1998 International Computer Symposium
Workshop on Image Processing and Character Recognition
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

MULTI-SCALE CONTOUR SEGMENTATION AND OCCLUDED OBJECT RECOGNITION

Xiangjian He, Tom Hintz and Ury Szewcow

School of Computing Sciences
University of Technology, Sydney
PO Box 123, Broadway 2007
Australia

Email:{sean, hintz, ury}@socs.uts.edu.au

ABSTRACT

Contour segmentation plays an important role in occluded object recognition. In this paper, we propose a segmentation method which is robust with respect to noise. Our approach is based on the curvature zero-crossing points extracted from outermost contours of multi-scale images. By using the segmentation scheme, a matching procedure for object recognition with occlusion is presented. This requires an affine integral invariant representation of contours which have an *arclength* as parameter.

Keywords. Scale-space Theory, Object Recognition, Contour Extraction.

1 INTRODUCTION

Object recognition has many applications in industry, defense and medical science. A fundamental problem of object recognition is recognising occluded objects within a given scene. Feature-based methods for object recognition have been the goal of much recent research [1-3]. Finding efficient invariant features [4] of contours extracted from images has been a useful solution to this problem. A system for object recognition with occlusion can be found in [5]. A segmentation of the extracted contour has proved important in occluded object recognition [6]. Segmentation algorithm must be considered to be robust with respect to noise and be invariant under general affine transformation. There has been considerable research carried out on contour segmentation. Kass et al. [7] focused on polygon approximation of contour data. Since the locations of polygon vertices are arbitrary, their technique is sensitive to the noise and the affine transformation of the contour. Rattarangsi et al. [8] and Rosin [9] extracted local maxima from contour smoothed using a fixed Gaussian scale. These points were then used for the segmentation. Though this approach is less sensitive to noise than the polygon approximation method, it can suffer from loss of structure due to over-smoothing since the extracted points are always from a single scale. To overcome this problem, Mokhtarian [6] suggested a multi-scale segmentation method based on the curvature scale space. He segmented contours using the curvature zero-crossing points of multi-scale contours. However, the paper neglected a basic fact that the Gaussian evolved version of the initial contour is not the contour of the Gaussian evolved image of the input image. This means the extracted zero-crossing points may not be on any contours of the Gaussian evolved images. Using these points for the segmentation may result in inaccurate object matching results from their procedure.

In this paper, we consider the contours of multi-scale image representations. For each candidate image, we first use the edge focusing technique [2] to find the appropriate Gaussian scale for the extraction of the image contour. This is based on the Spiral architecture [10]. The curvature zero-crossing points will then be extracted from the contour of the Gaussian evolved image by using Mokhtarian's method [6]. This leads to a new contour segmentation. Besides this, a matching procedure will be proposed for occluded object recognition.

Our segmentation method avoids the creation of pseudo zero-crossing points of curvature and guarantees that the contour is not over-smoothed. Furthermore, our approach is robust in the presence of noise.

2 MULTI-SCALE EDGE DETECTION

An image may be considered as the collection of pixels (picture elements). These elements correspond to the position of the photo receiving cells of the image capturing device. In the case of the human eye, these elements would represent the relative position of the rods and cones on the retina. The geometric arrangement of cones on the primate's retina can be described in terms of a hexagonal grid. This leads to the consideration of an image as the collection of hexagonal cells (in contrast with the traditional collection of rectangular cells). The new representation of an image is the foundation of Spiral Architecture. The importance of the Spiral arrangement, apart from its intriguingly beautiful geometry, is that it possesses powerful computational advantages for computer vision [11].

Let $f : \Re^2 \to \Re$ be a signal of an image. The scale-space representation $L : \Re^2 \times [0, \infty) \to \Re$ is defined such that the representation at 'zero scale' is equal to the original signal, i.e.,

$$L(\cdot; 0) = f(\cdot),$$

and the representation at 'coarser scales' is the convolution

$$L(\cdot; t) = g(\cdot; t) * f(\cdot), \tag{1}$$

where $g : \Re^2 \times (0, \infty) \to \Re$ is the Gaussian kernel [12].

Scale-space representation is used to suppress and remove unnecessary and distorting details e.g. noise so that later stage processing tasks can be simplified.

A natural way to define edges from a continuous grey-level image L is as the set of points for which the gradient magnitude assumes a maximum in the gradient direction [12]. Hence, if $P = (x, y) \in \Re^2$ is an edge point, then [12], at this point,

$$\begin{aligned} & L_x^2 L_{xx} + 2L_x L_y L_{xy} + L_y^2 L_{yy} = 0, \\ & L_x^3 L_{xxx} + 3L_x^2 L_y L_{xxy} \\ + \; & 3L_x L_y^2 L_{xyy} + L_y^3 L_{yyy} < 0. \end{aligned}$$

Given discrete data, note that the six neighbouring hexagons of the hexagon at (x, y) have Cartesian coordinates $(x, y-1)$, $(x - \sqrt{3}/2, y - 1/2)$, $(x - \sqrt{3}/2, y + 1/2)$, $(x, y+1)$, $(x + \sqrt{3}/2, y + 1/2)$ and $(x + \sqrt{3}/2, y - 1/2)$ (Fig. 1).

The derivative operators can then be obtained in finite difference form:

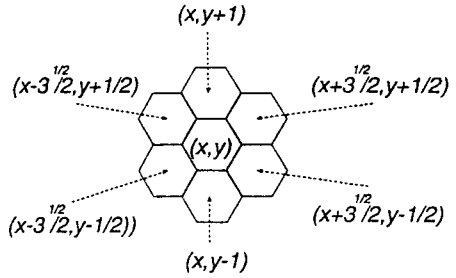


Figure 1: Ordinary coordinates of a cluster of 7 hexagons.

$$\begin{aligned} & L_x(x, y; t) \\ = \; & \frac{1}{\sqrt{3}}[L(x + \sqrt{3}/2, y + 1/2; t) \\ & + L(x + \sqrt{3}/2, y - 1/2; t)] \\ & - \frac{1}{\sqrt{3}}[L(x - \sqrt{3}/2, y + 1/2; t) \\ & + L(x - \sqrt{3}/2, y - 1/2; t)] \end{aligned}$$

and

$$\begin{aligned} & L_y(x, y; t) \\ = \; & \frac{1}{2}[L(x + \sqrt{3}/2, y + 1/2; t) \\ & + L(x - \sqrt{3}/2, y + 1/2; t) + L(x, y + 1; t)] \\ & - \frac{1}{2}[L(x + \sqrt{3}/2, y - 1/2; t) \\ & + L(x - \sqrt{3}/2, y - 1/2; t) + L(x, y - 1; t)]. \end{aligned}$$

The second and third order derivatives can be obtained respectively from the first and second order derivatives in the same way. An edge detection algorithm using the edge focusing technique [13] is as follows:

1. An edge map is defined as a binary image represented by 0's and 1's with a Gaussian resolution parameter t defined earlier in this paper, denoted by $E(x, y; t)$. $E(x, y; t) = 1$ if the pixel (x, y) is an edge point, otherwise $E(x, y; t) = 0$.

2. Create an initial coarse-level edge map $E(x, y; t_0)$, using an edge detector based on the Gaussian bluring defined in (1) and the differential representation decribed above in this section.

3. For $i = 1, 2, \cdots$, create the ith-level edge map $E(x, y; t_i)$, where $t_i = t_{i-1} - \Delta t$. The resolution parameter, t is decreased by Δt at each blurring step. Note that the Gaussian operator only acts on the edge points, i.e., $\{(x, y) | E(x, y; t_{i-1}) = 1\}$ detected in the

previous iteration, and their neighbouring areas.

4. The edge detection iteration continues until the blurring scale t_i is too small to blur the image further.

Remark. *The edge map obtained by a larger Gaussian scale has less noise and the edge map obtained by using smaller scale has more precise edge locations. Step 3 above compares the two edge maps with two different scales to obtain a new edge map with low noise. The new locations of the edges are based on the information in the edge map obtained with the smaller scale. The final edge map obtained in this way has the edge points in the right locations and is less sensitive to the noise. Furthermore, a Gaussian scale is finally determined.*

3 CONTOUR SEGMENTATION

In the previous section, we have presented a method for image edge detection. The detected edge points are used to extract the outermost contour/s of the image. In this section, we show a way for segmenting the extracted contour/s based on the curvature representation of the contour.
Let Γ be the contour represented by

$$\Gamma(u) = (x(u), y(u)),$$

where u is the *arc length* parameter, the value of which can easily be calculated numerically at each point on the contour. The curvature k on Γ is given by

$$k(u) = \frac{x_u(u)y_{uu}(u) - x_{uu}(u)y_u(u)}{(x_u^2(u) + y_u^2(u))^{\frac{3}{2}}}, \qquad (2)$$

where x_u and y_u are the 1st order derivatives of x and y respectively with respect to u, and x_{uu} and y_{uu} are the 2nd order derivatives of x and y respectively with respect to u.
The curvature zero-crossing points are used as the feature points to segment the contour since they are invariant to general affine transformation, e.g., rotation, scaling or/and translation. A segment of the contour is defined as a segment on the contour delimited by two consecutive curvature zero-crossing points refered to as its *endpoints*.
The segmentation procedure can now be described as follows.

1. Find and locate the curvature zero-crossing points of the contour using equation (2) above.
2. Create all segments of the contour by the definition of segment above.

In the next section, we will present a procedure for recognizing the occluded object based on the segments constructed in the current section.

4 OCCLUDED OBJECT RECOGNITION

Due to occlusion, the recognition algorithm employed is a local one and includes several stages described in the following subsections in details.

4.1 Affine invariant representation of segment

Let C be a segment of the extracted contour represented by $(x(t), y(t))$ with parameter t (not necessarily the arc length parameter). It is well known that the arc length parameter defined by

$$u = \int_{t_0}^{t} (\dot{x}\ddot{y} - \ddot{x}\dot{y})^{\frac{1}{3}} dt \qquad (3)$$

is linearly transformed under an affine transformation [4]. In the following, we will define a parameter which will be used to construct an affine invariant. We start with the expression of general affine transformation. A 2-dimensional affine transformation is a linear transformation as follows

$$\begin{aligned} \tilde{x} &= a_{11}x + a_{12}y + b_1 \\ \tilde{y} &= a_{21}x + a_{22}y + b_2. \end{aligned} \qquad (4)$$

Suppose $\tilde{C}$ is the curve derived from C by the affine transformation (4). It is easy to see that (4) transforms (3) to

$$|A|^{\frac{1}{3}} \int_{t_0}^{t} (\dot{\tilde{x}}\ddot{\tilde{y}} - \ddot{\tilde{y}}\dot{\tilde{x}})^{\frac{1}{3}} dt,$$

where A is the cofficient matrix of (4) and $|A|$ denotes the determinant of A. This means that parameter u is transformed to $|A|^{\frac{1}{3}}\tilde{u}$ by (4), where $\tilde{u}$ is the *arclength* parameter of $\tilde{C}$. Define a new parameter s as follws

$$s = \frac{u}{\int_C \ddot{x}\dot{y} - \ddot{y}\dot{x})^{\frac{1}{3}} dt} \qquad (5)$$

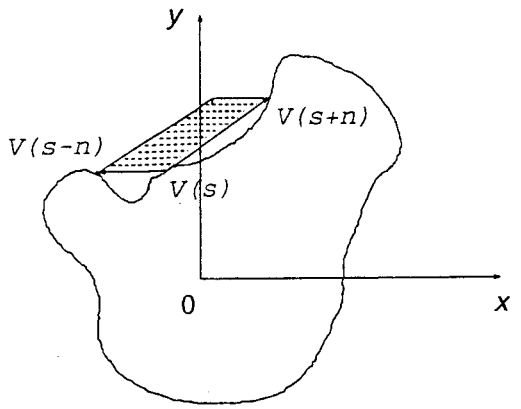


Figure 2: $I(s)$ is the area made by $V(s+n)-V(s)$ and $V(s-n)-V(s)$.

where int_C denotes the line integral along C. The parameter s defined in (5) is called *normalized arclength* parameter. s is invariant under the affine transformations (4), i.e., (4) transforms s to

$$\tilde{s} = \frac{\tilde{u}}{\int_{\tilde{C}} \dot{\tilde{x}}\ddot{\tilde{y}} - \ddot{\tilde{y}}\dot{\tilde{x}})^{\frac{1}{3}} dt}$$

We now derive an invariant representation of contour segment under the affine transformations defined in (4).
Let the contour segment C be represented by

$$V(s) = \begin{pmatrix} x(s) \\ y(s) \end{pmatrix}$$

for $s \in [0,1]$ and $V_s(s)$ denote the derivative of $V(s)$ with respect to s. Define

$$\begin{aligned} I(s) &= [V(s+n)-V(s-n), V(s+n)-V(s)] \\ &= [V(s+n)-V(s), V(s+n)-V(s)] \\ &\quad +[V(s)-V(s-n), V(s+n)-V(s)] \\ &= [V(s+n)-V(s), V(s-n)-V(s)], \end{aligned} \quad (6)$$

which is the area made by two vectors, $V(s+n)-V(s)$ and $V(s-n)-V(s)$, as shown in Fig. 2. In (6), $V(s+n)$ ($V(s-n)$) represents the nth point (or pixel) in the clockwise (anticlockwise) direction counted from $V(s)$ along the contour.

We now prove that $I(s)$ shown in (6) is a relative invariant under the affine transformation shown in (4). Assume that $\tilde{V}(\tilde{s})$ is the derivation of $V(s)$ by (4) for any $s \in [0,1]$ and $\tilde{I}(\tilde{s})$ is the corresponding transformation of $I(s)$. Note that, s is an absolute invariant under (4). So, $\tilde{V}(\tilde{s}) = \tilde{V}(s)$ and hence

$$\begin{aligned} &\tilde{I}(\tilde{s}) \\ &= [\tilde{V}(\tilde{s}+n)-\tilde{V}(\tilde{s}), \tilde{V}(\tilde{s}-n)-\tilde{V}(\tilde{s})] \\ &= [A(V(s+n)-V(s)), A(V(s-n)-V(s))] \\ &= |A|[V(s+n)-V(s), V(s-n)-V(s)]. \end{aligned} \quad (7)$$

This means that $I(s)$ is an affine relative invariant representation, i.e., it relates to its image by a multiplication constant $|A|$. The multiplication constant can be removed if the representation is expressed in ratio form as

$$M(s) = \frac{I(s)}{I(v)}, \quad (8)$$

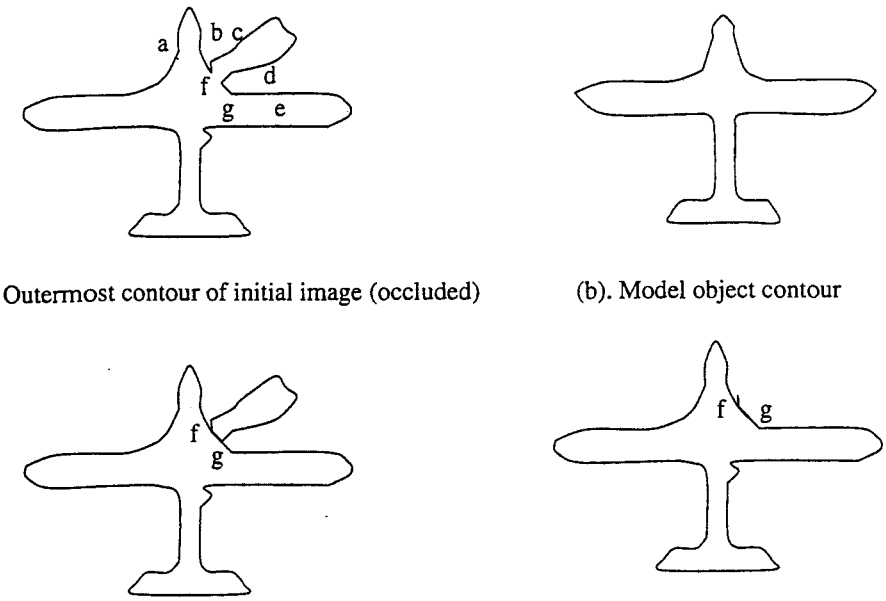
where the position v is the location of the largest absolute magnitude of the relatively invariant representation.

4.2 Index table

In order to speed up the search of model segments, an index table is employed. After segmentation, the average $M(s)$ of each model contour segment is computed and used as an entry of the index table.
Once the segmentation of the image contour is completed, the average $M(s)$ of each image contour segment is computed. The average $M(s)$ now serves as an index into the model contour segment index table to recover a most likely model contour segment, i.e, to get the model contour segment which has the closest average $M(s)$ to that of the image contour segment. A candidate is then generated corresponding to the match of each image contour segment and a model contour segment recovered from the index table. A candidate is defined and stored as a data dtructure consisting of an image segment, a matched model segment [5] and the difference between the average $M(s)$'s of the two segments. The difference is denoted by *segment-diff*. We disqualify any candidate which has a big *segment-diff*.

4.3 Candidate merging

Due to occlusion, it is possible that some image contour segments are not fully on the boundary of the object of interest. For example, in Fig. 3(a), the segment from point b to c is not all on the boundary of the airplane (the object of interest) and the segment from c and d is totally not on the boundary. Hence, it is necessary to merge the neighbouring image contour segments if they are



(a). Outermost contour of initial image (occluded)

(b). Model object contour

(c). Occluded object contour formed by grouping segments in (a)

(d). Extracted contour of interest object

Figure 3: Image and model contours.

both on the boundary or disqualify the segment which is not completely on the boundary. Two candidates c_1 and c_2 will be merged if they satisfy the following criteria [5]:

1. c_1 and c_2 must be differnt candidates and not be previously merged.
2. c_1 and c_2 must correspond to the same model.
3. The corresponding image contour segments of c_1 and c_2 must be neighbouring.
4. The corresponding model contour segments of c_1 and c_2 must be neighbouring.
5. A new candidate has a smaller *segment-diff* than those of c_1 and c_2. The three components of this new candidate are the image coutour segment merged from the corresponding image segments of c_1 and c_2, the model contour segment merged from the corresponding model segments of c_1 and c_2, and the *segment-diff* between the new image segment and the new model segment .

When two candidates can not be merged, the segment should be extended. We present this idea in the next subsection.

4.4 Candidate extension

Again due to occlusion, it is possible that only part of an image contour segment belongs to the boundary of the interest object. For example, in the Fig. 3(a), the segment delimited by curvature zero-crossing points b and c has the part from point b to the point f belonging to the interest object, an airplane, and the part from f to b not belonging to it. Hence, we need to include the part belonging to the boundary into one of the image contour segments. In other worlds, it is necessary to extend an image segment to contain the part. As a result, it is important to find the object boundary intersection points. In general [5], the intersection point of two object boundaries in the input image does not coincide with an endpoint of an image segment. Therefore, in order to find the exact location of such intersection points, it is necessary to gradually extend the image and model contour segments until the smallest *segment-diff* is reached. If an image segment can not be merged with its left or right neighbouring segment, the extension will be carried out at the left or right *endpoint*.

Since object intersection points are normally a subset of the curvature extrema on the image contour, we may simply extend an image segment

to the next curvature extremum when we need to do so. The following procedure [5] is applied at each *endpoint* for the segment extension.

1. Extend the image segment (required for extension) to the next curvature extremum.
2. The corresponding model contour segment is extended by the same length.
3. Form a new candidate using the extended image segment and model segment. This implies that a new segment-diff is computed.

Extension stops when next new candidate has a larger segment-diff than the current one.
In figure 3(a), the initial image segment from a to b will be extended from the right *endpoint* to f.

In order to extract the full contour of the interest object, we move to the next stage.

4.5 Candidate grouping and joining

This step is to group compatible but disjoint candidates. We say candidates are *compatible* if they are corresponding to the same model.
In this step, we simply connect all corresponding image contour segments (after previous two stages) of compatible candidates one to another in a clockwise direction they are picked up from the image contour using straight lines. Then a contour is formed by the connected segments. Compute the *segment-diff* between this new contour and the model contour. Denote this *segment-diff* by $Sd(n)$ which is depending on the choice of n.

Remark. *If there is only one candidate in the compatible class, i.e., there is a single candidate which is only compatible to itself, and the corresponding image segment is not a closed curve, then the* **grouping** *simply means connecting the endpoints of the segment by a straight line.*

Fig. 3(c) shows a straight line connection of emtrema f and g and figure 3(d) is the final extracted contour of an airplane, the interest object included inthe original image.

Having the above stages ready, we are now able to suggest an algorithm for occluded object recognition in the next section.

5 RECOGNITION ALGORITHM

The following is a pseudo-code description of our approach.

1. Acquire model contours as described in section 2.
2. Extract model contour segments as decribed in section 3.
3. Compute average $M(s)$ for each model segment as described in subsection 4.1.
4. Make an index table for each model as described in subsection 4.2.
5. Process input image to recover image contour as described in section 2.
6. Segment the image contour using the method described in section 3.
7. Compute the average $M(s)$ for each image segment as described in subsection 4.1.
8. Generate initial candidates while applying the object indexing scheme described in subsection 4.2.
9. Merge candidates as described in subsection 4.3.
10. Extend the new candidates as described in subsection 4.4.
11. Group and joint the extended candidates as described in subsection 4.5.
12. Select the model with the smallest *segment-diff* computed at previous step.

In practice, in order to make the matching procedure be more accurate, we may take a set of values of parameter n noting that n is an arbitrary positive integer. For each value of n, we repeat above steps 3, 7 through 12 and find a model object which matches best the unknown object. The model object which is selected most frequently by the step 12 above during the tests is considered to be 'identical' with the unknown occluded object.

6 CONCLUSION

In this paper, we apply the Spiral Architecture and the multi-scale theory for contour extraction. Then we segment contours using curvature zero-crossing points. A new recognition algorithm for occluded object is presented.

References

[1] D. L. Borges and R. B. Fisher, "Class-based recognition of 3D objects represented by columetric primitives", *Image and Vision Computing*, Vol. 15, pp.655-664, 1997.

[2] X. He, T. Hintz and P. Sheridan"Object recognition with Spiral Architecture", *Proc. 3rd Australasian Conf. Parallel and Real-time Systems*, pp.230-235, 1996.

[3] C. A. Rothwell, A. Zisserman, D. A. Forsyth and J. L. Mundy, "Planar object recognition using projective shape representation", *International Journal of Computer Vision*, Vol. 16, pp.57-99, 1995.

[4] J. Sato and R. Cipolla, "Affine integral invariants for extracting symmetry axes", *Image and Vision Computing*, Vol. 15, pp.627-635, 1997.

[5] F. Mokhtarian, "Silhoutte-based occluded object recognition through curvature scale space", *Machine Vision and Applications*, Springer-Verlag Vol.10, pp.87-97, 1997.

[6] F. Mokhtarian, "Multi-scale contour segmentation", *Scale-Space Theory in Computer Vision*, Lecture Notes in Computer Science; Vol.1252, Springer-Verlag, Berlin, pp.296-307, 1997.

[7] M. Kass, A. Witkin and D. Terzopoulos, "Snakes: active contour models", *Proc. International Conference on Computer Vision*, pp.259-268, 1987.

[8] A. Rattarian and R. T. Chin, "Scale-based detection of conners of planar curves", *IEEE Trans Pattern Analysis and Machine Intelligence*, Vol.14(4), pp.430-449, 1992.

[9] P. L. Rosin, "Representing curves at their natural scales", *Pattern Recognition*, Vol.25, pp.1315-1325, 1992.

[10] P. Sheridan, *Sprial Architecture for Machine Vision*, Ph.D Thesis, University of Technology, Sydney, Australia, 1996.

[11] P. Sheridan, T. Hintz and D. Alexander, "Geometric invariance on a hexagonal grid", submited to Journal of Image and VIsion Computing.

[12] T. Lindeberg, *Scape-Space Theory in Computer Vision*, Kluwer Academic Publishers, London, 1994.

[13] X. Zhang and H. Deng, "Distributed image edge detection methods and performance", *Proc. 6th IEEE Symposium on Parallel and Distributed Processing*, IEEE CS Press, October 1994.