METHODS TO ESTIMATE AREAS AND PERIMETERS OF BLOB-LIKE OBJECTS: A COMPARISON

Luren Yang, Fritz Albregtsen, Tor Lønnestad and Per Grøttum
Dept. of Informatics, Univ. of Oslo, P.O. Box 1080, Blindern, N-0316 Oslo, Norway

ABSTRACT

The area and the perimeter of a planar object are two useful features to describe the shape of the object, and again the motion of it. This paper deals with the estimation of the two features from a discrete binary image. The area can often be accurately estimated by counting the number of pixels inside the object. However, the estimation of the perimeter is a problem, since many possible contours, all having different lengths, correspond to a specific discrete realization. Thus, to develop a practical length estimator, some reasonable assumption about the original contour should be made. We assume that the boundary of a blob-like object consists of chains of circular arcs, and therefore evaluate the precision of several area and length estimators applied to circles. The problem of efficient computation is also discussed.

INTRODUCTION

Let A be the area, and P be the perimeter of a planar object. The circularity C defined by $C = 4\pi A/P^2$ is 1 for a circle and between 0 and 1 for all other shapes. The area, perimeter and circularity are useful features to describe the shape of a 2D object, and again the motion of it (e.g. in medical applications [1]). This paper deals with the estimation of the features from a discrete binary image.

The area can often be accurately estimated by counting the number of pixels inside an object. However, the estimation of the perimeter is a problem, since the length of the original contour might be considerably different from the length of the digital contour. It is impossible to reconstruct a general continuous contour from discrete data, because many possible contours, all having different lengths, correspond to a specific discrete realization. Thus, to develop a practical length estimator, some reasonable assumptions about the original contour should be made. Many authors [2]-[6] developed and evaluated length estimators for straight lines. One of these estimators was also found to be accurate for the boundaries of blob-like objects [2, 6].

We assume that the boundary of a blob-like object consists of chains of circular arcs, and therefore evaluate the precision of several area and length estimators applied to circles. The circularity C is a scale, translation and rotation invariant shape feature. The precision of the circularity is used as one of the measures for the evaluation. The problem of efficient and simultaneous computation of area and perimeter is also discussed.

A REVIEW OF METHODS

We describe some methods to estimate the area and the perimeter of objects represented by square pixels.

Methods Based on Bit Quads

Gray [7] proposed a systematic approach to computing the area and the perimeter. The method is also presented in the book of Pratt [8]. Each small region in a binary image is matched with some pixel patterns. The number of matches for each pattern is counted. The area and the perimeter are then formulated as weighted sums of the counts. Gray [7] designed a set of 2×2 pixel patterns called Bit Quads:

$$Q_0: \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix}$$

$$Q_1: \begin{matrix} 1 & 0 \\ 0 & 0 \end{matrix} \quad \begin{matrix} 0 & 1 \\ 0 & 0 \end{matrix} \quad \begin{matrix} 0 & 0 \\ 0 & 1 \end{matrix} \quad \begin{matrix} 0 & 0 \\ 1 & 0 \end{matrix}$$

$$Q_2: \begin{matrix} 1 & 1 \\ 0 & 0 \end{matrix} \quad \begin{matrix} 0 & 1 \\ 0 & 1 \end{matrix} \quad \begin{matrix} 0 & 0 \\ 1 & 1 \end{matrix} \quad \begin{matrix} 1 & 0 \\ 1 & 0 \end{matrix}$$

$$Q_3: \begin{matrix} 1 & 1 \\ 0 & 1 \end{matrix} \quad \begin{matrix} 0 & 1 \\ 1 & 1 \end{matrix} \quad \begin{matrix} 1 & 0 \\ 1 & 1 \end{matrix} \quad \begin{matrix} 1 & 1 \\ 1 & 0 \end{matrix}$$

$$Q_4: \begin{matrix} 1 & 1 \\ 1 & 1 \end{matrix}$$

$$Q_D: \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \quad \begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix}$$

Let $n\{Q\}$ be the number of matches between the image pixels and the pattern Q. Gray computed the area of the object as

$$A = \frac{1}{4}[n\{Q_1\} + 2n\{Q_2\} + 3n\{Q_3\} + 4n\{Q_4\} + 2n\{Q_D\}] \quad (1)$$

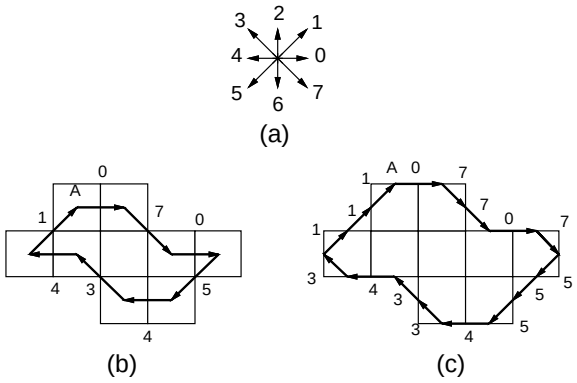


Figure 1: (a) *8 code words represent 8 directions.* (b) *The Freeman chain code of the object is 07054341.* (c) *The mid-crack code of the object is 0770755543343111. "A" indicates a start point.*

and the perimeter as

$$P = n\{Q_1\} + n\{Q_2\} + n\{Q_3\} + 2n\{Q_D\} \qquad (2)$$

The area computed by Eq. (1) is equal to the number of pixels of the object, which is known to be accurate. However, the perimeter formula of Gray is in considerable error for many types of objects [8].

Pratt [8] presented more accurate formulas for the area and the perimeter, citing an unpublished note of Duda

$$A = \frac{1}{4}n\{Q_1\} + \frac{1}{2}n\{Q_2\} + \frac{7}{8}n\{Q_3\} + n\{Q_4\} + \frac{3}{4}n\{Q_D\} \qquad (3)$$

and

$$P = n\{Q_2\} + \frac{1}{\sqrt{2}}\left[n\{Q_1\} + n\{Q_3\} + 2n\{Q_D\}\right] \qquad (4)$$

Methods Based on Chain Codes

Chain coding is a method to represent a binary object. The 8-connected Freeman chain coding [9] uses a 3-bit code $0 \le c \le 7$ for each boundary point. The number c indicates the direction in which the next boundary point is located, as shown in Fig. 1(a). The 8-connected Freeman chain coding strategy is shown in Fig. 1(b). There are some variations of the Freeman chain coding, for example, the 4-connected chain coding and the generalized chain coding [10].

The mid-crack chain coding [11] considers the mid-cracks instead of the centers of the boundary points. Assume that a pixel is a square with four sides. A mid-crack is then the mid-point of a pixel side. An example of the mid-crack coding is given in Fig. 1(c). The mid-crack codes possess some special properties in measuring shape features [11, 12].

Boundary chain codes can be determined using a contour following [8], which is a traversing process to identify the boundary of a binary object. The algorithm requires operations of $O(N)$.

Freeman [9] computed the area enclosed by the contour of the Freeman chain codes $c_1 c_2 \cdots c_n$

$$A = \sum_{i=1}^{n} c_{ix}(y_{i-1} + c_{iy}/2) \qquad (5)$$

where n is the length of the chain, c_{ix} and c_{iy} are the x and y components of the ith chain element c_i (c_{ix}, $c_{iy} \in \{1, 0, -1\}$ indicating the change of the x- and y-coordinates), and y_{i-1} is the y-coordinate of the start point of the chain element c_i in an arbitrary coordinate system. The values of c_{ix}, c_{iy} and y_{i-1} can be computed under the contour following.

Freeman [9] computed the perimeter as the length of the chain. The formula for the perimeter is

$$P = n_e + \sqrt{2}n_o \qquad (6)$$

where n_e is the number of even chain elements and n_o the number of odd chain elements. Referring to Fig. 1, an even chain element indicates a vertical or horizontal connection between two boundary pixels, having length 1, while an odd chain element indicates a diagonal connection, which has length $\sqrt{2}$.

Vossepoel and Smeulders [3] improved Freeman's method in estimating lengths of straight lines by using a corner count n_c, defined as the number of occurrences of consecutive unequal chain elements in the Freeman chain code string. The length is given by

$$P = 0.980n_e + 1.406n_o - 0.091n_c \qquad (7)$$

where the weights were found by a least-square fitting for all straight lines with $n_e + n_o = 1000$.

When the mid-crack chain codes are used, Eq. (5) can still be used to estimate the area. In this case, the computation of c_{ix} and c_{iy} is more complex since more possible values are involved, i.e., c_{ix}, $c_{iy} \in \{-1, -1/2, 0, 1/2, 1\}$. During the contour following, a sequence of background-to-object transitions can be detected. c_{ix} and c_{iy} can then be determined according to the types of two subsequent transitions. To estimate the perimeter, Eq. (6) becomes

$$P = n_e + \frac{\sqrt{2}}{2}n_o \qquad (8)$$

Although the methods are related to the chain coding, they can in fact determine the area and the perimeter without generating any chain codes. The values A, n_e, n_o and n_c can be computed by accumulation during the contour following.

The methods based on the chain coding compute the perimeter as the length of the chain, and often give an overestimated result. Kulpa [2] derived a compensation factor for computing the length of straight lines. With this factor, Eq. (6) becomes

$$P = \frac{\pi}{8}(1 + \sqrt{2})(n_e + \sqrt{2}n_o) \qquad (9)$$

where the factor is approximately 0.948. Kulpa [2] found that this compensation also gave good results for most of the blob-like objects met in practice. Dorst and Smeulders [6] proved that Eq. (9) gave a consistent estimate for the length of a circular arc of $\pi/4$.

Discrete Green's Theorem

Freeman's method evaluates the area of a polygon enclosed by the chain elements, using an $O(N)$ algorithm. The result is different from that of Gray's method, which equals the area to the number of pixels of a discrete region. Gray used an $O(N^2)$ algorithm to count the number of pixels. However, the counting can be done in the time of $O(N)$ by using a discrete Green's theorem [13], which computes a sum of a two-dimensional function over a discrete region by a summation along its discrete boundary. The discrete Green's theorem gives exact result of a double sum, and has been used for fast and exact computation of geometric moments [14]. The area is the zeroth order moment of a homogeneous region.

EXPERIMENTS AND RESULTS

The methods to be tested are the Bit Quad methods of Gray and Duda, Freeman's method and its analogue using the mid-crack chain codes, and Kulpa's method given by Eq. (9). We tested the precision of these methods in estimating the areas and the perimeters of circles of radius R having integer values from 5 to 70 pixels.

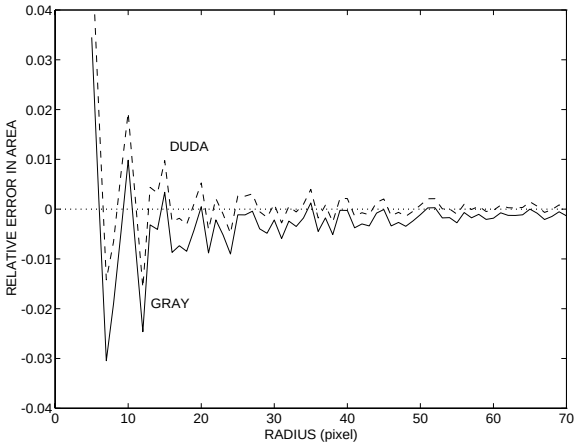
Binary test images of the circles were generated by giving intensity value

$$g(x,y) = \begin{cases} 1 & \text{if } (x-x_0)^2 + (y-y_0)^2 \le R^2 \\ 0 & \text{otherwise} \end{cases}$$

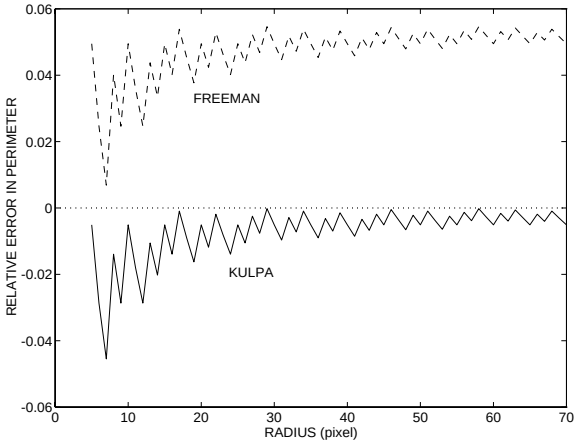
where (x_0, y_0) is the coordinate of the centroid.

Using the above methods, we estimated the areas $\hat{A}$ and the perimeters $\hat{P}$ of the circles, and computed the relative errors defined by "relative error $= (\hat{x} - x)/x$" where x is the true value. The true values of the area and perimeter are evaluated by $A = \pi R^2$ and $P = 2\pi R$.

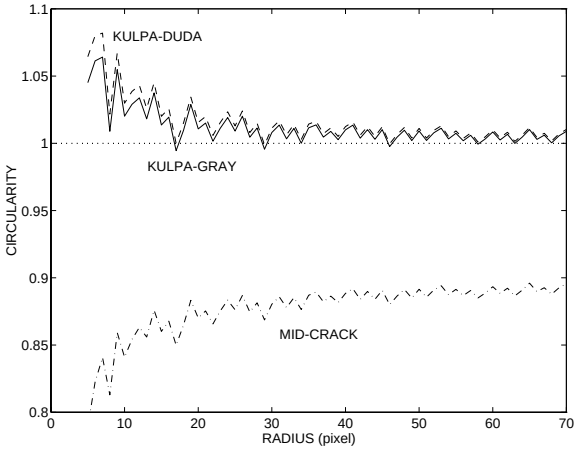
The relative errors in area given by the Gray and the Duda method are shown in Fig. 2(a). We see that the area estimations of Gray and Duda are both good. The result of the Duda method is slightly better. The average relative error for $15 \le R \le 70$ was 0.0003 for the Duda method and -0.0025 for the Gray method. The mid-crack method gave a result very similar to that of Gray. The Freeman method underestimated the area, giving a relative error similar to that of the Duda method if we assume that the radius is $R - 0.5$.



(a)



(b)



(c)

Figure 2: (a) *The relative errors in the areas estimated by the method of Gray, and Duda.* (b) *The relative errors in the perimeters estimated by the method of Freeman, and Kulpa.* (c) *The circularities estimated by combining different area and perimeter estimators, i.e. Kulpa's perimeters with Gray's areas, Kulpa's perimeters with Duda's areas, and the mid-crack perimeters with the mid-crack areas. The radius has integer values from 5 to 70 pixels.*

From Fig. 2(b) we see that the perimeters estimated by using Kulpa's compensation factor is more accurate than those estimated by the method of Freeman, which gave an overestimation. Gray's method was very inaccurate, giving a relative error of about 0.3 (overestimated). The methods of Duda and mid-crack all overestimated the perimeters. The relative errors of these two methods are similar to that of the Freeman method if we assume the radius is $R + 0.5$.

Combining the estimators of different methods, we computed the circularities shown in Fig. 2(c). We see that using Kulpa's perimeter estimator together with Gray's area estimator gives the best result, which is close to but often slightly larger than the true value 1. It is better than combining Kulpa's perimeter with Duda's area although Duda's area is better than Gray's area. This is because Kulpa's perimeter and Gray's area are both slightly underestimated. Other combinations do not give good results. As an example we show the results when the areas and the perimeters are both computed by the mid-crack method.

We can observe that the variance of the error is large when the value of R is small. This is because the spread of the ground truth is large when R is small, and suggests that we should have a sufficiently large resolution in order to obtain a good estimation.

DISCUSSION

Different methods have different computational complexity. Using the Bit Quads, the order of the computation is N^2. (We assume that an image has N^2 pixels.) It can be reduced to N by using a contour following algorithm.

The area can be formulated as the number of pixels in the object, or an integration over an approximated continuous region. The perimeter can be formulated as the length of the boundary of a polygon, approximating the original object. This length can be multiplied by a compensation factor, giving a better estimation. All the length estimators presented above can be generalized as a linear model

$$P = \mathbf{w}^T \mathbf{n} \tag{10}$$

where the object is characterized by a set of counts $\mathbf{n}$, such as the counts of the pixel patterns, or the counts of the even and odd chain elements. The perimeter is computed as linear combinations of the counts, using $\mathbf{w}$ as a set of weights. Nonlinear estimators for straight lines have also been developed [6].

We tested the accuracy of several area and perimeter estimators for circles, assuming that the boundary of a blob-like object is approximately a chain of circular arcs. From the above experiment, we see that Gray's Bit Quad method gives a good estimation of the area, but a bad estimation of the perimeter. Gray's method has been improved by Duda in both the area and the perimeter estimation. But there is still a large bias in the perimeter which causes a relative error of about 5 percent, and Duda's method overestimates the perimeter compared to the area. Freeman's method and the mid-crack method give results which are similar to that of Duda's method, but improve the computational performance by reducing the order from N^2 to N. The perimeters computed by Kulpa's method are much better than all the other methods, giving a small underestimation.

Different methods may have different assumptions of the location of the object boundary. The mid-crack method assumes that the boundary goes through the mid-cracks, and Freeman's method assumes that the boundary goes through the centers of the boundary pixels. The two boundaries are located in a distance of a half pixel. That means the area and the perimeter estimated by the mid-crack method are always larger than those estimated by the Freeman method.

To compute the circularity, the best result is obtained by using Gray's estimator of the area and Kulpa's estimator of the perimeter. However, they can not be computed simultaneously. Gray's area is equal to the number of pixels in the region, which can be computed by using a discrete Green's theorem. This suggests the use of the discrete Green's theorem [13, 14] instead of Gray's algorithm. Then the two estimators can be computed simultaneously by a contour following. Analogous to Green's theorem, the discrete Green's theorem evaluates a double sum over a discrete region by a single summation along the discrete boundary of the region, and thus gives computational advantages. As shown in a recent paper [15], it can also be extended to estimate the volume of a 3D object.

The results of the test, using test images of circles, should be useful for other blob-like objects. However, different shapes may require different perimeter estimators. It is therefore interesting to see how a good estimator can be found for a given type of shape. If one desires an optimal estimator, a faithful characterization (a set of counts) should be made. Dorst and Smeulders [6] believed that it was very difficult, and was even impossible for circular arcs. But, as a method to analyze a given characterization, they divided the parameter space of the continuous data (one dimensional R-space for the case of a circle) into regions each corresponding to one value of the discrete characterization $\mathbf{n}$. The region imply the spread of the ground truth for a given value of $\mathbf{n}$. Vossepoel and Smeulders [3] used three counts (see Eq. 7) as a characterization to estimate the length of straight lines. They found the optimal weights by a least-square fitting. This method suggests a way to design a linear estimator. Using the linear model

given by Eq. 10, the problem of finding a good estimator is to find a set of counts (also known as the characterization of the discrete data [6]), and then to determine the optimal weights.

CONCLUSION

In this paper, we give a review of several area and perimeter estimation techniques. The area, perimeter and circularity are features used in shape analysis. An accurate estimation of the circularity depends on accurate estimations of the area and the perimeter. The area of a binary region can be accurately estimated by counting the number of the pixels inside the region. However, to estimate the perimeter is more difficult. To find a good perimeter estimator, it is necessary to make some assumptions about the boundary of the object. We assume that the boundary is a chain of circular arcs. This assumption should be useful for many blob-like objects met in practice. Many estimators have been tested for circles of different sizes. We conclude that with a sufficiently large resolution all the methods give good estimations of the area, and the method of Kulpa gives a good estimation of the perimeter. To compute the circularity, the best result can be obtained by using Kulpa's perimeter, and Gray's area, which is the number of the pixels of the region. The Gray's area can be computed by a discrete Green theorem. Then the area and the perimeter can be computed simultaneously and efficiently, based on a contour following algorithm.

References

[1] M. D. Levine and P. B. Noble and Y. M. Youssef. Understanding blood cell motion. *Comput. Vision Graph. Image Process.*, **21**:58–84, 1983.

[2] Z. Kulpa. Area and perimeter measurement of blobs in discrete binary pictures. *Comput. Graph. Image Process.*, **6**:434–451, 1977.

[3] A. M. Vossepoel and A. W. M. Smeulders. Vector code probability and metrication error in the representation of straight lines of finite length. *Comput. Graph. Image Process.*, **20**:347–364, 1982.

[4] L. Dorst and A. W. M. Smeulders. Discrete representation of straight lines. *IEEE Trans. Pattern Anal. Machine Intell.*, **6**(4):450–463, 1984.

[5] L. Dorst and A. W. M. Smeulders. Best linear unbiased estimators for properties of digitized straight lines. *IEEE Trans. Pattern Anal. Machine Intell.*, **8**:276–282, 1986.

[6] L. Dorst and A. W. M. Smeulders. Length estimators for digitized contours. *Comput. Vision Graph. Image Process.*, **40**:311–333, 1987.

[7] S. B. Gray. Local properties of binary images in two dimensions. *IEEE Trans. Computers*, **20**(5):551–561, 1971.

[8] W. K. Pratt. *Digital image processing.* Wiley-Interscience, 2 edition, 1991.

[9] H. Freeman. Boundary encoding and processing. In B. S. Lipkin and A. Rosenfeld, editors, *Picture Processing and Psychopictorics*, pages 241–266. Academic Press, 1970.

[10] J. A. Saghri and H. Freeman. Analysis of the precision of generalized chain codes for the representation of planar curves. *IEEE Trans. Pattern Anal. Machine Intell.*, **3**(5):533–539, 1981.

[11] K. A. Dunkelberger and O. R. Mitchell. Contour tracking for precision measurement. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 22–27, St. Louis, 1985.

[12] F. Y. Shin and W.-T. Wong. A new single-pass algorithm for extracting the mid-crack codes of multiple regions. *Journal of Visual Communication and Image Representation*, **3**(3):217–224, 1992.

[13] L. Yang and F. Albregtsen. Discrete Green's theorem and its application in moment computation. In *Proc. 1st Int. Conf. Electronics and Information Technology*, pages 27–31, Beijing, China, 1994.

[14] L. Yang and F. Albregtsen. Fast computation of invariant geometric moments: a new method giving correct results. In *Proc. 12th Int. Conf. Pattern Recognition*, Jerusalem, Israel, 1994.

[15] L. Yang and F. Albregtsen. Fast and exact computation of Cartesian geometric moments using discrete Green's theorem. Submitted to *Pattern Recogn.*, 1994.