Comparison of Different Approaches for the Calculation of Projective Symmetry or the Axis of a SHGC

Sven Utcke
Arbeitsbereich Kognitive Systeme, Fachbereich Informatik,
Universität Hamburg, Germany
utcke@informatik.uni-hamburg.de

Zusammenfassung. Ein wesentlicher Schritt zur Erkennung von Objekten mit symmetrischem Umriss — wie zum Beispiel planare, symmetrische Objekte, aber auch Rotations-Körper — ist die Berechnung der Transformation, welche die beiden Seiten des Umriss aufeinander abbildet. Innerhalb eines komplexeren Erkennungssystems muss im Rahmen der Hypothesen-Generierung eine große Zahl derartiger Berechnungen durchgeführt werden. Es ist von daher zwingend erforderlich, dass diese Berechnungen sowohl schnell als auch möglichst korrekt durchgeführt werden können. In diesem Bericht werden verschiedene Methoden verglichen und gezeigt, dass die Wahl der Methode erheblichen Einfluss auf die Zuverlässigkeit des Ergebnisses haben kann. Die beschriebenen Methoden lassen sich auch auf Straight Homogeneous Generalized Cylinders anwenden, obwohl deren Umriss im allgemeinen keinerlei Symmetrien aufweißt.

Comparison of Different Approaches for the Calculation of Projective Symmetry or the Axis of a SHGC

Sven Utcke
Arbeitsbereich Kognitive Systeme, Fachbereich Informatik,
Universität Hamburg, Germany
utcke@informatik.uni-hamburg.de

Abstract

Calculating the projective transformation which maps the two sides of a symmetric contour onto each other is an important step in the recognition of objects with symmetric contours, such as planar symmetric objects or surfaces of revolution. Within a more complex recognition system, many such calculations have to be performed as part of the hypotheses generation process, and it is therefore essential that the calculations are both fast as well as accurate. This paper compares different approaches and shows that the method selected can critically influence performance. The discussion trivially extends to finding the axis of a straight homogeneous generalised cylinder, even though its contour will not, in general, exhibit any symmetries.

1. Introduction

Objects with a symmetric contour form an important class for computer vision [3, 4, 6, 7, 10, 11, 13, 14, 17, 20] (see Sec. 2.1 for the definition of symmetry used in this paper). There are basically two families of such objects, namely planar (2D) symmetric objects and surfaces of revolution (SORs). However, many other man-made objects can be considered symmetric for practical purposes, including airplanes viewed from a distance [1], tools like a pair of pliers or scissors [3, 10], objects like spoons or ashtrays [11], or individual faces of complex objects [6]. And although the projections of straight homogeneous generalised cylinders (SHGC) generally exhibit *no* symmetry [15], we will later see that much of the methodology used for symmetric objects also applies to SHGCs.

Recognising symmetry in images can usually be divided into the following three steps: first (1) corresponding curve segments from each side of the contour need to be identified, e. g. based on projective invariants [4, 16], a necessary condition for symmetry. Next (2) an approximation of the transformation between the two sides of the contour will be computed, a harmonic homology [14]. Finally (3) we can refine our initial approximation of that homology, using, e. g., the snake-like algorithm given in [20]. This last (and time-consuming) step will, however, only be necessary if the contour tested successfully in the previous step.

This paper is only concerned with the second step. As complex scenes like the ones shown in [13, 20] can contain several hundred concavities, which in turn can lead to several hundred putative symmetries, it is important that the initial test for symmetry should be fast, reliable (only few false positives, and, if possible, no false negatives), and should provide a good initial estimate of the harmonic homology (so that the algorithm in the last step will require fewer iterations). This becomes even more important where automatic classification is applied to several thousand or even million images, such as with image-databases or web-crawlers. In our analysis of 48 competing algorithms we will see that the naive approach can create results which are grossly wrong and effectively unusable, and that even the algorithm used in [12, 14, 20] is not necessarily optimal. Although most of the algorithms have not previously been used for the detection of symmetry they are by no means new and used frequently for a range of computer vision tasks (most notably the detection of vanishing-points and -lines). It should therefore be instructive for the computer vision community as a whole to analyse their respective merits and weaknesses — this is particularly true as we will see the most widespread algorithm perform poorest.

The remainder of this paper is organised as follows: Section 2 gives a brief overview over the terminology and geometric constraints used, leading up to a description of the actual algorithms in Section 3. Section 4 gives a comparison and Sec. 5 a discussion of the results obtained.

2. Notation

2.1. Projective Symmetry

In 2D Euclidean space (Euclidean) symmetry is usually defined in terms of angles and lengths (this paper mainly deals with symmetry with respect to a line — also called axial-, reflectional- or mirror-symmetry). Angles and lengths are, however, inherently non-projective features — so much so that [5] explicitly denied the term symmetry for the very

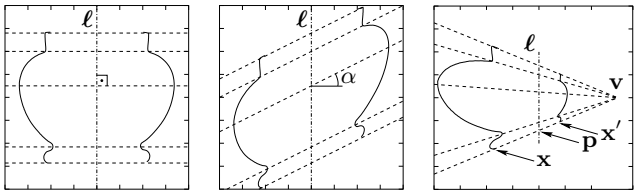


Figure 1: Euclidean symmetry (left), affine symmetry (middle), and projective symmetry (right).

same outlines for which we are now going to claim *projective symmetry*.

What we mean by the term *projective symmetry* is the existence of a non-trivial transformation H so that for corresponding points $\mathbf{x}$ and $\mathbf{x}'$ on the two sides of the contour we get $\mathsf{H}\mathbf{x} = \mathbf{x}'$, $\mathsf{H}\mathbf{x}' = \mathbf{x}$, and consequently $\mathsf{H}\mathsf{H}\mathbf{x} = \mathsf{I}_3\mathbf{x} = \mathbf{x}$ (the matrix I_3 is the identity matrix). It can be shown that only if H is a so called *plane harmonic homology*

$$\mathsf{H} = \mathsf{I}_3 - 2\,\frac{\mathbf{v}\boldsymbol{\ell}^\mathsf{T}}{\mathbf{v}^\mathsf{T}\boldsymbol{\ell}} \tag{1}$$

is it possible to find a projective transformation which will turn the outline into one exhibiting Euclidean symmetry. In (1) vector $\boldsymbol{\ell}$ is the image of the *axis of symmetry*, and vector $\mathbf{v}$, which we will call the *vertex*, denotes the direction of symmetry. It will be at infinity for *affine symmetry* (also called *skewed symmetry*) and orthogonal to the axis for *Euclidean symmetry*. The cross-ratio between the two contour-points $\mathbf{x}$ and $\mathbf{x}'$, the point on the axis between the two $\mathbf{p}$, and the vertex $\mathbf{v}$ is $\mathrm{cr} = \overline{\mathbf{px}}/\overline{\mathbf{px}'} \cdot \overline{\mathbf{vx}'}/\overline{\mathbf{vx}} = -1$, which is called *harmonic separation*.

2.2. Distinguished Points

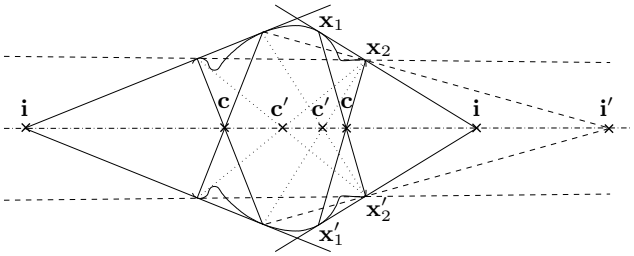


Figure 2: Features used: bitangent intersections are marked $\mathbf{i}$, crosspoints are marked $\mathbf{c}$. The points marked $\mathbf{i}'$ and $\mathbf{c}'$ are inter-pair features.

It is generally not trivial to decide which points from each side of the contour correspond to each other. Only for a small number of so called *distinguished points* is this correspondence easily found. Particularly useful in this respect are points of bitangency, i. e. tangent points of a line tangent at two points on the contour; the line is called a bitangent-line. Corresponding bitangent-lines will intersect each other on the (projection of the) axis of symmetry [5] — we will call this point the (bitangent) *intersection* throughout the remainder of this paper. It is marked $\mathbf{i}$ in Fig. 2. Additionally, for any two pairs of distinguished points $\{\mathbf{x}_1, \mathbf{x}_1'\}, \{\mathbf{x}_2, \mathbf{x}_2'\}$ the lines through the point-pairs $\{\mathbf{x}_1, \mathbf{x}_2'\}$ and $\{\mathbf{x}_2, \mathbf{x}_1'\}$ will intersect on the axis too (except for SHGCs), we will call this point a *crosspoint*. It is marked $\mathbf{c}$ in Fig. 2.

So far we considered each bitangent-pair separately, calculating only *intra-pair* features. However, if more than two distinguished points on each side of the contour are known, each pairing of two points and their corresponding points on the other side of the contour can be used to calculate additional intersections and crosspoints. Figure 2 shows a selection of such *inter-pair* features marked $\mathbf{i}'$ and $\mathbf{c}'$.

2.3. Object Classes

While planar symmetric objects will by definition have a symmetric outline, this is actually not the case for SHGCs and not immediately obvious for SORs, which can be described as a special SHGC. These are therefore described in more detail in the next two sections.

2.3.1 SHGCs

Straight homogeneous generalised cylinders (SHGCs) can be constructed from an arbitrarily shaped *reference cross-section* which gets scaled according to a *scaling function* while being swept along a straight axis (note that the axis need not even pass through the reference cross-section, in which case banana-like shapes would be the result). The contour of a SHGC will in general not display any kind of symmetry or even only qualitative symmetry [15], nor is there any straightforward transformation which would map one side of the contour onto the other without additional information. However, some properties which are true for symmetric contours also hold for SHGCs, and it is in particular the case that bitangents form distinguished points which can in general be identified and matched on both sides of the contour. What is more, for SHGCs too bitangent intersections will lie on the image of the SHGC's axis [15]. Unfortunately crosspoints do *not* lie on the axis, and results relating to those therefore cannot be used, nor is it possible to calculate the axis from any contour points other than distinguished points. This makes an accurate algorithm of particular interest for SHGCs.

2.3.2 Surfaces of Revolution

A surface of revolution (SOR) can be constructed from a circle which is translated along an axis through its centre and orthogonal to the circle's plane, and which is scaled at the same time by the so called scaling function or *generating function*. It is obvious from the above description that SORs are a special kind of SHGCs, and it is clear that

any result valid for SHGCs will also hold for SORs, but not vice-versa.

An SOR's contour generator will generally be a complicated, non-contiguous, non-smooth curve in 3D. It can nonetheless be shown that the corresponding 2D-contour will exhibit projective and, to a very good approximation, even affine symmetry [14]. This allows us to treat the contour of an SOR exactly like that of a planar symmetric object.

2.4. Error Measure

For contours related by a planar harmonic homology, it is directly possible to quantify the quality of the *calculated* planar harmonic homology even without ground truth — we can simply use H to map one side of the contour onto the other side, and use some error measure between the two curves to assess the goodness of fit. This is often done using the Hausdorff distance of the two contours, basically the maximum distance between the two sets. This is, however, not a very intuitive or descriptive measure, and we use instead the area between the two contours divided by the height of the object. This gives us the average difference between edgels on each side of the contour. Note that even for perfect symmetry this error-measure or residual will not be zero, as the position of the edgels along the contour will be noisy. We can therefore only expect a value in the same order as the standard deviation of our edge detection algorithm (or, more accurately, $\sqrt{2}$ times the standard deviation, as both sides will be subject to measurement errors). Here we use a very simple implementation of the Canny edge finder [2] to extract the edges. Its standard deviation on grey-level images is in the order of $0.1\,\mathrm{pxl} \leq \sigma \leq 0.3\,\mathrm{pxl}$.

3. Algorithms

Rather than trying to solve for the plane harmonic homology all at once (for which usually no closed form solution exists), it is far easier to compute separate results for the axis and vertex. Doing so basically means to compute a best-fit line through a number of points (the axis), and the most likely intersection of a number of lines (the vertex). This is a standard problem in computer vision (and consequently should have a standard solution), but nonetheless many different algorithms for the solution of this problem are in widespread use — we will see in Sec. 4 that the most commonly used algorithm also tends to be the least reliable. In the following, we distinguish algorithms by geometric- and error-model; a further subdivision is possible by the number and type of features used. All these different aspects are discussed below for the calculation of the axis.

3.1. Axis Calculation

Crosspoints and intersections, which we will use to calculate the axis, are basically measurements in the image-plane (although not necessarily within the image) and as such can be characterized by their position and an error-distribution describing the measurement error. The position is in the following represented by triples $\mathbf{p} = (x, y, z)^\mathsf{T}$, where the location in the image plane is calculated as $(x/z, y/z)$. For a unique representation an additional constraint is needed, and we use $z = 1$ to represent Euclidean coordinates or $x^2 + y^2 + z^2 = 1$ for homogeneous coordinates, which represents point in 2D as points on a 3D unit-sphere. As for the error-distribution, we distinguish between two models: a model *implicit* in algorithms which minimise the sum of squared distances, namely independently, identically, and isotropically distributed (iiid) features with Gaussian distribution, and an *explicit* model which uses standard error propagation to derive the intersections' and crosspoints' covariance matrices from iiid Gaussian distributed bitangent points. More complicated models will be discussed in Sec. 5 and 6. Although different geometric- and error-models are used we can nonetheless for all four combinations calculate the best-fitting axis $\boldsymbol{\ell}$ through some intersections and crosspoints $\mathbf{p}_i$ by minimising

$$F(\boldsymbol{\ell}) = \frac{1}{N}\sum_{i=1}^{N} \frac{\boldsymbol{\ell}^\mathsf{T}\mathbf{p}_i\mathbf{p}_i^\mathsf{T}\boldsymbol{\ell}}{\boldsymbol{\ell}^\mathsf{T}\boldsymbol{\Sigma}_{\mathbf{p}_i}\boldsymbol{\ell}} + \lambda(\boldsymbol{\ell}^\mathsf{T}\mathsf{W}\boldsymbol{\ell} - 1) \qquad (2)$$

for different values of W and $\boldsymbol{\Sigma}_{\mathbf{p}_i}$. The W is either the identity-matrix I_3 (for homogeneous coordinates) or has its last diagonal element set to 0 (for Euclidean coordinates); the $\boldsymbol{\Sigma}_{\mathbf{p}_i}$ is either the same matrix as W (implicit error model), a full covariance-matrix (explicit model, homogeneous coordinates) or a covariance-matrix with it's last row and column set to 0 (explicit model, Euclidean coordinates). The minimum can be calculated explicitly if an implicit error-model is used, or else using Kanatani's unbiased estimator [9].

In addition to the geometric- and error-model we can further subdivide algorithms by features used. Using only intersections will result in an algorithm suitable to SHGCs as well as symmetric contours, while the additional use of crosspoints excludes most SHGCs. Finally we can differentiate between algorithms which only use the information from corresponding bitangent pairs (intra-pair relations) and such which use all pairwise combinations of bitangent-points (inter-pair). Throughout this paper we only use bitangent-points, but the use of inflections would provide additional information and will be discussed in Sec. 6. For symmetric outlines finally it is possible to construct an arbitrary number of additional features, see [11].

We are coding the approaches as follows:

8	4	2	1
Error Model	Geom. Model	Object Model	Combinations
expl./impl.	$xyz/xy1$	SOR/SHGC	inter/intra
1/0	1/0	1/0	1/0

This results in 16 different methods for the calculation of the axis, numbered 0–15. Alg. 3 is the one most commonly used, while Alg. 10 was, e. g., used in [12–14].

3.2. Vertex Calculation

So far we have only implemented three different algorithms. We will, however, see in Section 4 that for SORs the actual vertex-model chosen makes little difference (a more complex model would be required for planar symmetric objects which we didn't actually test on). The three models are labelled 0 (an affine model without explicit error-model, implemented as the average angle towards the vertex at infinity), 2 (a projective model using Euclidean coordinates and no explicit error-model), and 14 (a projective model using homogeneous coordinates and an explicit error-model) — the latter two are calculated by substituting $\boldsymbol{\ell}$ with $\mathbf{v}$ in (2) and $\mathbf{p}_i$ with the line through two corresponding distinguished points. As for the number of features used, each bitangent-pair creates exactly 2 lines through the vertex; pairing non-corresponding distinguished points is not possible.

4. Results and Discussion

Each of the 16 algorithms for the calculation of the axis and 3 algorithms for the calculation of the vertex were run on a total of 49 images[1] of 6 SORs (see Fig. 3) which have previously appeared in publications about the recognition of SORs [12–14]. This resulted in 48 different values for the harmonic homology in each image and $48 \times 49 = 2352$ different harmonic homologies overall. For each homology we also calculated the residual as described in Sec. 2.4 and used this to determine the relative goodness of fit for each approach.

What are the results we would expect? As stated in the previous section, algorithms basically differ in three areas: by the number of features used (intra-pair versus inter-pair and intersections only versus intersections and cross-points); by geometric model (Euclidean coordinates versus homogeneous coordinates); and by error model (no explicit model versus simple explicit model). Generally speaking, we would expect the algorithms to perform better which use more features, homogeneous coordinates and an explicit error-model. However, it is important to realise that here, as everywhere else, no silver bullet exists — we will in fact later see that the particularities of most of the imaged objects considerably skew the outcome. We therefore use

[1]The relevant contours and bitangents were selected by hand, so as not to confound the comparison with additional issues.

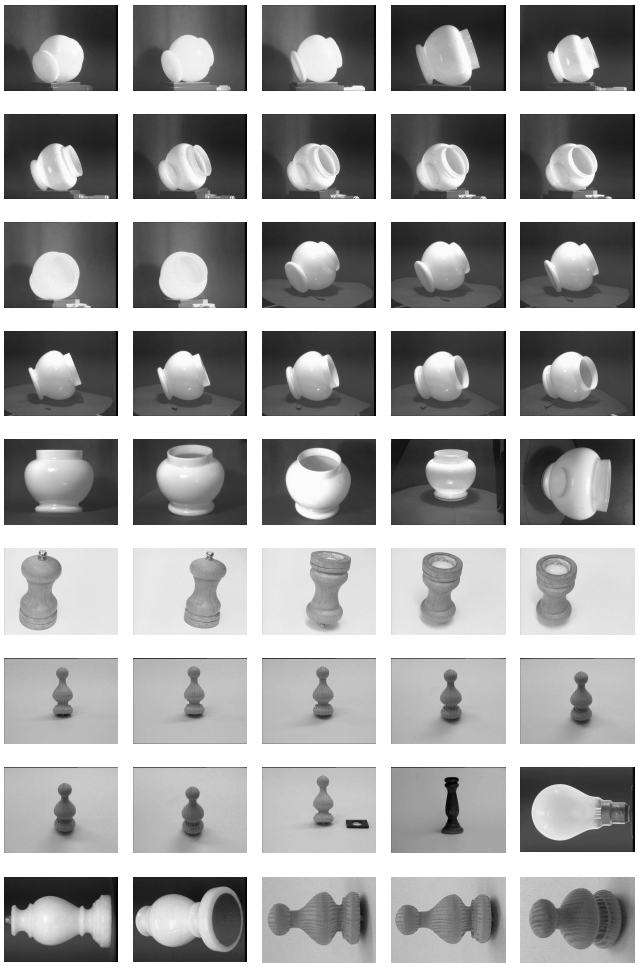
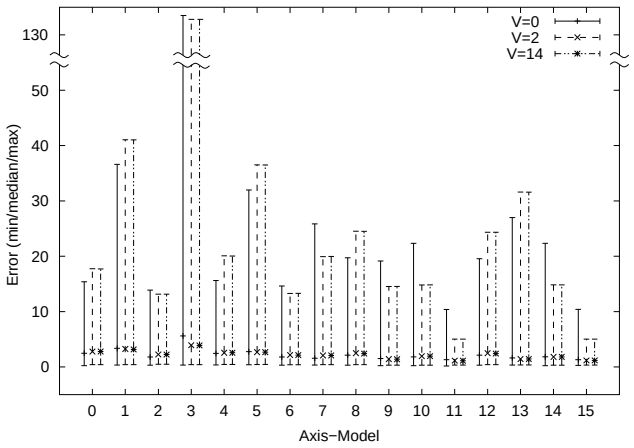


Figure 3: 45 of the 49 images of SORs used.

several different measures of fitness to assess the quality of the algorithms. These measures are either based on the actual residual calculated, or on an algorithm's relative performance compared to all other algorithms, its ranking. Both will be described in more detail in the next two sections, while Sec. 5 discusses the reasons for some unexpected observations.

4.1. Residual

Figure 4 shows the range (minimum, median, and maximum) of residuals encountered for each of the 48 combinations of axis- and vertex-model, ordered by axis-model. In the following we will mostly be interested in the maximum residual calculated, as some algorithms could clearly result in unacceptably wrong results, which of course needs to be avoided if the number of false negatives is to be kept small; only then will we consider the median residual, which gives information about the algorithms' average performance. The minimum residual is of little interest to us, as it will always be in the order of $\sqrt{2}$ times the edgels'



	0	*1*	2	3	*4*	*5*	6	*7*	*8*	*9*	10	11	*12*	*13*	14	15
max	18.	41.	13.	123	20.	37.	13.	20.	25.	15.	15.	**5.0**	24.	32.	15.	**5.0**
med	2.8	3.2	2.2	3.9	2.6	2.7	2.2	2.1	2.4	**1.4**	2.0	**1.1**	2.4	**1.4**	**1.8**	**1.2**
min	.40	.40	.49	.41	.46	.44	**.39**	.41	.41	**.32**	**.32**	**.34**	.41	**.36**	**.32**	**.35**

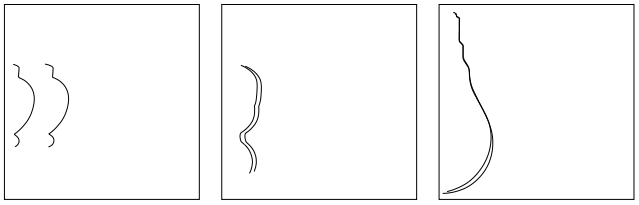
Figure 4: Range of residuals. For each algorithm the minimum, median and maximum residual are plotted — note the discontinuity along the ordinate. The table shows the numerical values for the second best vertex-model.

variance for any algorithm worth considering.

A number of things can be learned from Fig. 4: first, the choice of the axis-model has a much bigger influence on the overall performance than the choice of the vertex-model — this can be seen from the fact that for each axis-model the error-bars for all three vertex-models are very similar, while differing wildly between different axis-models. We will therefore group algorithms by axis-model in the following.

Next, it is instructive to look for any systematics in the maximum residuals — in order to avoid false negatives and to speed up subsequent processing the residual should not become too big. Ignoring axis-model 3 for now, we will observe that the axis models number 1, 5, and 13 can give quite unreliable results. These are the algorithms which use all inter-pair combinations of bitangent-points but no crosspoints (and are therefore suitable for SHGCs too). Only axis-model 9, which is the forth model in that series does not follow this trend. However, it too can give unacceptable results as can be seen from Fig. 5, middle. The counterintuitive observation that models using more points (1, 5, and 13) perform worse than models using fewer points (0, 4, and 12) is due to errors in those additional points and will be explained in Section 5.

Ignoring the axis-models 1, 5, 9, and 13 from Fig. 4 we next concentrate on the models 3, 7, 11, and 15; these models all use inter-pair intersections and crosspoints, i. e. the maximum number of features. It is therefore interesting to note that the models 11 and 15, which use an explicit error model, perform extremely well (as we expected them to do), while the models 7 and 3 in particular perform extremely poor. Again, an explanation for this will be given in Section 5. It should be noted that the axis-model number 3 (quadratic distance error model, Euclidean geometry, using all possible combinations of bitangent-intersections and crosspoints) is the model most commonly used for the calculation of a line through several points — but clearly the solutions found by this approach cannot be relied on at all. Even the median residual calculated with this method is higher than that from any other model.



Axis-model 3
Vertex-model 2
$r = 122.763$

Axis-model 9
Vertex-model 2
$r = 14.5296$

Axis-model 15
Vertex-model 14
$r = 5.03427$

Figure 5: Cases of maximum residual for three axis-models (using the best vertex-model). The graphs show the left contour mapped onto the right one — ideally only one contour would be visible.

We can now ignore axis-models 3, 7, 11, and 15 too and concentrate on the remaining models, which use solely *intra-pair* intersections (0, 4, 8, 12) or *intra-pair* intersections and crosspoints (2, 6, 10, 14). Now three more interesting observations can be made: first, we notice that all models from the second set (using cross-points) perform better than the corresponding algorithms from the first set (without crosspoints). This is in keeping with the assumption that more features are better. Secondly, we notice that the algorithms without an explicit error model (0–7) actually exhibit smaller maximum errors than the ones based on such a model (8–15), which is counterintuitive (the median and minimum error however behave roughly as expected). It is not clear to us what conclusions can be drawn from this. Finally, it is interesting to note that for the axis-models 0, 4, 8, and 12, which use only intra-pair intersections, the affine vertex-model (0) results in a considerably smaller maximum residual than the other two models, while for axis-models which also use crosspoints it results in higher maximum errors. Again, the median and minimum error do not mirror this behaviour, and the implications are unclear.

For SHGCs finally only the Alg. 0, 1, 4, 5, 8, 9, 12, and 13 can be used. Based on Fig. 4 we notice that the maximum error seems to be best contained using only intra-pair intersections, and seems otherwise quite independent from the geometric- or error-model used. However, if we also take the median and minimum residual into consideration, things change and the two axis-models 9 and 13 (inter-pair, explicit error model) perform best.

To sum up: as expected a high number of features is indeed preferable, but only if used together with an explicit

Table 1: Histogram of how often out of 49 runs each algorithm was among the best 3.

Vertex Model	Axis Model																Sum
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	3	2	7	2	2	4	8	14	4	9	5	8	3	7	5	8	91
2		2		2		1	3	2		1	2	1		1	1	6	22
14		3		3			2	2	1	2	2	5	1	2	2	9	34
Sum	3	7	7	7	2	5	*13*	**18**	5	*12*	9	*14*	4	10	8	**23**	147

Table 2: Histogram of how often out of 49 runs each algorithm was among the worst 3.

Vertex Model	Axis Model																Sum
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	8	11	1	19	5	2	1	2	1	1	2	1		2	2	1	59
2	5	11		19		1		1	1		1		1	1	4		45
14	6	11		18		1		2			2		1	1	1		43
Sum	*19*	**33**	1	**56**	5	4	1	5	2	1	5	1	2	4	7	1	147

error-model; without such a model the emphasis should be put on accurate rather than numerous features — this is in direct conflict with the assumption underlying many algorithms that more features are always better. For objects with a symmetric contour, Alg. 11 or 15, which use the most features and an explicit error-model, are clearly the best choice. For SHGCs no crosspoints can be used, and here Alg. 9 and 13 (inter-pair intersections, explicit error-model) will best contain the average error, although the maximum residual can still be quite high, as can be seen from Figure 5. There examples for the actual homologies computed in the case of maximum error are given for three of the algorithms (3, 9, and 15). We can see that even the algorithm with the lowest maximum residual (axis-model 15, vertex-model 14 — the most refined model using the most features) will produce noticeable errors for some input-constellations. The results are, however, much more usable, as can be seen from Fig. 5, right, which shows the example with the maximum residual for this algorithm. It should be noted that the object is actually not quite symmetric, although in this particular case 8 out of the 48 algorithms tested performed better. Ranking the relative performance of all algorithms is indeed another possibility to determine fitness, and will be done in the next section.

4.2. Rank

Although any algorithm might return the smallest residual for one particular outline, we would nonetheless expect that the better an algorithm is suited for the task, the more often should it show up among the best N algorithms; conversely the more often it is placed among the worst N algorithms, the more unsuitable will we deem this algorithm. Table 1 lists, for each algorithm, how often it was observed among the best 3 algorithms. We see that Algorithm 15 (most features, homogeneous coordinates, explicit error-model) is indeed the most performant algorithm, but closely followed by Alg. 7 (most features, homogeneous coordinates, *no* explicit error-model). As expected the Algorithm 11 (most features, Euclidean coordinates, explicit error model) performs quite reasonably too, while the good ratings of Alg. 6 and 9 come as a surprise. With regard to the vertex-model it might seem surprising that the affine model performs so well, but the two sides of an SOR are indeed to a very good approximation affinely related, as was shown in [14].

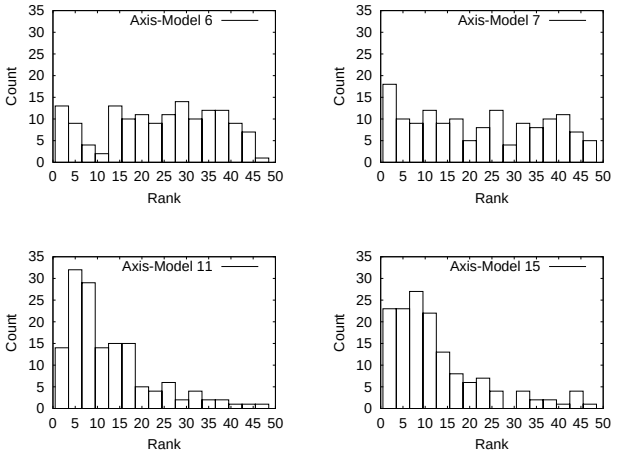


Figure 6: Histograms of rank for axis-models 6 (top left), 7 (top right), 11 (bottom left), and 15 (bottom right).

Additional insights can be gained if we group algorithms by features used. Algorithms based on an explicit error model account for 58 % of the best 3 algorithms; using homogeneous coordinates accounts for 56 %; using intersections and crosspoints accounts for 67 % of the best 3 algorithms, and using inter-pair features for 65 %. All this suggests that more features and better geometric- and error-models do indeed improve the performance, but not considerably.

The image becomes somewhat clearer if we also consider the 3 worst algorithms, shown in Table 2. Here, using no explicit error-model accounts for 84 % of the worst 3 algorithms; using Euclidean coordinates for 80 %.

The usefulness of an explicit error-model becomes even more apparent if we look at a histogram of the ranks achieved with the Algorithms 6, 7, 11, and 15 (which, according to Tables 1 and 2, all performed similarly, while in theory 11 and 15 should exhibit superior performance). Figure 6 shows a clear difference between the algorithms which do not use an explicit error model (6 and 7, top row) on the one hand and the ones which do (11 and 15, bottom row) on the other. The former (as do most other models) show a nearly uniform distribution, which means that they are similarly likely to be among the N best as well as the N worst algorithms, while the latter's distribution looks

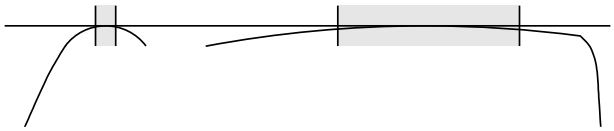


Figure 7: The accuracy with which bitangent-points can be located depends on the contour's curvature in that region. This has little influence on bitangent-intersections, but can considerably influence the position of crosspoints and inter-pair intersections.

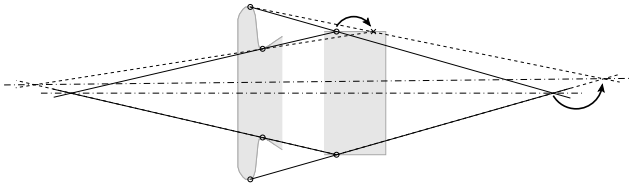


Figure 8: *Inter-pair* intersections can be quite inaccurate for many common objects, yet at the same time far enough away from the object to have considerable influence on the location of the axis.

somewhat like a Poisson distribution, with good ranks much more likely than bad ones. This shows that the overall likelihood of an acceptable result is much higher for axis-models which use as many features as possible *together* with an explicit error-model.

We believe this to be strong evidence that the use of an explicit error-model, at least when used together with many features of varying quality, can considerably improve an algorithm's performance. Only the use of an explicit error model can prevent the extremely erroneous results exhibited by Alg. 3, and the use of an explicit error-model together with many features will in most cases give better results than any other approach.

5. Discussion

We have seen in Section 4.1 and 4.2 that using more features and an explicit error-model will indeed overall improve the performance of an algorithm (as we expected). However, we also noticed some inconsistencies, and these will be explained in the following.

Considering only the maximum residuals in Fig. 4 we noticed that the axis-models 1, 5, and 13 (*inter-pair* intersections, no crosspoints) can perform noticeably worse than the models 0, 4, and 12 (*intra-pair* intersections only), even though for the former many more features can be calculated. Although counterintuitive at first, this is easily explained based on the particular shape of most of our test-objects. Consulting Fig. 3 we notice that most objects contain sections of extremely low curvature (nearly straight in fact), and that in most cases a bitangent will touch the object in that area. This is true for the neck of the first object and the foot of the second and sixths, which together contribute about 66 % of all contours. The position of a bitangent-point along such a low-curvature segment can only be calculated quite inaccurately, see Fig. 7. This has very little influence on the direction of that particular bitangent (whose accuracy basically depends on the distance between the two points of tangency), and consequently little influence on the position of the bitangent-intersection (and, in consequence, little influence on the axis-models 0, 4, 8, and 12); it can, however, greatly influence the position of inter-pair intersections (1, 5, 9, and 13) — Figure 8 gives an example. Given N bitangent-pairs, only 1 intersection (containing the erroneous bitangent-point) will be calculated correctly, but $2(N-1)$ intersections will be incorrect (compare Fig. 2). Additionally, many of those intersections will be far away from the object, and will consequently have high influence on the final result (in particular if Euclidean coordinates were used). It is therefore not surprising that results can become nearly arbitrarily wrong. Using crosspoints (models 3, 7, 11, 15) can mediate this effect; while their position will be wrong too, they will actually be on the other side of the axis and therefore offset some of the effect. The correct solution of course would be an error model which correctly computes a point's accuracy along the bitangent based on the contour's curvature around the point. However, as curvature is impossible to compute accurately for low-curvature contours [19], such a model will be difficult to implement (see Sec. 6).

6. Conclusions and Outlook

A computer-vision system which aims to locate, identify and possibly reconstruct planar symmetric objects, SORs or even SHGCs needs to calculate the objects' axes and, for the first two classes of objects, the harmonic homology relating the two sides of the contour. A very accurate algorithm for the calculation of the harmonic homology has been known for a long time [20], however, this algorithm is based on numerical minimisation and, depending on the initial estimate of its parameters, might require many iterations in order to converge. While not a problem for a single outline (each iteration is quite fast), this can nonetheless severely limit its usefulness in the case of cluttered images containing many possibly symmetric objects, or in the case where huge numbers of images need to be analysed, as for image-database applications or within webcrawlers. Having a faster algorithm which serves both to weed out many wrong matches, as well as providing the following stages with accurate initial values, can provide a considerable speedup. Having an algorithm available which is based solely on distinguished points is also of particular importance for SHGCs, where no better algorithm is known.

In this paper 48 different algorithms for this intermediate step have been compared and it has been demonstrated that using an explicit error model can considerably improve the

results, in particular where many features of varying accuracy are used. This can mean the difference between completely useless results in the naive — but widely used — case on the one side and highly reliable results on the other.

There is, however, still room for improvements. We have seen in Sec. 5 that the proposed method, while already of very high accuracy even in the worst case, could most likely be further improved by the use of a curvature-based error model. While easily enough done in theory it unfortunately suffers from the fact that curvature for low-curvature regions cannot be calculated accurately in practice [19]. We are currently working on an error-model which would nonetheless be able to take these effects into account.

Once such an error-model is in place it would also enable us to use an additional kind of distinguished point, namely inflections, isolated points of zero curvature. All the properties of bitangent-points given above also hold for inflections; it is however difficult to accurately compute their orientation (needed for the intersection) and position (needed for crosspoints, inter-pair intersections and the calculation of the vertex). A curvature-based error-model would allow us to quantify this uncertainty and take it into account.

This paper concentrated on the calculation of an SOR's axis of symmetry. In order to become truly comprehensive, additional tests should be run on SHGC's and planar symmetric objects; the latter will also allow a more detailed study of algorithms for the calculation of the vertex.

Additionaly, a different error measure could be developed based on the fact that for truly symmetric objects the crossratrio between two corresponding points, the vertex and a point on the axis along a line through the other three points should have a crossratio of -1; such an error measure could be based, e. g., on the work done in [18].

Due to the duality between lines and points it is straightforward to extend the above to point-symmetry.

References

[1] K. Arbter, W. E. Snyder, H. Burkhardt, and G. Hirzinger. Application of affine-invariant fourier descriptors to recognition of 3-d objects. *IEEE Trans Pattern Anal Mach Intell*, 12(7):640–647, July 1990.

[2] J. F. Canny. A computational approach to edge detection. *IEEE Trans Pattern Anal Mach Intell*, 8(6):679–698, Nov. 1986.

[3] T.-J. Cham and R. Cipolla. A local approach to recovering global skewed symmetry. In *Proc Int Conf Pattern Recognit*, volume I, pages 222–226, Jerusalem, Israel, Oct. 1994. IAPR, IEEE CS Press.

[4] R. W. Curwen, J. L. Mundy, and C. V. Stewart. Recognition of plane projective symmetry. In Proc Int Conf Comput Vision [8], pages 1115–1122.

[5] D. A. Forsyth, J. L. Mundy, A. Zisserman, and C. A. Rothwell. Recognising rotationally symmetric surfaces from their outlines. In G. Sandini, editor, *Proc Eur Conf Comput Vision*, LNCS, pages 639–647. Springer Verlag, 1992.

[6] A. D. Gross. Toward object-based heuristics. *IEEE Trans Pattern Anal Mach Intell*, 16(8):794–802, Aug. 1994.

[7] A. D. Gross and T. E. Boult. Analyzing skewed symmetries. *Int J Comput Vision*, 13(1):91–111, 1994.

[8] IEEE CS. *Proc Int Conf Comput Vision*, Bombay, Jan. 1998. Narosa Publishing House, New Delhi.

[9] Y. Kanazawa and K. Kanatani. Optimal line fitting and reliability evaluation. *IEICE Transactions on Information & Systems*, E79–D(9):1317–1322, Sept. 1996.

[10] Y. Lei and K. C. Wong. Detection and localisation of reflectional and rotational symmetry under weak perspective projection. *Pattern Recognit*, 32(2):167–180, Feb. 1999.

[11] D. P. Mukherjee, A. Zisserman, and J. M. Brady. Shape from symmetry—detecting and exploiting symmetry in affine images. In *Philosophical Transactions of the Royal Society of London, Series A*, volume 351, pages 77–106, 1995.

[12] J. Mundy, A. Liu, N. Pillow, A. Zisserman, S. Abdallah, S. Utcke, S. Nayar, and C. Rothwell. An experimental comparison of appearance and geometric model based recognition. In *Proc. Object Representation in Computer Vision II*, LNCS 1144, pages 247–269. Springer-Verlag, 1996.

[13] J. L. Mundy, C. Huang, J. Liu, W. Hoffman, D. A. Forsyth, C. A. Rothwell, A. Zisserman, S. Utcke, and O. Bournez. MORSE: A 3D object recognition system based on geometric invariants. In M. Kaufmann, editor, *Image Understanding Workshop*, pages II:1393–1402, Monterey, CA, November 13–16 1994. ARPA.

[14] N. Pillow, S. Utcke, and A. Zisserman. Viewpoint-invariant representation of generalized cylinders using the symmetry set. *Image Vision Comput*, 13(5):355–365, June 1995.

[15] J. Ponce, D. Chelberg, and W. B. Mann. Invariant properties of straight homogeneous generalized cylinders and their contours. *IEEE Trans Pattern Anal Mach Intell*, 11(9):951–966, Sept. 1989.

[16] C. A. Rothwell, A. Zisserman, D. A. Forsyth, and J. L. Mundy. Canonical frames for planar object recognition. In G. Sandini, editor, *Proc Eur Conf Comput Vision*, LNCS, pages 757–772. Springer Verlag, 1992.

[17] F. Ulupinar and R. Nevatia. Shape from contour: Straight homogeneous generalized cylinders and constant cross section generalized cylinders. *IEEE Trans Pattern Anal Mach Intell*, 17(2):120–135, Feb. 1995.

[18] S. Utcke. Grouping based on projective geometry constraints and uncertainty. In Proc Int Conf Comput Vision [8], pages 739–746.

[19] S. Utcke. Error-bounds on curvature estimation. In *Scale Space*, Isle of Skye, Scotland, UK, June 2003. BMVA. accepted.

[20] A. Zisserman, J. Mundy, D. Forsyth, J. Liu, N. Pillow, C. Rothwell, and S. Utcke. Class-based grouping in perspective images. In *Proc Int Conf Comput Vision*, pages 183–188, Cambridge, MA, USA, June 1995. IEEE CS, IEEE CS Press.