points (e.g., derivative information). It does not require that contours be smooth or even continuous, but is restricted to skew ambiguity of degree 4 or less. The general algorithm consisted of decomposing the problem into two parts : finding the rotational component of a pure skew and finding the skew component of a pure symmetry. Examples of the method for real and synthetic images were given.

The second algorithm is a local tangent-based method. First, we derived a necessary constraint among pairs of skew contour pairs. In particular, it was shown that the sum of the cotangent values among skew pairs is constant and is twice the cotangent of the skew angle. This can be used as the basis for finding axes of skew. For each projection angle, points lying along that angle (with respect to each other) are matched together. A correct projetion angle is one that best satisfies the constraint of having a consistent skew value. Examples of the method on both real and synthetic images were shown.

Aside from finding axes of skew given a symmetric contour, methods for finding skew symmetric contours in an image are needed. It was shown that for a large class of symmetric curves, those having double reflexive symmetry, there is a moment-based signature that can be used for the purpose of detection. In particular, doubly reflexive symmetric curves were shown to project to π-rotational symmetric contours, which have the property that 3rd order moments are equal to zero.

Many of the results in this paper were incorporated into SYMAN, a symmetry analyzer. Elements of the SYMAN system were discussed in section 6. SYMAN is intended as part of a larger system that can infer shape from monocular intensity images under the assumption of a highly symmetric universe.

Future work involves research into moment-based techniques that do not suffer from the problems of occlusion or an upper bound on the degree of skew symmetry. For example, the projection method could be combined with the moment-based technique to avoid problems of occlusion. This work on symmetry analysis is mostly concerned with recovering local 2D coordinate systems, i.e., local axes of skew. It needs to be incorporated into a system that is designed to, given a set of local image coordinate systems, come up with a globally consistent 3D coordinate system.

Acknowledgements

Support for this work was provided in part by the Advanced Research Projects Agency of the Department of Defense under contract #N00039-84-C-0165 and by the National Science Foundation Grant #IRI8800370. Thanks to Cliff Beshers for the graphics package used in generating some of the images for this paper.

Appendix A

A.1 Code for functions F

```
void        cubic(moms, coef) double       moms[MOM_DEG][MOM_DEG]; float        coef[]; {
    double  m10, m01, m20, m02, m11, m30, m03, m21, m12;
    double  m40, m04, m31, m13, m22;
    double  m20_2, m02_2, m11_2, m30_2, m03_2, m21_2, m12_2;
    double  m40_2, m04_2, m31_2, m13_2, m22_2;
    float   a, b, c, d, e;

    /* initialize variables */

    /* moms[] contains the moments of the image contour */

    m11 = moms[1][1];
    m20 = moms[2][0];
    m02 = moms[0][2];
    m21 = moms[2][1];
    m12 = moms[1][2];
    m30 = moms[3][0];
    m03 = moms[0][3];
    m40 = moms[4][0];
    m31 = moms[3][1];
    m22 = moms[2][2];
    m13 = moms[1][3];
    m04 = moms[0][4];

    m11_2 = m11*m11;
    m20_2 = m20*m20;
    m02_2 = m02*m02;
    m30_2 = m30*m30;
    m21_2 = m21*m21;
    m12_2 = m12*m12;
    m03_2 = m03*m03;
    m40_2 = m40*m40;
    m04_2 = m04*m04;
    m31_2 = m31*m31;
    m13_2 = m13*m13;
    m22_2 = m22*m22;
```

```
    /* equations for cubic equation : F = axˆ3 + bxˆ2 + cx + d = 0  */

    a = (m11_2+m20_2)*m03_2+((-2*m11*m12+2*m30*m11-4*m20*m21)*m02-2*m20*m11*m12+2*m21*
m11_2+2*m20*m30*m11-2*m20_2*m21)*m03+(m12_2-2*m30*m12+m30_2+4*m21_2)*m02_2+ (4*m20*m12_2+(-
6*m21*m11-4*m20*m30)*m12-2*m21*m30*m11+4*m20*m21_2)*m02+ (m11_2+4*m20_2)*m12_2+(2*m30*m11_2-
6*m20*m21*m11)*m12+(m30_2+m21_2)*m11_2-2* m20*m21*m30*m11+m20_2*m21_2;

    b = -m20_2*m03_2+((4*m20*m21-2*m30*m11)*m02+2*m20*m11*m12-2*m21*m11_2-4*m20*
m30*m11+4*m20_2*m21)*m03+(2*m30*m12-2*m30_2-4*m21_2)*m02_2+(-4*m20*m12_2+(6*m21*m11+8*
m20*m30)*m12+4*m21*m30*m11-8*m20*m21_2)*m02+(-m11_2-8*m20_2)*m12_2+(12*m20*m21*m11-4*
m30*m11_2)*m12+(-3*m30_2-2*m21_2)*m11_2+6*m20*m21*m30*m11-3*m20_2*m21_2;

    c = (2*m20*m30*m11-2*m20_2*m21)*m03+m30_2*m02_2+(-4*m20*m30*m12-2*m21*m30*m11+4*
m20*m21_2)*m02+4*m20_2*m12_2+(2*m30*m11_2-6*m20*m21*m11)*m12+(3*m30_2+m21_2)*m11_2-6
*m20*m21*m30*m11+3*m20_2*m21_2;

    d = -m30_2*m11_2+2*m20*m21*m30*m11-m20_2*m21_2;

    coef[3] = a;
    coef[2] = b;
    coef[1] = c;
    coef[0] = d;

/* solving this equation yields 3 solutions for cosˆ2(alpha), where alpha is rotation
   component of skew transformation */

}
```

A.2 Code for Function G

```
void quartic(moms, coef) double moms[MOM_DEG][MOM_DEG]; float coef[]; {
    double  m10, m01, m20, m02, m11, m30, m03, m21, m12;
    double  m40, m04, m31, m13, m22;
    double  m20_2, m02_2, m11_2, m30_2, m03_2, m21_2, m12_2;
    double  m40_2, m04_2, m31_2, m13_2, m22_2;
    float   a, b, c, d, e;

    /* initialize variables */

    m11 = moms[1][1];
    m20 = moms[2][0];
    m02 = moms[0][2];
    m21 = moms[2][1];
    m12 = moms[1][2];
    m30 = moms[3][0];
    m03 = moms[0][3];
    m40 = moms[4][0];
    m31 = moms[3][1];
    m22 = moms[2][2];
    m13 = moms[1][3];
    m04 = moms[0][4];

    m11_2 = m11*m11;
    m20_2 = m20*m20;
    m02_2 = m02*m02;
    m30_2 = m30*m30;
    m21_2 = m21*m21;
    m12_2 = m12*m12;
    m03_2 = m03*m03;
    m40_2 = m40*m40;
    m04_2 = m04*m04;
    m31_2 = m31*m31;
    m13_2 = m13*m13;
    m22_2 = m22*m22;

    /* equations for quartic equation : axˆ4 + bxˆ3 + cxˆ2 + dx + e == 0  */

    a =   (m11*m11+m20*m20)*m04*m04+((-2*m11*m13+6*m31*m11+2*m20*m40-6*m20*m22)*m02-2*m20*m11
        *m13-2*m40*
        m11*m11+6*m20*m31*m11-6*m20_2*m22)*m04+(m13_2-6*m31*m13+m40_2-6*m22*m40+9*m31_2+9*m22_2)*
        m02_2+(6*m20*m13_2+((6*m40-12*m22)*m11-20*m20*m31)*m13+(-2*m31*m40-12*m22*m31)*m11-6*m20
        *m22*m40+6*m20*m31_2+18*m20*m22_2)*m02+(4*m11_2+9*m20_2)*m13_2+(8*m31*m11_2+(6*m20*m40-12
```

```
        *m20*m22)*m11-6*m20_2*m31)*m13+(m40_2+4*m31_2)*m11_2+(-2*m20*m31*m40-12*m20*m22*m31)*m11
        +m20_2*m31_2+9*m20_2*m22_2;

    b =   -m20_2*m04_2+((-6*m31*m11-4*m20*m40+6*m20*m22)*m02+2*m20*m11*m13+4*m40*m11_2-12*
        m20*m31*m11+12*m20_2*m22)*m04+(6*m31*m13-3*m40_2+12*m22*m40-18*m31_2-9*m22_2)*
        m02_2+(-6*m20*m13_2+((12*m22-12*m40)*m11+40*m20*m31)*m13+(6*m31*m40+24*m22*m31)
        *m11+18*m20*m22*m40-18*m20*m31_2-36*m20*m22_2)*m02+(-4*m11_2-18*m20_2)*m13_2+
        (-16*m31*m11_2+(24*m20*m22-18*m20*m40)*m11+18*m20_2*m31)*m13+(-4*m40_2-12*m31_2)
        *m11_2+(8*m20*m31*m40+36*m20*m22*m31)*m11-4*m20_2*m31_2-27*m20_2*m22_2;

    c  =  (2*m20*m40*m02-2*m40*m11_2+6*m20*m31*m11-6*m20_2*m22)*m04+(3*m40_2-6*m22*m40+9*m31_2)
        *m02_2+((6*m40*m11-20*m20*m31)*m13+(-6*m31*m40-12*m22*m31)*m11-18*m20*m22*m40+18*m20*
        m31_2+18*m20*m22_2)*m02+9*m20_2*m13_2+(8*m31*m11_2+(18*m20*m40-12*m20*m22)*m11-18*
        m20_2*m31)*m13+(6*m40_2+12*m31_2)*m11_2+(-12*m20*m31*m40-36*m20*m22*m31)*m11+6*m20_2*
        m31_2+27*m20_2*m22_2;

    d =   -m40_2*m02_2+(2*m31*m40*m11+6*m20*m22*m40-6*m20*m31_2)*m02+(6*m20_2*m31-6*m20*
        m40*m11)*m13+(-4*m40_2-4*m31_2)*m11_2+(8*m20*m31*m40+12*m20*m22*m31)*m11-4*m20_2
        *m31_2-9*m20_2*m22_2;

    e =   m40_2*m11_2-2*m20*m31*m40*m11+m20_2*m31_2;


    coef[4] = a;
    coef[3] = b;
    coef[2] = c;
    coef[1] = d;
    coef[0] = e;

  }
```

Appendix B

Code for bias elimination function K

```
void        pol(degree, theta, m) int        degree; float        *theta; double        m[MOM_DEG][MOM_DEG]; {

    double    func_err_4(), func_err_3();

    bit = 0;
    ang = *(theta);

    m11 = m[1][1];
    m20 = m[2][0];
    m02 = m[0][2];
    m30 = m[3][0];
    m21 = m[2][1];
    m12 = m[1][2];
    m03 = m[0][3];
    m40 = m[4][0];
    m31 = m[3][1];
    m22 = m[2][2];
    m13 = m[1][3];
    m04 = m[0][4];

    if (degree == 4) {

    f1 = func_err_4(ang - ANG_EPSILON);
    f2 = func_err_4(ang);
    f3 = func_err_4(ang + ANG_EPSILON);

           /* if error function changes sign within epsilon  of current theta value then return */
    if (f1*f2<0 || f2*f3<0)          return;

    else   /* look for a zero crossing */
    {            direction = 0;

        if (fabs(f1)-fabs(f3)>0)    { direction = 1;
                                    cur_f = f3;
                                    cur_theta = ang+ANG_EPSILON; }
        else                        { direction = -1;
```

```
                cur_f = f1;
                cur_theta = ang-ANG_EPSILON; }

        new_f = cur_f;
        new_theta = cur_theta;

        while (new_f*cur_f > 0) { /*        printf("cur_f %lf new_f %lf cur_theta %f new_theta %f0,cur_f,new_f, cur_theta,
new_theta); */
            cur_f = new_f;
            cur_theta = new_theta;
            new_theta = cur_theta + T_INCR*direction;
            new_f = func_err_4(new_theta); }

        *theta = new_theta;
        return;     }     } }


double  func_err_4(f_alpha) float   f_alpha; {
    ca = cos(f_alpha);
    sa = sin(f_alpha);
    ca_2 = ca*ca;
    sa_2 = sa*sa;
    ca_3 = ca_2*ca;
    sa_3 = sa_2*sa;
    ca_4 = ca_3*ca;
    sa_4 = sa_3*sa;
    sa_5 = sa_4*sa;
    sa_6 = sa_5*sa;
    ca_5 = ca_4*ca;
    ca_6 = ca_5*ca;
    ca_7 = ca_6*ca;
    ca_8 = ca_7*ca;
    sa_7 = sa_6*sa;
    sa_8 = sa_7*sa;

    m11_2 = m11*m11;
    m11_3 = m11*m11_2;
    m11_4 = m11*m11_3;

    m20_2 = m20*m20;
    m20_3 = m20*m20_2;
    m20_4 = m20*m20_3;

    m02_2 = m02*m02;
    m02_3 = m02*m02_2;
    m02_4 = m02*m02_3;


    /* moment (1,3) of the recovered contour as a function of image moments and rotation    angle  f_alpha */

    n13    =    -((ca_4*m11-ca_3*sa*m20)*m04+(-ca_4*m13+(ca-ca_3)*sa*m40+(3*ca_4-3*ca_2)*m31
+3*ca_3*sa*m22)*m02+((3*ca_2-3*ca_4)*m20-2*ca_3*sa*m11)*m13+((-ca_4+2*ca_2-1)*m40+(2*ca-2*ca_3)
*sa*m31)*m11+(ca_4-2*ca_2+1)*m20*m31+(3*ca_3-3*ca)*sa*m20*m22)/(ca_2*m02-2*ca*sa*m11+sa_2*m20) ;


    /* moment (3,1) of the recovered contour as a function of image moments and rotation    angle  f_alpha */

    n31  =  -((ca_4*m11_3-3*ca_3*sa*m20*m11_2+(3*ca_2-3*ca_4)*m20_2*m11+(ca_3-ca)*sa*m20_3)*m04   +(ca_3*sa*m40-
ca_4*m31)*m02_3+(((3*ca_4-3*ca_2)*m40+3*ca_4*m22)*m11+(3*ca_2-3*ca_4)*m20*m31-   3*ca_3*sa*m20*m22)*m02_2+((-
3*ca_4*m11_2+6*ca_3*sa*m20*m11+(3*ca_4-3*ca_2)*m20_2)*m13+        ((3*ca-3*ca_3)*sa*m40+(3*ca_2-3*ca_4)*m31-
3*ca_3*sa*m22)*m11_2+(6*ca_3-6*ca)*sa*m20*m31*m11+   (3*ca-3*ca_3)*sa*m20_2*m22)*m02+(2*ca_3*sa*m11_3+(3*ca_4-
3*ca_2)*m20*m11_2+(ca_4-2*ca_2+1)*m20_3)     *m13+((-ca_4+2*ca_2-1)*m40+(2*ca_3-2*ca)*sa*m31)*m11_3+((3*ca_4-
6*ca_2+3)*m20*m31+(3*ca-3*ca_3)*                     sa*m20*m22)*m11_2+(-3*ca_4+6*ca_2-
3)*m20_2*m22*m11)/(ca_6*m02_3+(3*ca_4*sa_2*m20-6*ca_5*sa*m11)*            m02_2+(12*ca_4*sa_2*m11_2-
12*ca_3*sa_3*m20*m11+3*ca_2*sa_4*m20_2)*m02-8*ca_3*sa_3*m11_3+12*ca_2*          sa_4*m20*m11_2-
6*ca*sa_5*m20_2*m11+sa_6*m20_3);


     /* computes d(m13(f_alpha;{P}))/d(f_alpha) */

     d13    =    (((2*ca_5*sa*m11+(2*ca_6-ca_4)*m20)*m02+(4*ca_6-6*ca_4)*m11_2+(8*ca_3-6*ca_5)    *sa*m20*m11+(-
2*ca_6+5*ca_4-3*ca_2)*m20_2)*m04+(-2*ca_5*sa*m13+(2*ca_6-ca_4-ca_2)*m40+6*          ca_5*sa*m31+(3*ca_4-
```

```
6*ca_6)*m22)*m02_2+((4*ca_4*m11+(-4*ca_5-4*ca_3)*sa*m20)*m13+ ((-6*ca_5+4*ca_3+2*ca)*sa*m40+(16*ca_6-
20*ca_4+4*ca_2)*m31+(12*ca_5-12*ca_3)*sa*m22)*m11 +(-2*ca_6+5*ca_4-4*ca_2+1)*m20*m40+(-4*ca_5+12*ca_3-
8*ca)*sa*m20*m31+(12*ca_2-12*ca_4)*m20*m22) *m02+((8*ca_3-8*ca_5)*sa*m11_2+(-16*ca_6+28*ca_4-
12*ca_2)*m20*m11+(6*ca_5-12*ca_3+6*ca)*sa*m20_2) *m13+((-4*ca_6+6*ca_4-2)*m40+(8*ca_3-
8*ca_5)*sa*m31)*m11_2+((2*ca_5-4*ca_3+2*ca)*sa*m20*m40+ (4*ca_4-8*ca_2+4)*m20*m31+(12*ca_5-
12*ca_3)*sa*m20*m22)*m11+(-2*ca_5+4*ca_3-2*ca)*sa*m20_2*m31+ (6*ca_6-15*ca_4+12*ca_2-
3)*m20_2*m22)/(ca_4*m02_2+(2*ca_2*sa_2*m20-4*ca_3*sa*m11)*m02+4*ca_2 *sa_2*m11_2-
4*ca*sa_3*m20*m11+sa_4*m20_2) ;


   /* computes d(m31(f_alpha;{P}))/d(f_alpha) */

   d31 = -(((2*ca_5*sa*m11_3+(6*ca_6-9*ca_4)*m20*m11_2+(12*ca_3-6*ca_5)*sa*m20_2*m11+(-2* ca_6+7*ca_4-
5*ca_2)*m20_3)*m02+(4*ca_6+2*ca_4)*m11_4+(-14*ca_5-4*ca_3)*sa*m20*m11_3+ (-
18*ca_6+15*ca_4+3*ca_2)*m20_2*m11_2+(10*ca_5-8*ca_3-2*ca)*sa*m20_3*m11+(2*ca_6-3*ca_4+1)* m20_4)*m04+((3*ca_4-
2*ca_6)*m40-2*ca_5*sa*m31)*m02_4+(((10*ca_5-12*ca_3)*sa*m40+(-4*ca_6-2*ca_4)
*m31+6*ca_5*sa*m22)*m11+(2*ca_6+ca_4-3*ca_2)*m20*m40+(16*ca_3-4*ca_5)*sa*m20*m31+(6*ca_6-9*ca_4)
*m20*m22)*m02_3+((-6*ca_5*sa*m11_2+(18*ca_4-12*ca_6)*m20*m11+(6*ca_5-12*ca_3)*sa*m20_2)*m13+ ((18*ca_6-
39*ca_4+21*ca_2)*m40+(12*ca_3-6*ca_5)*sa*m31+(18*ca_6-3*ca_4)*m22)*m11_2+((6*ca-6*ca_5) *sa*m20*m40+(-
24*ca_6+60*ca_4-36*ca_2)*m20*m31+(-18*ca_5-12*ca_3)*sa*m20*m22)*m11+(6*ca_5-6*ca) *sa*m20_2*m31+(24*ca_2-
24*ca_4)*m20_2*m22)*m02_2+((-16*ca_6*m11_3+36*ca_5*sa*m20*m11_2+(24*ca_6-12* ca_4-12*ca_2)*m20_2*m11+(-
4*ca_5-8*ca_3+12*ca)*sa*m20_3)*m13+((-14*ca_5+32*ca_3-18*ca)*sa*m40+(-16* ca_6+32*ca_4-16*ca_2)*m31-
12*ca_5*sa*m22)*m11_3+((-6*ca_6+9*ca_4-3)*m20*m40+(36*ca_5-72*ca_3+36*ca) *sa*m20*m31+(24*ca_2-
24*ca_4)*m20*m22)*m11_2+((12*ca_6-18*ca_4+6)*m20_2*m31+(-18*ca_5+48*ca_3-30*ca)* sa*m20_2*m22)*m11+(-
6*ca_6+9*ca_4-3)*m20_3*m22)*m02+(8*ca_5*sa*m11_4+(16*ca_6-16*ca_4)*m20*m11_3+ (6*ca-
6*ca_5)*sa*m20_2*m11_2+(4*ca_6-14*ca_4+16*ca_2-6)*m20_3*m11+(-2*ca_5+4*ca_3-2*ca)*sa*m20_4)* m13+((-
4*ca_6+14*ca_4-16*ca_2+6)*m40+(8*ca_5-16*ca_3+8*ca)*sa*m31)*m11_4+((2*ca_5-4*ca_3+2*ca)*sa*m20 *m40+(16*ca_6-
48*ca_4+48*ca_2-16)*m20*m31+(-12*ca_5+24*ca_3-12*ca)*sa*m20*m22)*m11_3+((-6*ca_5+12* ca_3-
6*ca)*sa*m20_2*m31+(-18*ca_6+51*ca_4-48*ca_2+15)*m20_2*m22)*m11_2+(6*ca_5-12*ca_3+6*ca)*sa*m20_3*
m22*m11)/(ca_8*m02_4+(4*ca_6*sa_2*m20-8*ca_7*sa*m11)*m02_3+(24*ca_6*sa_2*m11_2-24*ca_5*sa_3*m20*m11+
6*ca_4*sa_4*m20_2)*m02_2+(-32*ca_5*sa_3*m11_3+48*ca_4*sa_4*m20*m11_2-24*ca_3*sa_5*m20_2*m11+4*ca_2*
sa_6*m20_3)*m02+16*ca_4*sa_4*m11_4-32*ca_3*sa_5*m20*m11_3+24*ca_2*sa_6*m20_2*m11_2-8*ca*sa_7*m20_3
*m11+sa_8*m20_4);

  cur_err = n13*d13 + n31*d31;

  return(cur_err);
 }
```

References

[1] Barnard, S., Choosing a basis for perceptual space, *IEEE Workshop on Computer Vision: Representation and Control*, 1984.

[2] Brady, J.M., and Asada, H., Smoothed local symmetries and their implementation, *International Journal of Robotics Research 3*, No. 3, 1984.

[3] Brady, J.M., and Yuille, A., An extremum principle for shape from contour, MIT, AI Lab., MIT-AIM 711, 1983.

[4] Brooks, R. A., Symbolic reasoning among 3-D models and 2-D images, *Artificial Intelligence* **17**, 285-348, 1981.

[5] Buchberger, B., Applications of Grobner bases in non-linear computational geometry. In R. Janssen, editor, *Trends in computer algebra: international symposium*, pages 52-80. Springer-Verlag, 1987.

[6] Buchberger, B., Collins, G.E., and Loos, R., editors, *Computer Algebra: Symbolic and Algebraic Computation*, Springer-Verlag, 1983, New York.

[7] Burt, P.J., Moment images, polynomial fit filters, and the problem of surface interpolation, in the *Proceedings of the 1988 Conference on Computer Vision and Pattern Recognition*, Ann Arbor, Michigan, June 5-9, pages 144-152.

[8] Cyganski, D., Orr, J.A., and Pinjo, Z., A tensor operator method for identifying the affine transformation relating image pairs, in *Proc. IEEE-CS Conference on Computer Vision and Pattern Recognition*, Montreal, Quebec, 1983, pages 361-363.

[9] Freidberg, S.A., Finding Axes of Skewed Symmetry, in *Computer Vision, Graphics, and Image Processing*, **34**, 138-155, 1986.

[10] Gross, A.D., Shape from a Symmetric Universe, Technical Report CUCS-065-90, Department of Computer Science, Columbia University.

[11] Gross, A.D., and Boult, T.E., Error of fit measures for recovering parametric solids, in *Proceedings of the 2nd International Conference on Computer Vision*, pages 690-694, Tampa, Fl., December 1988.

[12] Gross, A.D., Boult, T.E., Skew symmetry detection: Evaluating the state of the art, Technical Report, Department of Computer Science, Columbia University.

[13] Horaud, R., and Brady, J.M., On the geometric interpretation of image contours, in *Proc. First Int'l Conference on Computer Vision*, London, U.K., June 1987.

[14] Ibrahim, H.A.H., Image understanding algorithms on fine-grained tree-structured SIMD machines, PhD thesis, Department of Computer Science, Columbia University, 1984.

[15] Kanade, T., Recovery of the 3D shape of an object from a single view, *Artificial Intelligence*, vol. 17, pages. 409-460, 1981.

[16] Mohan, R., Perceptual organization for computer vision, PhD thesis, School of Engineering, University of Southern California, August, 1989.

[17] Nalwa, Vic, Line-drawing interpretation: Bilateral symmetry, in *Proceedings of the Image Understanding Workshop*, vol. 2, Feb., 1987, pp. 956-967.

[18] Nalwa, Vic, Line-drawing interpretation: Straight-lines and conic-sections, in *Proceedings of the Image Understanding Workshop*, vol. 2, Feb., 1987, pp. 942-955.

[19] Nevatia, R., Computer analysis of scenes of 3-d curved objects, Birkhauser-Verlag, Basel, Switzerland, 1976.

[20] Nevatia, R., and Binford, T.O., Description and recognition of complex curved objects, *Artificial Intelligence* **8**, 77-98, 1977.

[21] Ponce, J., On characterizing ribbons and finding skewed symmetries, accepted for publication in *Computer Vision, Graphics, and Image Processing*, Fall 1990.

[22] Ponce, J., Chelberg, D., and Mann, W., Invariant properties of straight homogeneous Generalized Cylinders and their Contours, in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. II, No. 9, September 1989.

[23] Ponce, J., and Kriegman, D.J., On recognizing and positioning curved 3D objects from image contours, in *IEEE Proceedings of the Workshop on 3D Interpretation of Scenes*, pages 116-123, Nov. 27-29, 1989, Austin, TX.

[24] Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T., *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, New York, 1989.

[25] Rao, K., Shape description from sparse and imperfect data, Technical Report IRIS-250, Institute for Robotics and Intelligent Systems, USC, December 1988. Ph.D. Thesis.

[26] Rosenfeld, A., Axial representation of shape, *Journal of Computer Vision, Graphics, and Image Processing*, (33):156-173, 1986.

[27] Saint-Marc, P., and Medinoni, G., B-spline contour representation and symmetry detection, technical report, Institute for Robotics and Intelligent Systems, University of Southern California, Los Angeles, CA, Feb 20, 1990.

[28] Stevens, K.A., The visual interpretation of surface contours, *Artificial Intelligence***17**, 47-73, 1981.

[29] Tsukune, H., and Goto, K., Extracting elliptical figures from an edge vector field, in *Proc IEEE Conf. on Computer Vision and Pattern Recognition*, WA, USA (1983), pp. 138-141. *IEEE Tran*

[30] Ulupinar, F., and Nevatia, R., Using symmetries for analysis of shape from contour, in *Proceedings of the Second International Computer Vision Conference*, 414-426, Dec. 5-8, 1988.

[31] Yuen, H.K., Illingwirth, J., Kittler, J., Detecting partially occluded ellipses using the Hough transform, in *Image and Vision Computing*, vol. 7, no. 1, Feb, 1989.

[32] Yuen, S.Y.K., Shape from contour using symmetries, techincal report, School of Cognitive Sciences, University of Sussex, Falmer, Brighton, BN1 9QN, 1989.