



0031-3203(95)00095-X

THE FAMILY OF UNIVERSAL AXES

JA-CHEN LIN

Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan,
Republic of China

(Received 8 July 1994; in revised form 18 January 1995; received for publication 3 July 1995)

Abstract—The family of universal axes (UA) is introduced in this paper. This family has infinitely many members, each of which generates a method that is universal in the sense that each generated method can define shape orientation for almost any kind of shape and once the numerical computing system has been set up, the users of this orientation detection system need not supply the system the information about whether a given shape is mirror-symmetric, rotationally symmetric, irregular, etc. Some mathematical properties of the proposed UA are investigated and numerical techniques for implementing UA are discussed. The method of universal principal axes (UPA) introduced recently is just one of the infinitely many members of the proposed family of UA. The UPA method yields at least two orientations for each shape and thus must be combined with another non-universal single-orientation method in order to reduce the time needed to match irregular shapes. Most members of the proposed UA family do not need to be supplemented by a single-orientation method in this way.

Shape orientation	Universal method	Numerical techniques
Learning by mathematical property		Single-orientation

1. INTRODUCTION

Detecting orientations of two-dimensional (2D) shapes is often necessary to match two shapes. Although there are many tools for detecting orientations, these tools are seldom universal,⁽¹⁾ i.e. each tool can define orientations for certain types of shapes only. For example, mirror-symmetric axes^(2,3) do not exist if the given shape is not mirror-symmetric and the methods of fold principal axes,⁽⁴⁾ generalized principal axes⁽⁵⁾ or fold-invariant shape-specific points⁽⁶⁾ can only be used with rotationally symmetric shapes.⁽⁷⁾ Similarly, the method using conventional principal axes^(8,9) and the method using the line through the centroid and radius weighted mean⁽¹⁰⁾ give no direction if the given shape is a rotationally symmetric shape [the reasons for this are explained in reference (1)]. Lin therefore designed theoretically the universal principal axes (UPA) in reference (1), with the emphasis that the method of UPA has the following two desired universal properties. First, UPA can be used to define orientation for almost every type of shape (in our experience, only circular disks and rings do not have UPA, but defining orientations for circular disks and rings is meaningless). Second, when UPA are used to detect shape orientation, the user need not supply the information about whether the given shape is rotationally symmetric, mirror-symmetric or irregular. Hereafter, any orientation detection method with these two features will be called a universal method. The present paper is intended

to answer the following two questions: “Theoretically speaking, how many universal methods are there?” and “Numerically speaking, how should we implement these universal methods?” The first question is answered in the next paragraph. To answer the second question, we describe several mathematical properties of certain types of shapes and compare the numerical values with the analytical values claimed by these properties. We then use the difference between these values to get the numerical rule. The details will be given in Section 3.

To answer the first question, we introduce here a family of axes $\{UA_\mu | \mu \geq 0\}$, where UA_μ is read as the universal axes of version μ , such that, as will be seen later, each version is associated with a universal method having the two universal properties mentioned above. (Therefore, there are infinitely many universal methods available.) The construction of UA_μ is described by the algorithm below. This algorithm is similar to the one used in reference (1) and the major change is made in Steps 1 and 2 so that the version parameter μ can play a role in the computation. Note that the format and notation used in the algorithm of reference (1) are followed below so that the reader can get the idea more quickly, and the details of the proof can also be borrowed from reference (1).

Theoretical algorithm (Universal Axes of Version μ)

Step 0. Translate the coordinate system so that the origin O becomes the centroid of the given shape S .

Step 1. Compute:

$$\begin{aligned} x_\mu^{(l)} + iy_\mu^{(l)} &= \iint_S (\sqrt{x^2 + y^2})^\mu \left(\frac{x + iy}{\sqrt{x^2 + y^2}} \right)^l dx dy \\ &= \iint_S r^\mu e^{il\theta} dx dy \end{aligned} \quad (1)$$

for $l = 1, 2, 3, \dots$, until an l_1 appears where l_1 is the smallest $l \in \{1, 2, 3, \dots\}$ making $x_\mu^{(l)} + iy_\mu^{(l)} \neq 0$.

Step 2. Compute the polar angle $\Theta_\mu \in [0, 2\pi]$ so that:

$$R_\mu e^{i\Theta_\mu} = x_\mu^{(l_1)} + iy_\mu^{(l_1)} \quad (2)$$

with R_μ being the absolute value of $x_\mu^{(l_1)} + iy_\mu^{(l_1)}$.

Step 3. Let:

$$\theta_j = \left(\frac{\Theta_\mu}{l_1} \right) + (j-1) \frac{2\pi}{l_1} \quad \text{for } j = 1, 2, \dots, l_1. \quad (3)$$

The l_1 half lines starting from the centroid O and having directional angles $\{\theta_j\}_{j=1}^{l_1}$ are the expected l_1 universal axes of version μ for shape S . The directions of shape S are then defined to be these axes. (Therefore, shape S has l_1 directions, $(2\pi/l_1)$ apart from one another.)

The reason we use l_1 directions instead of a single direction is similar to the reason there are l_1 UPA; refer to Section 2 of reference (1) for a detailed explanation.

In the algorithm above, the parameter μ can be either a constant or a function $\mu(l)$ of l . Although any non-negative function of domain $N = \{\text{natural numbers}\}$ can be used as μ , here, for simplicity, we shall consider only those μ in the set:

$$H = \{\mu = \mu(l) = al + b | a, b \text{ are constants and } al + b \geq 0 \text{ when } l \in N\}. \quad (4)$$

Any μ chosen from this set can be taken as a version number. For example, version 4 and version $2.5l + 3.7$ are both allowed. Of course, version $2.5l + 3.7$ will mean that the last integral in equation (1) becomes $\iint_S r^{2.5l+3.7} e^{il\theta} dx dy$.

The definition of UA_μ implies that UA_μ always exist unless the l_1 defined in the algorithm is ∞ , i.e.

$$x_\mu^{(l)} + iy_\mu^{(l)} = 0 \quad \text{for all } l = 1, 2, 3, 4, \dots \quad (5)$$

Due to its strictness, very few shapes can meet requirement (5). To the author's knowledge, only circular disks and rings meet (5) and hence have no UA_μ . However, as stated earlier, defining shape orientations for circular disks and rings is meaningless.

The remainder of this paper is organized as follows. In Section 2 we show that the proposed UA is qualified in defining shape orientations, because the position and orientation defined by the UA are unchanged relative to the shape if the shape is translated, scaled or rotated. In Section 3 we discuss in detail the numerical techniques needed for implementing the proposed algorithm. Many examples are given to illustrate these techniques. Due to the limitation of the space, we only demonstrate a version that is arbitrarily taken, namely, Version $2l$, in each of the examples throughout this paper. However, the numerical techniques introduced

can be used for other versions as well. Several procedures that summarize these numerical techniques are included at the end of the last three subsections of Section 3. The relationship between UPA and the proposed UA is discussed in Section 4. A summary of the paper is given in Section 5.

2. QUALIFICATION OF UA IN DEFINING SHAPE ORIENTATIONS

In this section we prove that for each specified μ , the corresponding UA_μ really has the ability to detect shape orientation. That is, the relative position and direction of the defined orientations (relative to the shape) are unchanged when the shape is translated, scaled or rotated. Before giving a formal mathematical proof, we present the experimental results of two examples; the version μ used in both examples is $\mu(l) = 2l$. In the first example, the shape is a handwritten letter "P". We used a scanner to capture the shifted, rotated and reduced shapes of the original shape. The algorithm was used once on each of these three shapes and the original shape. The computer results obtained show that $l_1 = 1$ in each case. The detected orientation was then sketched for each of these four shapes (see Fig. 1 for illustrations). We can see that the detected orientation is indeed relatively invariant. In the second example, the shape used is a square. We repeated the experiment and found that the method works again (see Fig. 2 for the results). The only change of the experimental results is that there are four orientations per square because the detected value of l_1 is 4 in the second example. The reason the orientations are relatively invariant is due to the following property which holds for any shape and any version.

Property 1. The l_1 orientations defined are relatively unchanged if the shape is scaled, translated or rotated.

Proof.

(i) Translation

In Step 0 of the algorithm we always shift the origin back to the centroid, hence the x, y values used in the

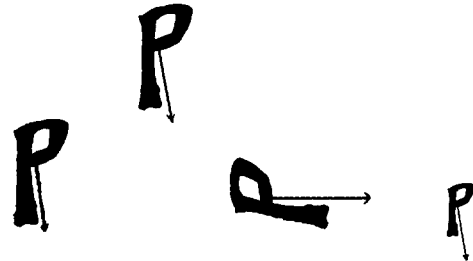


Fig. 1. The detected UA is relatively invariant if the shape is translated, rotated, or scaled. The last three shapes were created by a scanner, which grabbed the translated, rotated, and reduced shapes of the input shape (the leftmost shape).

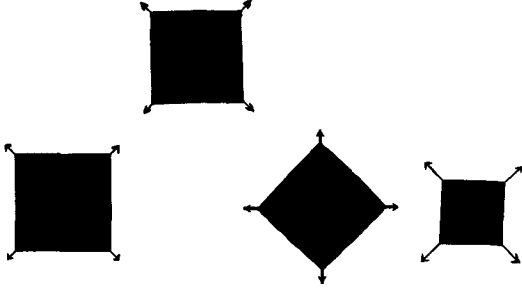


Fig. 2. The experiment done in Fig. 1 is repeated again using a square as the input shape (the leftmost shape).

algorithm are in fact “relative” coordinates (relative to the centroid of the given shape) instead of absolute coordinates. Therefore, translation of the shape does not affect the orientation (s).

(ii) Scaling and rotation

Here we prove only the part of the statement concerning scaling, because the part concerning rotation can also be proved in a similar way by the calculus technique of changing variables. Let shape S be enlarged (reduced) by a factor λ to obtain a new shape S_{new} which is λ times larger (smaller). Then equation (1) implies that the $x_{\mu}^{(l)} + iy_{\mu}^{(l)}$ of shape S_{new} is:

$$x_{\mu}^{(l)} + iy_{\mu}^{(l)}|_{S_{\text{new}}} = \int \int_{S_{\text{new}}} (\sqrt{x^2 + y^2})^{\mu} \left(\frac{x + iy}{\sqrt{x^2 + y^2}} \right)^l dx dy. \quad (6)$$

If we change variables in the integration so that the integral domain is S again, i.e. if we let $x = \lambda t$ and $y = \lambda w$ in the above integral, we will have:

$$\begin{aligned} x_{\mu}^{(l)} + iy_{\mu}^{(l)}|_{S_{\text{new}}} &= \int \int_{S_{\text{new}}} (\sqrt{(\lambda t)^2 + (\lambda w)^2})^{\mu} \\ &\quad \times \left(\frac{(\lambda t) + i(\lambda w)}{\sqrt{(\lambda t)^2 + (\lambda w)^2}} \right)^l (\lambda^2 dt dw) \\ &= \lambda^{\mu+2} \int \int_S (\sqrt{t^2 + w^2})^{\mu} \left(\frac{t + iw}{\sqrt{t^2 + w^2}} \right)^l dt dw \\ &= \lambda^{\mu+2} (x_{\mu}^{(l)} + iy_{\mu}^{(l)}|_S) \\ &= \lambda^{\mu(l)+2} (x_{\mu}^{(l)} + iy_{\mu}^{(l)}|_S), \end{aligned} \quad (7)$$

where $\mu = \mu(l)$ is used to denote the fact that μ could be a function of l . Therefore, the $x_{\mu}^{(l)} + iy_{\mu}^{(l)}$ of shape S_{new} is zero if and only if the $x_{\mu}^{(l)} + iy_{\mu}^{(l)}$ of shape S is zero. The l_1 defined in Step 1 of the algorithm is therefore unchanged. Moreover, if we let $l = l_1$ in equation (7), we can see that $x_{\mu}^{(l_1)} + iy_{\mu}^{(l_1)}|_{S_{\text{new}}}$ and $x_{\mu}^{(l_1)} + iy_{\mu}^{(l_1)}|_S$ have the same polar angle, although their absolute values are distinct. The Θ_{μ} defined in equation (2) and hence the l_1 orientations defined in Step 3 of the algorithm are therefore unchanged. $\square$

3. THREE NUMERICAL TECHNIQUES FOR THE IMPLEMENTATION OF UA

The numerical techniques needed to implement the proposed UA are provided in this section. In reference (1) there is no discussion about how to implement UPA numerically in a digital system. Therefore, the techniques discussed here in Section 3 are all new. The reader can get the idea of the section more quickly if he reads the titles of all subsections first. Also note that a summary of the proposed numerical techniques is given in Section 3.5.

3.1. The major difficulty to be overcome in the implementation of UA

In practice, a given shape is usually sampled in a, say, 512-by-512 grid. Without loss of generality, we assume that each pixel is one unit wide and one unit long. The computation of:

$$x_{\mu}^{(l)} + iy_{\mu}^{(l)} = \int \int_S (\sqrt{x^2 + y^2})^{\mu} \left(\frac{x + iy}{\sqrt{x^2 + y^2}} \right)^l dx dy$$

then becomes:

$$\sum_{j=1}^J (\sqrt{x_j^2 + y_j^2})^{\mu} \left(\frac{x_j + iy_j}{\sqrt{x_j^2 + y_j^2}} \right)^l = \sum_{j=1}^J (r_j)^{\mu} e^{il\theta_j}$$

if there are J sampled points in S . Here, the polar length and polar angle of each sampled point (x_j, y_j) are denoted by r_j and θ_j , respectively. The major difficulty in implementing the UA_{μ} algorithm is that, numerically speaking, it is not easy to determine the value of l_1 , because it is hard to judge whether a number $x_{\mu}^{(l)} + iy_{\mu}^{(l)}$ is zero or not. To overcome this difficulty, we propose below three techniques, as listed in the titles of Sections 3.2–3.4, respectively. All these three techniques are new.

3.2. Numerical technique 1: Using the normalized counterpart $\tilde{x}_{\mu}^{(l)} + i\tilde{y}_{\mu}^{(l)}$ (of $x_{\mu}^{(l)} + iy_{\mu}^{(l)}$) to identify l_1

The first technique we suggest is to identify the l_1 in Step 1 of the algorithm as the smallest positive integer l making $\tilde{x}_{\mu}^{(l)} + i\tilde{y}_{\mu}^{(l)}$ non-zero. Here, the normalized counterpart $\tilde{x}_{\mu}^{(l)} + i\tilde{y}_{\mu}^{(l)}$ of $x_{\mu}^{(l)} + iy_{\mu}^{(l)}$ is defined as:

$$\tilde{x}_{\mu}^{(l)} + i\tilde{y}_{\mu}^{(l)} = \frac{x_{\mu}^{(l)} + iy_{\mu}^{(l)}}{\int \int_S (\sqrt{x^2 + y^2})^{\mu} dx dy}, \quad (8)$$

a complex number whose polar angle is identical to that of $x_{\mu}^{(l)} + iy_{\mu}^{(l)}$. Theoretically speaking, $x_{\mu}^{(l)} + iy_{\mu}^{(l)}$ is zero if and only if $\tilde{x}_{\mu}^{(l)} + i\tilde{y}_{\mu}^{(l)}$ is zero. (Therefore, the smallest positive integer l making $x_{\mu}^{(l)} + iy_{\mu}^{(l)}$ non-zero is also the smallest positive integer l making $\tilde{x}_{\mu}^{(l)} + i\tilde{y}_{\mu}^{(l)}$ non-zero.) Numerically speaking, however, $\tilde{x}_{\mu}^{(l)} + i\tilde{y}_{\mu}^{(l)}$ is easier to use in identifying l_1 , because the normalization from $x_{\mu}^{(l)} + iy_{\mu}^{(l)}$ to $\tilde{x}_{\mu}^{(l)} + i\tilde{y}_{\mu}^{(l)}$ gives the two benefits discussed below.

3.2.1. The normalization makes the numerical values of the theoretical zeros more like zeros (see Fig. 3). For example, when the version used is version 2l, the



l	$ x_\mu^{(l)} + iy_\mu^{(l)} $	$ \tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)} $
1	5.3E4 ○	3.8E-4 *
2	8.5E9 ○	6.5E-3 *
3	1.5E14 ○	9.9E-3 *
4	2.8E17 ○	1.3E-3 *
5	3.4E24	9.7E-1
6	6.8E26 ○	1.2E-2 *
7	1.8E31 ○	1.8E-2 *

Fig. 3. The computed $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ performs better than the computed $|x_\mu^{(l)} + iy_\mu^{(l)}|$ because the theoretical values of the numbers marked with ○ or * are all zeros.

computed sequences of $|x_\mu^{(l)} + iy_\mu^{(l)}|$ and $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ for the star shape sketched in Fig. 3 are as listed in the figure. In this paper, 3.8E-4 should be read as 0.00038, etc. Based on Property 2 (to be introduced later), $|\tilde{x}_\mu^{(5)} + i\tilde{y}_\mu^{(5)}|$ is the only possible value in the sequence $\{|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}| | l = 1, 2, \dots, 7\}$ to be non-zero theoretically. An analogous conclusion holds for $|x_\mu^{(5)} + iy_\mu^{(5)}|$ and $\{|x_\mu^{(l)} + iy_\mu^{(l)}| | l = 1, 2, \dots, 7\}$. In other words, the theoretical values of those entries marked with ○ and * in Fig. 3 should all be zero. However, the sequence $|x_\mu^{(l)} + iy_\mu^{(l)}|$ does not reflect this fact at all, although the sequence in $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ reflects this fact to some extent. The job of obtaining a threshold (to distinguish zeros from non-zeros) is therefore easier if the sequence $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ is used. We will thus use $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ to identify l_1 hereafter.

3.2.2. The normalization makes the threshold scale-independent. There is another benefit of using $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ instead of $|x_\mu^{(l)} + iy_\mu^{(l)}|$. If shape S is scaled by a factor of λ to generate a new shape S_{new} , then equation (7) implies that $|x_\mu^{(l)} + iy_\mu^{(l)}|$ is $\lambda^{\mu+2}$ times larger (smaller). However, the value of $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ will not change at all, because the magnitude of the denominator $\iint_S (\sqrt{x^2 + y^2})^\mu dx dy$ in equation (8) can also be proven to be $\lambda^{\mu+2}$ times larger (smaller) and hence we may eliminate the $\lambda^{\mu+2}$ factor in the numerator. In other words:

$$\begin{aligned} |\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|_{S_{\text{new}}} &= \frac{x_\mu^{(l)} + iy_\mu^{(l)}|_{S_{\text{new}}}}{\iint_{S_{\text{new}}} (\sqrt{x^2 + y^2})^\mu dx dy} \\ &= \frac{\lambda^{\mu+2} |x_\mu^{(l)} + iy_\mu^{(l)}|_S}{\lambda^{\mu+2} \iint_S (\sqrt{x^2 + y^2})^\mu dx dy} = |\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|_S, \end{aligned}$$

although the sizes of the two shapes S_{new} and S are distinct. This property is very useful in determining a threshold T that can be used to judge whether a number $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ evaluated by a computer is zero or not, for the threshold taken will then be scale-independent.

3.3. Numerical technique 2: Determining threshold T using certain shapes whose $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ can be proven to be zero for some l

It has been suggested in Section 3.2 that the value of l_1 should numerically be identified as the smallest positive integer l , making $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ non-zero (instead of making $|x_\mu^{(l)} + iy_\mu^{(l)}|$ non-zero). The remaining problem is therefore to determine a threshold T which can

be used to judge whether a computed number $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ is zero or not. To find a suitable value for the threshold T , the technique we suggest here is a learning procedure (also called training procedure), which inspects the computer results for certain types of shapes for which the values of some $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ can be proven to be theoretically zero. This learning procedure gives us a feeling of what a zero might look like and hence gives us a guideline to follow in determining the value of threshold T .

3.3.1. Some maths properties useful in creating shapes with $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}| = 0$ for some l . The following three properties hold for all versions. The author omits the proofs because they are quite trivial {the proof of Property 3 is similar to the proof given in equation (24) of reference (1), the proof of Property 2 uses the fact that the polar radii are periodic with period $2\pi/n$ [a proof related to version 1 using discrete coordinate is given in Theorem 1 of reference (11)]; finally, the proof of Property 4 uses only calculus techniques and the angularly periodic property of the shapes illustrated in the property}. These properties can be used to generate some shapes for which the theoretical values of $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ are zero for some l . In Properties 2 and 4 the definition⁽⁶⁾ of rotationally symmetric shapes will be needed.

Definition. A shape S is called an n -fold rotationally symmetric shape if it becomes identical to itself after being rotated around its centroid through any multiple of $2\pi/n$. (This n should be taken to be as large as possible. For example, the shape sketched in Fig. 2 is a four-fold rotationally symmetric shape.)

Property 2. If shape S is an n -fold rotationally symmetric shape and if l is not a multiple of n , then $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}| = |x_\mu^{(l)} + iy_\mu^{(l)}| = 0$ for every version μ .

Property 3. If shape S is a circular disk or ring, then $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}| = |x_\mu^{(l)} + iy_\mu^{(l)}| = 0$ for every natural number l and version μ .

Property 4. (See Fig. 4 for illustrations.) For every natural number n and for every positive constant R , define an n -fold rotationally symmetric shape with the first fold of the shape being:

$$\begin{aligned} \{(x, y) | x + iy = re^{i\theta}, \text{ where } 0 \leq r \leq R \text{ if} \\ 0 \leq \theta < \frac{\pi}{n} \text{ and } r = 0 \text{ if } \frac{\pi}{n} \leq \theta < \frac{2\pi}{n}\}. \end{aligned}$$

If $l \in \{2n, 4n, \dots\}$, i.e. l/n is an even number, then $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}| = |x_\mu^{(l)} + iy_\mu^{(l)}| = 0$ for every version μ .

Notice that the combination of Properties 2 and 4 implies that any n -fold rotationally symmetric shape of the type sketched in Fig. 4 cannot have $|x_\mu^{(l)} + iy_\mu^{(l)}| \neq 0$ ($|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}| \neq 0$) unless l is an odd multiple of n .

Remark. Property 2 above extends Theorem 1—a theorem which yields a fold number detector—of reference (11) from Version 1 to Version μ . [To see this,

l	The computed values of $ \tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)} $		
	n = 2	n = 3	n = 4
1	1.1E-4 *	6.3E-4 *	3.6E-6 *
2	[6.3E-1]	7.8E-3 *	4.6E-7 *
3	1.4E-3 *	[6.3E-1]	1.3E-3 *
4	7.7E-3 *	6.1E-3 *	[6.3E-1]
5	2.9E-3 *	4.4E-3 *	8.9E-4 *
6	2.2E-1	8.6E-3 *	3.3E-6 *
7	3.2E-3 *	2.7E-3 *	2.1E-3 *
8	3.3E-3 *	3.4E-3 *	9.3E-3 *
9	1.9E-3 *	2.2E-1	1.6E-3 *
			

Fig. 4. The computed $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ for the shapes defined in Property 4. The n used here are 2, 3 and 4, respectively.

just compare equation (3) of reference (11) with the formula in our Section 3.1]. Property 3 extends equation (24)—an equation which was used to verify that the UPA does not exist for each ring and circular disk—of reference (1) from Version 1 to Version μ . Property 4 was never reported in the literature.

3.3.2. A learning procedure to determine threshold T . To figure out what the so-called “zero” looks like in a digital system, the author used a computer to generate several (digitized) shapes mentioned in Properties 2–4, such as circular disks, rings and polygons. We then computed the values of $\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}$ for each of these shapes. The version used in the demonstrations is a version arbitrarily taken, namely, version 2l, as stated earlier at the end of Section 1. With the help of Properties 2–4, we may claim that many of the computed values are theoretically zero. As in Fig. 3, those entries marked with a “*” in Figs 4 and 5 should be treated as zeros theoretically. The shapes generated by the computer were then read in by a scanner and the author repeated the experiment on these scanned images to see what would happen. The results are shown in Fig. 6; it can be seen that the computed values of the entries marked with a “*”, i.e. the theoretically-zero values, become larger, but there is no noticeable change in the values of the entries that are not marked with a “*” (this is a major reason why the author inspects the values $\{|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|\}$ directly instead of inspecting the ratios $\{|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|/|\tilde{x}_\mu^{(l-1)} + i\tilde{y}_\mu^{(l-1)}|\}$ when the value of l_1 is to be determined. See the four fold shapes in Figs 5 and 6 for the illustration). The values of threshold T should be taken as a number larger than the maximum of those computed values marked with a “*” in Figs 3–6 and yet smaller than the minimum of those not marked with a “*”. Of course, the more shapes tested, the better our estimate of T becomes. We tested more than 60 shapes in addition to those in Figs 3–6 and finally decided to use $T = 0.1$, i.e. 1.0E-1. To summarize Section 3.3, the author gives below the learning procedure to determine threshold T for a specified version μ .

l	The computed values of $ \tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)} $		
	n = 2	n = 3	n = 4
1	1.4E-3 *	7.0E-17 *	1.1E-3 *
2	4.1E-3 *	4.2E-18 *	3.8E-3 *
3	[9.4E-1]	7.2E-17 *	7.7E-3 *
4	1.9E-2 *	[9.7E-1]	7.3E-3 *
5	2.8E-3 *	1.9E-16 *	[9.9E-1]
6	8.8E-1	2.7E-16 *	1.5E-2 *
7	3.9E-2 *	2.3E-16 *	1.9E-2 *
8	1.2E-3 *	9.8E-1	2.5E-2 *
9	8.1E-1	3.1E-16 *	2.0E-2 *
			

l	The computed values of $ \tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)} $		
	n = 2	n = 3	n = 4
1	1.8E-5 *	1.1E-17 *	1.1E-17 *
2	9.9E-3 *	1.3E-17 *	1.3E-17 *
3	3.2E-3 *	2.7E-17 *	2.7E-17 *
4	5.8E-3 *	1.6E-3 *	1.6E-3 *
5	3.0E-3 *	1.6E-17 *	2.6E-18 *
6	[4.2E-1]	5.9E-17 *	5.2E-17 *
7	1.6E-3 *	2.6E-17 *	2.6E-17 *
8	2.0E-2 *	1.1E-3 *	1.1E-3 *
9	6.4E-3 *	6.6E-18 *	6.6E-18 *
			

Fig. 5. The computed values of $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ for some shapes discussed in Properties 2 and 3. Note that the shapes used in Fig. 3 through 5 are all computer-generated, and hence discrete instead of continuous.

The Learning Procedure to determine threshold T The learning pool

Set up a pool for learning. The pool should contain a circular disk, a ring, some shapes defined in Property 4 and some other rotationally symmetric shapes.

The steps

- Take a shape from the pool.
- Compute the $\{|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}| : l = 1, 2, 3, \dots\}$ for this shape.
- If this shape is a circular disk or a ring, mark “*” for all computed $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ to denote the fact that they are all theoretically zero. If this shape is neither a circular disk nor a ring, then use Properties 2 and 4 to mark “*” for those computed $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$, which are theoretically zero.
- Repeat (i)–(iii) until every shape in the pool has been checked.
- Let max denote the maximum of all those computed $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ marked with “*” and min the minimum of all those computed $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ not marked with “*”. (In other words, let max denote the maximum of all those computed $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ which should be theoretically zero and min the minimum of all those computed $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ which should be theoretically non-zero.)
- Assign T as a number between max and min , say, assign T as $(max + min)/2$.

l	The computed values of $ \tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)} $		
1	4.8E-3 *	2.0E-3 *	3.4E-3 *
2	1.9E-2 *	1.9E-3 *	2.2E-2 *
3	9.5E-1	2.3E-2 *	1.8E-2 *
4	4.0E-2 *	9.7E-1	3.2E-2 *
5	5.2E-3 *	2.1E-2 *	9.9E-1
6	8.8E-1	6.7E-2 *	3.6E-2 *
7	7.3E-2 *	4.5E-2 *	5.3E-2 *
8	7.7E-2 *	9.8E-1	3.8E-2 *
9	8.1E-1	4.1E-2 *	8.2E-2 *

l	The computed values of $ \tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)} $		
1	4.9E-5 *	5.4E-5 *	8.4E-4 *
2	4.4E-2 *	4.4E-2 *	2.2E-2 *
3	4.6E-3 *	1.4E-2 *	1.5E-3 *
4	2.1E-2 *	7.7E-3 *	2.4E-3 *
5	3.4E-3 *	4.4E-3 *	3.7E-3 *
6	4.3E-1	3.8E-3 *	2.4E-3 *
7	5.6E-3 *	1.7E-3 *	4.3E-3 *
8	9.9E-2 *	3.4E-3 *	2.1E-3 *
9	6.6E-3 *	5.3E-3 *	1.2E-3 *

Fig. 6. The computed values of $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ for some scanner-grabbed shapes. The computer-generated shapes sketched in Fig. 5 are read in again by a scanner to generate these shapes.

3.4. Numerical technique 3: Using another threshold t to handle the shapes whose l_1 are 1

Although the learning procedure stated in Section 3.3 can provide us a threshold T , that T can only handle the shapes whose l_1 are greater than 1. In other words, another threshold t is needed in order to handle the shapes whose l_1 are 1. The reason that two thresholds are needed is that the values of $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ are quite different between the shapes with $l_1 = 1$ and the shapes with $l_1 > 1$, as is explained below in Sections 3.4.1 and 3.4.2. The detail of obtaining the second threshold, i.e. t , is illustrated in Section 3.4.3. A procedure to determine t is provided in Section 3.4.4.

3.4.1. The values of $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ are quite different between the shapes with $l_1 = 1$ and the shapes with $l_1 > 1$. The (theoretically non-zero) values of $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}| = |\tilde{x}_\mu^{(1)} + i\tilde{y}_\mu^{(1)}|$ for shapes for which the theoretical values of l_1 are 1 are often not of magnitudes similar to those of the $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ of the shapes with $l_1 > 1$. For example, the shapes in Fig. 7 all have $l_1 = 1$, but their values of $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ are only of the order 10^{-2} , somewhat smaller than the 10^{-1} order appearing in Figs 3–6. (To see this, just compare the entries enclosed by rectangular boxes in Fig. 7 with those enclosed in Figs 3–6; the enclosed entries denote the $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ for each corresponding shape throughout this paper.) In fact, the $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ listed in Fig. 7

l	The computed values of $ \tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)} $		
1	2.2E-2	1.5E-2	1.1E-2
2	9.4E-1	4.9E-1	2.4E-1
3	8.2E-2	6.5E-1	4.1E-1
4	9.2E-1	1.1E-1	1.2E-1

Fig. 7. The computed values of $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ for some shapes of which the value of l_1 is 1. Note that the values of $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ would usually become larger when the shapes are those with $l_1 > 1$ (cf. the entries enclosed by rectangular boxes in Fig. 3 through 7).

are even smaller than the $3.9E-2$ (which is theoretically zero) of the $\tilde{x}_\mu^{(7)} + i\tilde{y}_\mu^{(7)}$ of the three-fold shape sketched in Fig. 5. This means $|\tilde{x}_\mu^{(1)} + i\tilde{y}_\mu^{(1)}|$ will often be misjudged as zero when l_1 is in fact one. The situation is even worse when version $\mu(l)$ has the property that $\mu(l)$ is almost equal to one when $l = 1$. For example, when for version 0.99 the three-fold shape sketched in Fig. 5 has $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}| = |\tilde{x}_\mu^{(3)} + i\tilde{y}_\mu^{(3)}| = 8.8E-1$ and $|\tilde{x}_\mu^{(7)} + i\tilde{y}_\mu^{(7)}| = 1.3E-2$, which are theoretically non-zero and zero, respectively; whereas the theoretically non-zero values of $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ of the shapes sketched in Fig. 7 are about $1.0E-4$. A single threshold T cannot simultaneously identify $1.0E-4$ as non-zero and $1.3E-2$ as zero.

3.4.2. Why is $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ much smaller for the shapes whose l_1 are 1? The reason $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ has distinct behaviour for $l_1 = 1$ and $l_1 > 1$ is as follows. Note that if $\mu(l) = 1$ when $l = 1$, then $\tilde{x}_\mu^{(1)} + i\tilde{y}_\mu^{(1)}$ and hence $|\tilde{x}_\mu^{(1)} + i\tilde{y}_\mu^{(1)}|$ is theoretically zero by equation (1) and Step 0 of the algorithm. Therefore, if the version $\mu(l)$ used is a version that does not have $\mu(1) = 1$, the value of the $|\tilde{x}_\mu^{(1)} + i\tilde{y}_\mu^{(1)}|$ is still affected somewhat, namely, the closer $\mu(1)$ is to 1, the closer $|\tilde{x}_\mu^{(1)} + i\tilde{y}_\mu^{(1)}|$ is to 0.

3.4.3. Using two thresholds t and T to handle the different behaviour of $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ between $l_1 = 1$ and $l_1 > 1$. Throughout this subsection, it is assumed that $\mu(l) \neq 1$ when $l = 1$ (otherwise, $\tilde{x}_\mu^{(1)} + i\tilde{y}_\mu^{(1)}$ is theoretically zero for all shapes, as was explained in Section 3.4.2, and there is nothing to worry about because l_1 can never be 1 in this case). In order to overcome the difficulty caused by the different behaviour of $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ between $l_1 = 1$ and $l_1 > 1$, the third technique we suggest is to use two thresholds, t and T , instead of a single T . The t is used to judge whether a computed $\tilde{x}_\mu^{(1)} + i\tilde{y}_\mu^{(1)}$ is zero or not, while the T is used to judge whether a computed $\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}$, where $l > 1$, is zero or not. The method for determining the value of T was described earlier in Section 3.3. To determine the value of t , we generate several shapes that are not rotationally symmetric, e.g. shapes that

are irregular or mirror-symmetric with only one symmetric line. (See Fig. 7 for illustrations. The middle shape there is a mirror-symmetric shape which is symmetric about a line called a symmetric line.) (Note that we cannot use mirror-symmetric shapes with more than one symmetric line here because they will become rotationally symmetric. For example, the four shapes sketched in Figs 3 and 4 have 5, 2, 3 and 4 symmetric lines, respectively, and these four shapes are both mirror-symmetric and rotationally symmetric.) Shapes that are not rotationally symmetric seldom have $\tilde{x}_\mu^{(1)} + i\tilde{y}_\mu^{(1)} = 0$ theoretically. Therefore, the threshold t should be smaller than almost every (see the next sentence for the exception) computed value of $|\tilde{x}_\mu^{(1)} + i\tilde{y}_\mu^{(1)}|$ corresponding to these shapes and yet larger than the computed values of $|\tilde{x}_\mu^{(1)} + i\tilde{y}_\mu^{(1)}|$ for the shapes generated using Properties 2–4. [Should the computed value of $|\tilde{x}_\mu^{(1)} + i\tilde{y}_\mu^{(1)}|$ of a non-rotationally-symmetric shape be extremely small, i.e. smaller than the computed value of $|\tilde{x}_\mu^{(1)} + i\tilde{y}_\mu^{(1)}|$ of a shape generated using Properties 2, 3 or 4, then it is suggested that this non-rotationally-symmetric shape be discarded from the learning procedure determining t , for this extremely small (computed) number should be treated as a zero because it is smaller than a computed number which has already been treated as a zero.]

Example

In our example (Figs 4–7 and the rightmost column of Fig. 3), t should be smaller than the values appearing in the row corresponding to $l = 1$ in Fig. 7, but larger than the values appearing in the rows corresponding to $l = 1$ in Figs 3–6. In other words, t should be larger than $4.8E-3$ and smaller than $1.1E-2$. (The range between $4.8E-3$ and $1.1E-2$ is not wide, but this is because we wish to count those distorted shapes in Fig. 6 as some perfect shapes stated in Properties 2–4; otherwise, the range would be wider. Note especially that the distorted three-leaf rose in Fig. 6 is in fact not rotationally symmetric; its centroid is even not at the intersection point of the three leaves, as will be seen in Fig. 8.) After testing more shapes that are not rotationally symmetric, we decided to use $t = 0.008$, i.e. $8E-3$. Recall that $T = 0.1$ and the version used is $2l$. With these t and T , the detected UA for many shapes under version $2l$ are sketched in Fig. 8. It can be seen that UA can handle any kind of shape, including irregular, mirror-symmetric, and rotationally symmetric shapes.

Although the version used in Figs 3–7 is version $2l$, we have also tested many other versions, such as versions 0, 0.99, 2, $l - 1$, 4 and 8, and have found that the numerical techniques introduced in this section also work with these other versions.

3.4.4. A learning procedure to determine threshold t . To summarize Section 3.4, we give below the learning procedure to determine threshold t for a specified version μ .

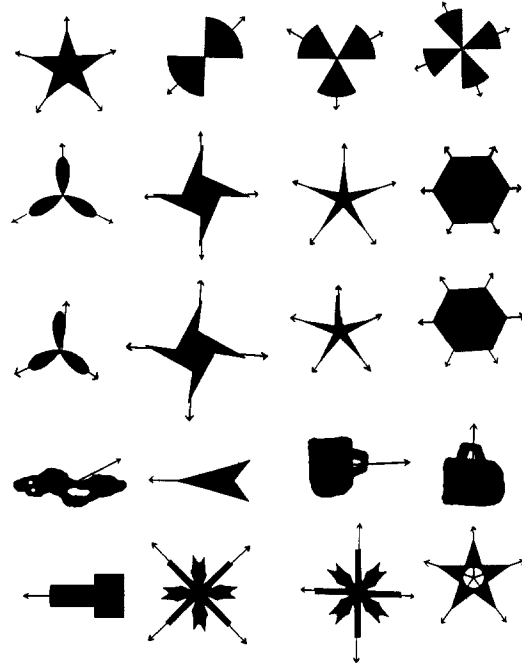


Fig. 8. The detected UA for the shapes sketched in Fig. 3 through 7 (according to the order) and several other shapes. The version used is version $\mu(l) = 2l$, and the thresholds are $t = 0.008 = 8E - 3$ and $T = 0.1 = 1E - 1$. Note that the four shapes in row three are scanner-generated and have some distortion, but the proposed UA still works to a certain extent (cf. row two and row three).

The learning procedure to determine threshold t .

The steps

- (i) Let $\max'$ be the maximum of the computed $|\tilde{x}_\mu^{(1)} + i\tilde{y}_\mu^{(1)}|$ for all shapes used in the learning procedure determining T , a procedure given in Section 3.3.2. (In our example, $\max'$ is the maximum of the first row of Figs 4–6, including the $|\tilde{x}_\mu^{(1)} + i\tilde{y}_\mu^{(1)}|$ of Fig. 3.)
- (ii) Let $\min'$ be the minimum of the computed $|\tilde{x}_\mu^{(1)} + i\tilde{y}_\mu^{(1)}|$ for several non-rotationally-symmetric shapes being used for learning. (In our example, $\min'$ is the minimum of the first row of Fig. 7.)
- (iii) Assign t as a number between $\max'$ and $\min'$, say, assign t as $(\max' + \min')/2$.

3.5. A summary of the proposed numerical techniques

For the paper presented here, the new aspects contributed by the author in Section 3 can be stated as: the three numerical techniques are new, whereas the three properties in Section 3.3 are partly new (see the remark given at the end of Section 3.3.1).

For the reader's benefit, the numerical techniques needed to implement the UA are summarized below in two parts. It is assumed that the version used is given and never changed. Part 1 corresponds to the set-up of the system and Part 1 is executed once and only once. As long as the threshold values T and t were found using Part 1, the reader may erase the program coding Part 1 from the computer forever. Part 2, however, is

a long-resident program staying in the computer system. Each time a new shape whose direction is to be detected, the reader executes Part 2 only.

The numerical techniques to implement UA (for a specified version μ)

PART1: System set-up.

Step (i) Use the learning procedure given in Section 3.3.2 to obtain threshold T .

Step (ii) Use the learning procedure given in Section 3.4.4 to obtain threshold t .

PART2: The numerical algorithm to obtain UA_μ for a new shape, i.e. a shape not used in Part 1.

Step 0'. Identical to Step 0 (of the theoretical algorithm).

Step 1'. If the computed value of $|\tilde{x}_\mu^{(1)} + i\tilde{y}_\mu^{(1)}|$ is larger than t , then assign $l_1 = 1$, else assign l_1 as the smallest $l \in \{2, 3, \dots\}$ making the computed value of $|\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}|$ larger than T .

Step 2'. Compute the polar angle $\Theta_\mu \in [0, 2\pi)$ so that $\tilde{R}_\mu e^{i\Theta_\mu} = \tilde{x}_\mu^{(l_1)} + i\tilde{y}_\mu^{(l_1)}$ with $\tilde{R}_\mu$ being the absolute value of $\tilde{x}_\mu^{(l_1)} + i\tilde{y}_\mu^{(l_1)}$.

Step 3'. Identical to Step 3 (of the theoretical algorithm).

Note that the Θ_μ in Step 2' is identical to the Θ_μ in equation (2) because the polar angle of $\tilde{x}_\mu^{(l_1)} + i\tilde{y}_\mu^{(l_1)}$ is identical to that of $\tilde{x}_\mu^{(l_1)} + i\tilde{y}_\mu^{(l_1)}$ by equation (8). Therefore, Step 2' may also be said to be identical to Step 2. (The author writes Step 2' here instead of Step 2, just because $\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}$, instead of $x_\mu^{(l)} + iy_\mu^{(l)}$, is computed in Step 1'.)

Also note that although μ is allowed to be very large theoretically, we suggest that an excessively large version not be used (e.g. version 64), because it might cause overflow when $x_\mu^{(l)} + iy_\mu^{(l)}$, the numerator of Equation (8), is evaluated. To save computational effort, we recommend versions 2 and 0 [see equation (1)]. As for version 1, although its computational effort is not high, it gives at least two orientations for each shape, a weakness that UPA also has (see the next section for reasons).

4. RELATIONSHIP BETWEEN UA AND UPA

If we let $\mu(l) = l$, the UA generated are in fact the universal principal axes (UPA) introduced in reference (1). Equation (1) now implies that:

$$\begin{aligned} x_\mu^{(1)} + iy_\mu^{(1)} &= \iint_S (x + iy) dx dy \\ &= \iint_S x dx dy + i \iint_S y dx dy = 0 + i0 = 0 \end{aligned} \quad (9)$$

for every shape because of Step 0 of the algorithm. In other words, l_1 is at least two. [That is why the algorithm given in reference (1) chooses l_1 from the set $\{2, 3, 4, \dots\}$, while that in this paper chooses l_1 from the set $\{1, 2, 3, \dots\}$.] The UPA system therefore always generates at least two axes and hence at least two

orientations for every shape. Thus, in reference (1) the UPA system was combined with another orientation-detection (non-universal) method in order to produce a method that usually gives a single orientation when the shape is not rotationally symmetric. (The reason fewer orientations are preferred is that if several given shapes are to be matched or registered, a smaller number of orientations means that fewer matching steps are needed.) In the proposed UA system, for most versions there is no need to employ other orientation-detection methods. If the version $\mu = \mu(l)$ used satisfies $\mu(1) \neq 1$, the UA_μ system proposed in this study will automatically generate a single orientation for almost every shape that is not rotationally symmetric.

Although Lin defined UPA analytically in reference (1), he did not discuss how to implement UPA numerically. Nevertheless, the numerical techniques introduced in Section 3 of the present paper can be used for the UPA system because the UPA system is just a special case of the UA system, namely, UA with version l . In other words, we recommend replacing $x_\mu^{(l)} + iy_\mu^{(l)}$ by $\tilde{x}_\mu^{(l)} + i\tilde{y}_\mu^{(l)}$ and determining the threshold T using the learning procedure based on Properties 2–4. The only change is that there is no so-called threshold t in the UPA system because of equation (9).

5. CONCLUSIONS

The universal axes (UA) introduced in this study have infinitely many versions. Each version alone can be used as a universal orientation-defining method for 2D shapes. Therefore, there are infinitely many universal methods available. Among them, any version $\mu = \mu(l)$ with $\mu(1) \neq 1$ is better than a version with $\mu(1) = 1$ in the sense that the latter always give at least two orientations for each shape and thus result in more matching steps when the detected orientations are used later in matching or registration. It has been observed that the method of universal principal axes (UPA) introduced in reference (1) is just one of the infinitely many versions introduced here. More precisely, letting $\mu(l) = l$ in our UA algorithm will generate UPA. Note that $\mu(1) = 1$ for UPA and hence UPA gives at least two orientations for each shape, a weakness that most of the other versions will not have. Besides showing analytically that each version can be used to define 2D shape orientation(s), we have also provided three numerical techniques for implementing the proposed UA practically when digitization error exists. These numerical techniques are based on some mathematical properties of UA. Experimental results are included. While there was no discussion in reference (1) of how to implement UPA if the given shapes are discrete instead of continuous, the techniques introduced in this paper can also be used to implement UPA practically. Just like UPA, each member of the proposed UA family has the ability to define shape orientation for almost every kind of shape, no matter if a given shape is mirror-symmetric, rotationally symmetric, irregular and so forth.

Acknowledgements—This work was supported by the National Science Council, Republic of China, under contract number NSC84-2213-E-009-042. The author wishes to thank the referee for valuable comments about the writing of the paper.

REFERENCES

1. J. C. Lin, Universal principal axes: an easy-to-construct tool useful in defining shape orientations for almost every kind of shape, *Pattern Recognition* **26**, 485–493 (1993).
2. P. T. Highnam, Optimal algorithm for finding the symmetries of a planar point set, *Inf. Process. Lett.* **22**, 219–222 (1986).
3. M. J. Atallah, On symmetry detection, *IEEE Trans. Comput.* **34**, 663–666 (1985).
4. S. L. Chou, J. C. Lin and W. H. Tsai, Fold principal axis—a new tool for defining the orientations of rotationally symmetric shapes, *Pattern Recognition Lett.* **12**, 109–115 (1991).
5. W. H. Tsai and S. L. Chou, Detection of generalized principal axes in rotationally symmetric shapes, *Pattern Recognition* **24**, 95–104 (1991).
6. J. C. Lin, S. L. Chou and W. H. Tsai, Detection of rotational symmetric shape orientations by fold-invariant shape-specific points, *Pattern Recognition* **25**, 473–482 (1992).
7. J. J. Leou and W. H. Tsai, Automatic rotational symmetry determination for shape analysis, *Pattern Recognition* **20**, 571–582 (1987).
8. A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Vol. II, pp. 289–290. Academic Press, New York (1982).
9. J. C. Lin and W. H. Tsai, Feature-preserving clustering of 2D data for two-class problems using analytical formulas: an automatic and fast approach, *IEEE Trans. Pattern Anal. Mach. Intell.* **16**(5) (1994).
10. A. Mitiche and J. K. Aggarwal, Contour registration by shape-specific points for shape matching, *Computer Vis. Graphics Image Process.* **22**, 396–408 (1983).
11. J. C. Lin, W. H. Tsai and J. A. Chen, Detecting number of folds by a simple mathematical property, *Pattern Recognition Lett.* **15**, 1081–1088 (1994).

About the Author—JA-CHEN LIN was born in 1955 in Taiwan, Republic of China. He received his B.S. degree in computer science in 1977 and M.S. degree in applied mathematics in 1979, both from National Chiao Tung University, Taiwan. In 1988 he received his Ph.D. Degree in mathematics from Purdue University, U.S.A. In 1981–1982, he was an instructor at National Chiao Tung University. From 1984 to 1988, he was a graduate instructor at Purdue University. He joined the Computer Vision Laboratory of the Department of Computer and Information Science at National Chiao Tung University in August 1988, and is currently an associate professor there. His recent research interests include pattern recognition, image processing, and parallel computing. Dr Lin is a member of the Phi-Tau-Phi Scholastic Honor Society.