Shape from Contour Using Symmetries

Shiu-Yin Kelvin Yuen
Cognitive Studies Programme,
University of Sussex,
Brighton BN1 9QN
UK
Janet : kelviny@cogs.susx.ac.uk

July 14, 1989

Kanade [1] proposed a heuristic for interpreting shape from contour. To apply his heuristic, skewed symmetries have to be found. In this paper, an algorithm for finding skewed symmetries in a planar point set is proposed. The method requires a simple rotation and midpoint finding, followed by a Hough transform. A variant of the Hough transform is reported. It is based on interpreting it as a rotation followed by a projection. This variant requires only a one dimensional accumulator array. An implementation is outlined. We then show (theoretically) that the skewed symmetry finding algorithm is as robust as the standard Hough transform (Hough transform for finding lines).

Shape from contour is the interpretation of a single line drawing as the projection of a three dimensional entity. Kanade [1] proposed the following heuristic for shape from contour :

"A skewed symmetry depicts a real symmetry viewed from some (unknown) view direction."

A "skewed symmetry" is a planar point pattern such that iff (x,y) exists, (-x, y) exists. The x axis is called the "skewed transverse axis" t_k; while the y axis is called the "skewed symmetric axis" s_k. If t_k and s_k are orthogonal, then the skewed symmetry degenerates into a reflectional (real) symmetry. Fig 1 are examples.

Much work on detecting reflectional symmetries in a given polygon or a simple closed curve has been reported [2 - 6]. The above results are not directly applicable to the more general notion of point sets. Highnam [7] reported a method for finding reflectional (and rotational) symmetries given a symmetric point set. Marola [8, 9] reported a method for detecting reflectional symmetries in a gray level image assuming that the image is symmetric in a neighborhood. The most serious deficiency of the above results is their presupposition that the input is symmetrical.

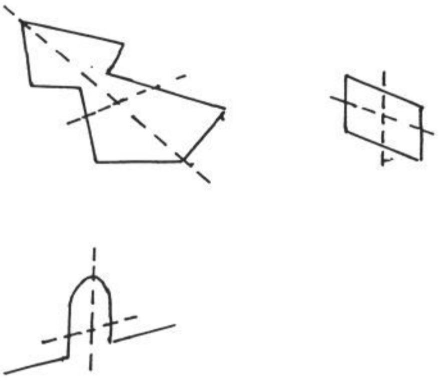

Figure 1: Some skewed symmetric figures taken from [1].

In this paper, we wish to treat the more general problem of finding skewed symmetry – which is more general than reflectional symmetry – in (planar) point sets without requiring that our input is symmetrical.

The only related research we know of on finding skewed symmetry is Friedberg [10], who assumed that the input is skewed symmetric. This allows him to use moments to formulate the problem.

Ponce [11] derived a necessary condition for two points to be skewed reflective to each other. A method is also sketched for using this condition to detect reflectional symmetry. However, the condition involves the curvatures at the two points and hence is limited to smooth curves. Our algorithm is applicable to general point sets.

Levitt [12] proposed an elegant method to detect reflectional symmetry in a point set, without requiring a priori that the point set be symmetric. Suppose there are n points. For each of the ${}_nC_2$ pairs, the midpoint is found and assigned a direction perpendicular to the line joining the pair. To this is added the n points with no direction assigned. A Hough transform is then used to find straight reflectional symmetry axes.

We may extend Levitt's method to detect skewed symmetry in the following way. Suppose P_1, P_2 is a pair of points. Then instead of storing the angle of the perpen-

AVC 1989 doi:10.5244/C.3.49

dicular, we may store the angle β of line $\overline{P_1 P_2}$ with some reference axis on a third dimension of the Hough space. High counts on the Hough space correspond to a skewed symmetric axis s_k where the skewed transverse axes are at the same angle with respect to s_k (Fig 2).

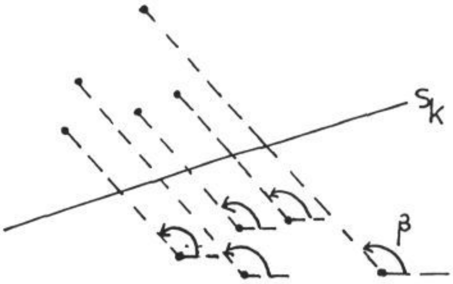

Figure 2: The line joining mapping points are all in the same direction β. Moreover, the locus of their midpoints is the skewed symmetric axis s_k. If s_k is straight, we may accumulate the midpoints together with their β in a three dimensional Hough space to detect skewed symmetric axes.

In addition to a three dimensional Hough space, this approach has a serious problem. Consider Fig 3. By successively covering segment $\overline{ab}$, $\overline{bc}$, and $\overline{cd}$, we obtain the skewed symmetric axes s_{ka} to s_{kd} ! Since the Hough transform does not encode connectivity, this suggests that the Hough space will have many confusing high points. Hence this method must be abandoned. Below, we describe a new method which is cleaner and furthermore requires much less storage.

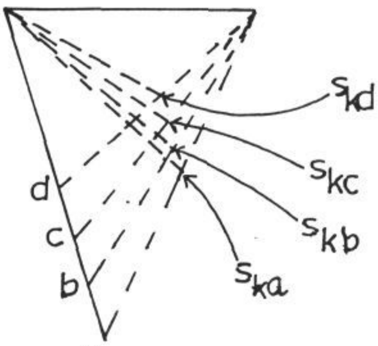

Figure 3: Four skewed symmetric axes s_{ka} to s_{kd} may be produced by successively covering line segments $\overline{ab}$, $\overline{bc}$ and $\overline{cd}$.

AN ALGORITHM FOR FINDING SKEWED SYMMETRY

A skewed symmetry is a parallel projection of a reflectional symmetry. Suppose p', q' are two three dimensional points and p' reflects onto q' (and vice versa). Then we shall say that they are *mapping points* or a *mapping pair.*

Suppose r', s' are another mapping pair. Then clearly $\overline{p'q'} // \overline{r's'}$. Now since a parallel projection is an affine transformation, parallelism is preserved. Hence upon projection $\overline{pq} // \overline{rs}$, where p is the projected point of p' etc. We shall call $\angle\overline{pq}$ the *mapping direction.*

Let the mapping direction make an angle α with the positive x axis.

Algorithm (Skewed symmetry finding)

Input : a two dimensional point set.

```
1.  $\alpha \leftarrow 0$;
    ;;; the initial mapping direction is parallel
        to the x axis

2.  clear the midpoint_array;
    rotate the point set by $-\alpha$ about the origin;
    ;;; after the rotation, the mapping direction is parallel to the x axis

3.  for all lines L parallel to the x axis do
      if two points p, q lie on the L then
        record the midpoint of p and q on midpoint_array;
      endif;
    endfor;

4. use a Hough transform to find straight lines in midpoint_array;

5. increment $\alpha$;

6. if $\alpha > \pi$, then exit else goto 2.
```

A VARIANT OF THE HOUGH TRANSFORM FOR STRAIGHT LINE FINDING

The Hough transform is an one-to-many mapping of a point into a curve in the transformed space. Duda and Hart [13] defined the following equation for a straight line, which serves as a mapping :

$$\rho = x_0 cos\theta + y_0 sin\theta \tag{1}$$

where ρ is the distance of the line from the origin and θ is the angle of the perpendicular (Fig 4).

For each point (x_0, y_0), they increment all (ρ, θ) satisfying (1) in a $\rho - \theta$ accumulator array. Detecting straight lines is converted to the problem of detecting *high counts* in the accumulator array. This transform is known as the Hough transform. It is ingenious because choosing

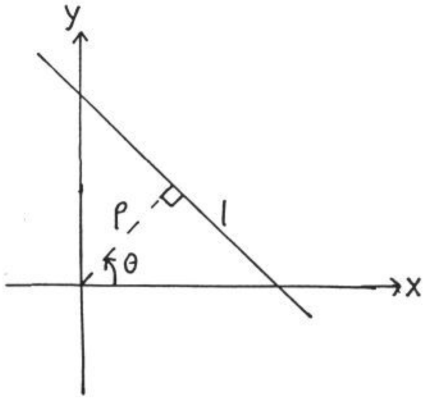

Figure 4: A parameterization of line l used by Duda and Hart [13].

an appropiate quantization of the accumulator array will permit multiple lines to pass through two nearby image points. The quantization errors of the image points is combatted by the quantization errors of the accumulator array !

If ρ is quantized into m_1 cells and θ into m_2 cells, then we have a $m_1 \times m_2$ accumulator array. Here we propose a modification of the Hough transform. Recall Fig 4. Suppose the x-y plane is rotated by $-\theta$ about the origin. Then the line l would become a line parallel to the y axis. If l is then projected to the (original) x axis, the detection of l is equivalent to the detection of high counts *along the x axis*. Actually the distance from the origin on the x axis is equal to ρ. Hence the x axis may be quantized in the same way as the ρ axis of Duda and Hart.

The rotation followed by projection is described by

$$[\ x_0'\ ] = [\ 1 \quad 0\ ] \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

$$[\ x_0'\ ] = [\ x_0 cos\theta + y_0 sin\theta\ ] \tag{2}$$

where $(x_0', 0)$ is the transformed point. This is exactly (1) when x_0' is replaced by ρ.

By varying θ and repeating the above, the variant may find all lines in the image.

Because (1) and (2) are exactly the same, the time complexity of the two transforms are strictly equal. However, if done serially, the variant need only have a one dimensional accumulator array of m_1 cells. Hence the space complexity of the variant is a factor of $1/m_2$. Moreover, the variant is more suitable for highly parallel implementation because it divides the task into m_2 independent subtasks.

The Hough transform (and the variant) is no more than a method for finding *collinearity*. A line segment has both collinearity and *connectivity*. That is, points on the segments are connected. O' Gorman and Clowes [14] solved this problem by attaching the *original* (x_0, y_0) position to the accumulator cells. Exactly the same can be done to the variant to detect collinearity.

Note also that the skewed symmetry algorithm involves (1) a rotation, (2) midpoint finding, (3) a rotation and (4) a projection. (2) and (4) are trivial. Efficient algorithm [15] exists for rotation. Moreover, we show in the next section that (1) does not involve a real rotation.

AN IMPLEMENTATION OUTLINE

We sketch an implementation of the skewed symmetry finding algorithm in this section. The image array is a finite grid which is the result of quantizing the image points. In effect, any image point p_r that lies within square s_p of the grid is quantized to its center p. Suppose we have two such quantized points p and q. Consider the line formed by the true points, i.e. $\overline{p_r q_r}$. Then the orientation of this line is limited by a pair of corner points of squares s_p and s_q. Let call this range of orientations by R_{pq} (Fig 5).

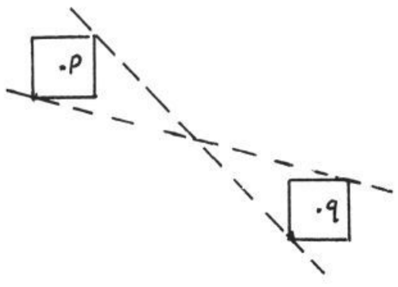

Figure 5: Limits of orientations of $\overline{p_r q_r}$

Given p and q, we may find its midpoint m_{pq}. Both p and q are integers. Hence m_{pq} may either be an integer or an integer with a fractional part of 0.5. We shall store m_{pq} up to an accuracy of 0.5. (One way to implement this is to map m_{pq} to an array twice as fine as the image array).

Since

$$p_r = p \pm \Delta$$
$$q_r = q \pm \Delta \tag{3}$$

where Δ is the quantization error. The *true* midpoint mr_{pq} is

$$mr_{pq} = (p_r + q_r)/2 = (p + q)/2 \pm \Delta = m_{pq} \pm \Delta \tag{4}$$

In other words, the quantization error of m_{pq} is *the* *same* as the quantization error of the image array. Let us then call the square around m_{pq} by s_m.

The physical meaning is as follows : Suppose we wish to rotate p and q by $-\alpha$. Then if and only if $\alpha \in R_{pq}$, there exists some p_r and q_r such that $\angle\overline{p_r q_r} = \alpha$. Moreover, there exists some midpoint $m_r = (p_r + q_r)/2$ which lies in s_m. Hence if we are to rotate by $-\alpha$, then the midpoint m_{pq} would be the midpoint subjected to the same quantization error of the image array.

Suppose we have n points in the image array. For each of the ${}_nC_2$ pairs p, q, we compute m_{pq} (and store it at twice the accuracy) and attach R_{pq} to it. The result is stored in a all_midpoint array [1] (which is twice as fine as the image array). Now suppose we wish to make a rotation of the image array by $-\alpha$. We need only suppress all midpoints for which $\alpha \notin R_{pq}$. Thus no actual rotation is involved and as alluded to above, no accuracy is lost. We may then use a Hough transform to the midpoint array.

Since the midpoint array has the same quantization error as the image array (it is as good as if the midpoint array is acquired *directly* from the real image), the symmetry finding algorithm is thus *as robust as* the standard Hough transform (Hough transform for finding lines).

ACKNOWLEDGEMENT

First and foremost I wish to thank for the supervision of Dr. David Hogg and for many sharp and enlighening comments. I am indebted to the following people for helpful discussions : Prof. Harry Barrow, Dr. David Hogg, Dr. Hlavac Vaclav, Dr David Young. I wish to thank Vaclav especially for many stimulating discussions on the robustness of the skewed symmetry algorithm. I am very grateful to a referee who raised the question about the robustness of the algorithm.

REFERENCES

1. **Kanade, T.** "Recovery of the three-dimensional shape of an object from a single view" *Artificial Intelligence* Vol. 17(1981) pp409-460.

2. **Davis, L.S.** "Understanding shape : II symmetry" *IEEE Trans. Systems, Man and Cybernetics* Vol. 7(1977) pp204-212.

3. **Chaudhuri, B.B., Majumder, D.D.** "Recognition and fuzzy description of sides and symmetries of figures by computer" *Int. J. Systems Sci.* Vol. 11(12) (1980) pp1435-1445.

4. **Atallah, M.J.** "On symmetry detection" *IEEE Trans. on Computers* Vol. C-34(7) (1985) pp663-666.

5. **Biswas, S.N., Majumder, D.D.** "Region symmetry analysis through cantor number strings" Proceedings International Conference on Pattern Recognition (1986) pp427-429.

6. **Radig, B., Schlieder, C.** "RS-automorphisms and symmetrical objects" Proceedings International Conference on Pattern Recognition (1984) pp1134-1140.

7. **Highnam, P.T.** "Optimal algorithms for finding the symmetries of a planar point set" *Inf. Process Lett.* Vol. 22(1986) pp219-222.

8 . **Marola, G.** "On the detection of the axes of symmetry of symmetric and almost symmetric planar images" *IEEE Trans. Pattern Anal. Mach. Intell.* Vol. 11(1) (1989) pp104-108.

9. **Marola, G.** "Using symmetry for detecting and locating objects in a picture" *Comput. Vision Graph. Image Process.* Vol. 46(2) (1989) pp179-195.

10. **Friedberg, S.A.** "Finding axes of skewed symmetry" *Comput. Vision Graph. Image Process.* Vol. 34(1986) pp138-155.

11. **Ponce, J.** "Ribbons, symmetries, and skewed symmetries" Proc. Image Understanding Workshop (1988) pp1074-1079.

12. **Levitt, T.S.** "Domain independent object description and decomposition" Proceedings AAAI-84 (1984) ppsss207-211.

13. **Duda, R.O., Hart, P.E.** "Use of the Hough transformation to detect lines and curves in pictures" *Commun. ACM* Vol. 15(1) (1972) pp11-15.

14. **O'Gorman, F., Clowes, M.B.** "Finding picture edges through collinearity of feature points" Proceedings IJCAI-73 (1973) pp543-555.

15. **Sanz, J. L. C., Dinstein, I.** "Projection-based feature extraction for computer vision : algorithms in pipeline architectures" *IEEE Trans. Pattern Anal. Mach. Intell.* Vol. 9(1) (1987)pp160-168.

[1] A more efficient implementation (suggested by Dr. David Hogg) is to use a list sorted by, say, the first entry of the orientation range