International Journal of Image and Graphics
Vol. 4, No. 1 (2004) 34–49
© World Scientific Publishing Company

AUTOMATIC DETECTION OF CIRCULAR OBJECTS BY ELLIPSE GROWING

KENICHI KANATANI
Department of Information Technology, Okayama University,
Okayama 700-8530, Japan
E-mail: kanatani@suri.it.okayama-u.ac.jp

NAOYA OHTA
Department of Computer Science, Gunma University,
Kiryu, Gunma 376-8515, Japan
E-mail: ohta@cs.gunma-u.ac.jp

Received 23 May 2002
Revised 25 December 2002

We present a new method for automatically detecting circular objects in images: we detect an osculating circle to an elliptic arc using a Hough transform, iteratively deforming it into an ellipse, removing outlier pixels, and searching for a separate edge. The voting space for the Hough transform is restricted to one and two dimensions for efficiency, and special weighting schemes are introduced to enhance the accuracy. We demonstrate the effectiveness of our method using real images. Finally, we apply our method to the calibration of a turntable for 3-D object shape reconstruction.

Keywords: ellipse fitting; Hough transform; outlier detection; robust estimation; LMedS

1. Introduction

Since a circle in the scene is projected into an ellipse in the image, circular objects in the scene can be detected by finding ellipses in the image. Recently, many algorithms have been proposed for fitting an ellipse to edge pixels with high precision[4,5,13,14]. Using the equation of the fitted ellipse, the 3-D position and orientation of the circular object can be computed analytically[8,19].

However, finding correct edge pixels to fit an ellipse is still a very difficult task. One approach is to segment digital curves into linear and curved parts, fitting lines to the linear parts, and ellipses to the curved parts[6,16]. The segmentation is based on the digital curvature, the residual of fitting, and miscellaneous heuristics, but the accuracy of the fit depends very much on segmentation. This may not be a great problem for visual applications but is crucial for precise measurement in robotics applications, with which we are concerned.

Another popular approach is to search for an ellipse directly using the Hough

transform. Since an ellipse is specified by five parameters, we can find it by voting in a quantized 5-dimensional Hough space the parameters of the ellipses that could pass through each edge pixel, and picking out the value that wins the maximum number of votes. However, directly computing this would require a long computation time and a memory space proportional to the fifth power of the quantization levels. Various techniques have been proposed for reducing the computation time and memory space. For example, the computation is divided into a cascade: voting is done in a 2-dimensional Hough space with the remaining three parameters fixed, and this is repeated for all quantized values of the three parameters. In this process, one can introduce various constraints for pruning unnecessary search, hierarchically change the resolution of the image and the Hough space, and do random sampling instead of exhaustive search, a variant of this being the genetic algorithm. Instead of using edge pixels alone, one can also use complex primitives and extra information such as point pairs, triplets, edge segments, and their orientations.

Although many extensions and variations have been proposed in the past [1,3,10,11,12,15,17,18,21,22,23,24], the Hough transform alone is not efficient enough, while the use of digital curves alone does not warrant sufficient accuracy. In this paper, we integrate these two: we detect a circle osculating to an edge segment by the Hough transform, find edge pixels tangent to it, fit an ellipse to them, update the edge pixels tangent to it, and repeat this process. We call these iterations *ellipse growing*.

A closely related method to ours is the work of Asayama and Shiono [1], who detected via the 5-dimensional Hough transform the two circles osculating to an ellipse from both sides, from which they computed the parameters of the ellipse. Their method works only when the entire ellipse is visible without occlusion, and a long computation time and a large memory space are required for the Hough transform. In contrast, our method is able to fit an ellipse to a partially occluded image of a circular object very efficiently with high precision.

The Hough transform we propose for detecting an osculating circle is made efficient by using, instead of the standard 3-dimensional Hough space, a 2-dimensional array for the center and a 1-dimensional array for the radius. We also introduce special weight functions for enhancing the vote peaks. We then iteratively deform the osculating circle into an ellipse. We also remove outlier edge pixels by LMedS, and search for another edge segment to combine. Figure 1 shows the flow chart of the procedure.

Our system is not intended to have universal merits, whose pursuit is rather unrealistic. We have in mind robotics environments where

(i) only a small number of circular objects exist in the field of view, and
(ii) the circular objects we are looking at have an appreciable size in the image (see the real image examples shown later).

We do not consider such complicated scenes as aggregates of circular and elliptic

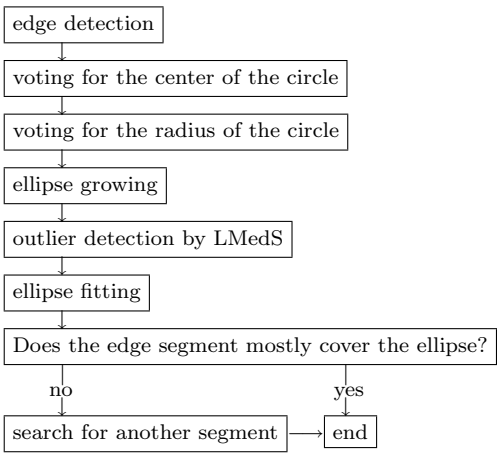


Fig. 1. Flow chart of the procedure.

particles or cells; our method is strengthened by specifically exploiting this situation. Still, it is impossible to cope with all possible scenes, so some kind of tuning is unavoidable. In this respect, there is still room for further improvement.

In the following, we describe each step separately and confirm the effectiveness of our method using real images. Finally, we apply our method to the calibration of a turntable for 3-D object shape reconstruction.

2. Procedure

2.1. *Edge detection*

From among many edge detection techniques available, we adopt the following method for obtaining long edge segments without branches. First, we cut out from the zero-crossing edges of one-pixel width those parts with high Sobel values. Starting from an arbitrary pixel of the resulting edges, we trace the 8-connected neighbors in both directions until we arrive at an end-point or a branch point, defining a connected edge segment without branches. This process is repeated until all the edge pixels are exhausted. Among the resulting edge segments, we discard short ones (less than 60 pixels in our experiment) and register the remaining ones in an edge segment list.

2.2. *Voting for the center*

Among the registered edge segments, we first look for those to which circles can fit well, at least locally, as a first approximation to ellipses. We begin by estimating their centers by voting.

For an exactly circular arc, its center is easily estimated by voting along all lines

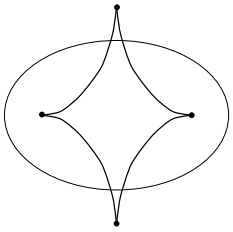
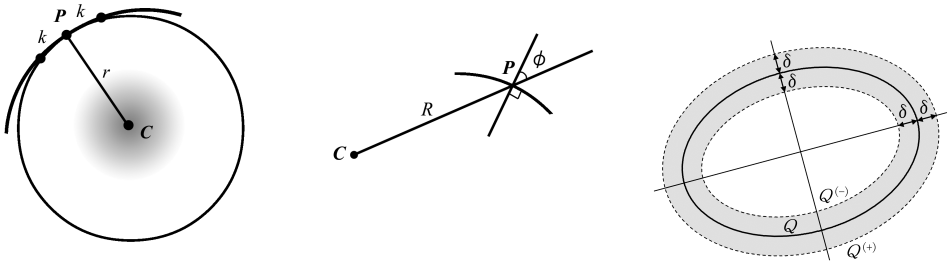


Fig. 2. The evolute of an ellipse and its singularities.

normal to the arc[7]. However, the lines normal to an ellipse envelop a star-like curve with four cusps, known as the *evolute* of the ellipse (Fig. 2). An osculating circle whose center is on the evolute is generally of second degree contact to the curve, but if the center is at one of the four singularities, the degree is three[2]. Hence, if we vote along all lines normal to an elliptic arc, we obtain ridges along the evolute and peaks at its singularities. However, the peaks are less conspicuous than for a circular arc. So, we directly vote on the evolute as follows.

For each pixel P on the edge segment, we compute the center C and the radius r of the circle that passes through P and two points on the segment k pixels away from it on both sides (we exclude the k end points of the segment). Then, we vote around C with Gaussian smoothing of standard deviation γr in all directions (Fig. 3(a)). We also weight each vote by $1/\sqrt{r}$ to give preference to edge segments of higher curvature if the length is the same and of larger radii if the central angle is the same. In our experiment, we set $k = 30$ and $\gamma = 1/10$. Since it is difficult to derive theoretically optimal values for the parameters and weights, we set them heuristically.

We do this for all the edge segments in the list[a] and detect peaks of the votes (a



(a) (b) (c)

Fig. 3. (a) Voting around the center of the circle passing through P and the two points away from it by k pixels on both sides. (b) A pixel away from the estimated center C by distance R votes the value R with weight $e^{-R^2/2s^2}\cos\phi$. (c) Fitting an ellipse to the longest edge segment inside the region within distance δ from the estimated ellipse.

[a]In our experiment, we voted every three pixels for efficiency. We also introduced various approximations such as limiting the voting region to a parallelogram with one edge orthogonal to the edge segment and the other parallel to one of the coordinate axes.

real image example is given in Fig. 5; the details are described in Sec. 3). Considering the fact that the evolute of an ellipse has at most four singularities (Fig. 2), we take as candidates the four pixels that win the largest numbers of votes.

2.3. *Voting for the radius*

For each center C thus detected, we estimate the corresponding radius by voting the distance R from C to each edge pixel and adopting the value that wins the maximum number of votes. This may be obvious for a circular arc[7], but for an elliptic arc the number of false peaks increases. So, we adopt the following strategies.

First, instead of casting one vote to the distance R to each edge pixel P, we cast a *positive/negative real value* $e^{-R^2/2s^2}\cos\phi$, where ϕ is the angle between the line CP and the normal direction to P (Fig. 3(b)), which is computed from the output of the Sobel operator. Second, we vote not merely for an integer obtained by rounding R but also the two integers on both sides by ± 1 with the weight $e^{-1/2}$ (i.e., the Gaussian weight of standard deviation 1).

The value[b] $e^{-R^2/2s^2}$ effectively limits the voting to approximately within distance s from C (in our experiment, we empirically set s to be 1/4 of the image size). Due to $\cos\phi$, those edge segments that make large angles from the circumference are given small weights. In particular, *the weight is 0 if the segment is in the radius direction*. Since edge segments with opposite gray-level gradients have weights of *opposite signs*, the contributions from closed loops that do not surround P are mutually canceled, enhancing only the contributions from the edge segment to which the circle osculates (see Fig. 6 for this effect; the details are described in Sec. 3).

After this voting, we choose the value that wins the maximum number of votes. We do this for each of the four candidates for the center and pick out the one that wins the largest number of votes.

2.4. *Ellipse growing*

Regarding the detected osculating circle as the initial ellipse, we make it grow into an ellipse that better fits the segment. For the current ellipse, we define a ring region around it by expanding and contracting the major and minor axes by δ pixels (Fig. 3(c)). From among the registered edge segments, we choose the one that has the largest number of consecutive pixels inside this region and fit an ellipse to it. For this, we used a technique called *renormalization*[c], which is known to attain the highest possible accuracy in the first order[9]. Around the fitted ellipse, we recompute a ring region, to which the same procedure is applied and repeated until no new pixels are added.

---

[b] We used a polynomial approximation of e^{-x^2} for efficiency.

[c] The program is publicly available from `http://www.ail.cs.gunma-u.ac.jp/Labo/programs-e.html`

The ring region is defined as follows. Quadratic curves (including ellipses, hyperbolas, and parabolas) have equations of the form

$$Ax^2 + 2Bxy + Cy^2 + 2f(Dx + Ey) + f^2F = 0, \tag{1}$$

where f is an arbitrarily fixed constant (e.g., the image size). The condition that this equation describes an ellipse (i.e., not a hyperbola or a parabola) is[8]

$$AC - B^2 > 0 \tag{2}$$

If we define the vector $\boldsymbol{x}$ and the matrix $\boldsymbol{Q}$

$$\boldsymbol{x} = \begin{pmatrix} x/f \\ y/f \\ 1 \end{pmatrix}, \qquad \boldsymbol{Q} = \begin{pmatrix} A & B & D \\ B & C & E \\ D & E & F \end{pmatrix}, \tag{3}$$

Eq. (1) is rewritten in the form

$$(\boldsymbol{x}, \boldsymbol{Q}\boldsymbol{x}) = 0, \tag{4}$$

where $(\boldsymbol{a}, \boldsymbol{b})$ denotes the inner product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ (see Ref. 8 for the general analysis of quadratic curves). The matrices $\{\boldsymbol{Q}^{(+)}, \boldsymbol{Q}^{(-)}\}$ of the expanded and contracted ellipses are computed as follows (Appendix A):

1 Compute the matrix $\boldsymbol{S}$, the vector $\boldsymbol{c}$, and the scalar c as follows:

$$\boldsymbol{S} = \begin{pmatrix} A & B \\ B & C \end{pmatrix}, \quad \boldsymbol{c} = \begin{pmatrix} D \\ E \end{pmatrix}, \quad c = (\boldsymbol{c}, \boldsymbol{S}^{-1}\boldsymbol{c}) - F. \tag{5}$$

2 Compute the eigenvalues $\{\lambda_1, \lambda_2\}$ of $\boldsymbol{S}$ and the corresponding unit eigenvectors $\{\boldsymbol{u}_1, \boldsymbol{u}_2\}$.

3 Compute $\lambda_1^{(\pm)}$ and $\lambda_2^{(\pm)}$ as follows:

$$\lambda_1^{(\pm)} = \frac{c}{(\sqrt{c/\lambda_1} \pm \delta/f)^2}, \qquad \lambda_2^{(\pm)} = \frac{c}{(\sqrt{c/\lambda_2} \pm \delta/f)^2}. \tag{6}$$

4 Compute the matrices $\boldsymbol{S}^{(+)}$ and $\boldsymbol{S}^{(-)}$ by

$$\boldsymbol{S}^{(\pm)} = \boldsymbol{U} \begin{pmatrix} \lambda_1^{(\pm)} & \\ & \lambda_2^{(\pm)} \end{pmatrix} \boldsymbol{U}^\top, \tag{7}$$

where $\boldsymbol{U}$ is the 2×2 matrix consisting of vectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ as its columns in that order.

5 Compute the matrices $\{\boldsymbol{Q}^{(+)}, \boldsymbol{Q}^{(-)}\}$ as follows:

$$\boldsymbol{Q}^{(\pm)} = \begin{pmatrix} \boldsymbol{S}^{(\pm)} & \boldsymbol{S}^{(\pm)}\boldsymbol{S}^{-1}\boldsymbol{c} \\ (\boldsymbol{S}^{(\pm)}\boldsymbol{S}^{-1}\boldsymbol{c})^\top & F + (\boldsymbol{c}, \boldsymbol{S}^{-1}(\boldsymbol{S}^{(\pm)} - \boldsymbol{S})\boldsymbol{S}^{-1}\boldsymbol{c}) \end{pmatrix}. \tag{8}$$

Then, the ring region is define by

$$(\boldsymbol{x}, \boldsymbol{Q}^{(+)}\boldsymbol{x})(\boldsymbol{x}, \boldsymbol{Q}^{(-)}\boldsymbol{x}) < 0. \tag{9}$$

The renormalization method fits a general quadratic curve of the form of Eq. (1), so it sometimes fits a hyperbola or a parabola when the segment is very short. In that case, we progressively increment δ from its initial value (4 pixels in our experiment) by one pixel at a time until Eq. (2) is satisfied. If δ reaches 10 pixels but Eq. (1) is still not satisfied, we return the initial circle and go on to the next process (see Fig. 7 for a real image example; the details are described in Sec. 3).

2.5. *Outlier removal*

The edge pixels to which the ellipse is thus fitted may not be part of an ellipse. So, we remove "outliers[d]" by *LMedS*[20]. Let $\{\boldsymbol{x}_\alpha\}$, $\alpha = 1, ..., N$, be the vector representations (see Eq. (3)) of the pixels to fit an ellipse. Initializing the matrix $\boldsymbol{Q}_m = \boldsymbol{O}$ and the scalar $S_m = \infty$, we repeat the following computation until it converges:

1 Randomly choose five points from $\{\boldsymbol{x}_\alpha\}$.

   # Five points can uniquely determine an ellipse.

2 Compute an ellipse $\boldsymbol{Q}$ that passes through them.

   # We solve simultaneous linear equations in A, B, ..., F in the form of Eq. (1) up to a constant factor.

3 Go back to Step 1 if Eq. (3) is not satisfied.

   # This excludes hyperbolas and parabolas.

4 Compute the following median:

$$S = \mathrm{med}_{\alpha=1}^{N} \frac{(\boldsymbol{x}_\alpha, \boldsymbol{Q}\boldsymbol{x}_\alpha)^2}{\|\boldsymbol{P}_k \boldsymbol{Q}\boldsymbol{x}_\alpha\|^2}. \tag{10}$$

   # Here, $\boldsymbol{P}_k = \mathrm{diag}(1, 1, 0)$, i.e., the diagonal matrix with 1, 1, and 0 as the diagonal elements in that order.
   # The quotient on the right-hand side equals the squared distance of the αth pixel to the fitted ellipse to a first approximation[9]:

5 If $S < S_m$, update $\boldsymbol{Q}_m \leftarrow \boldsymbol{Q}$ and $S_m \leftarrow S$.

   # The value S_m records the maximum median obtained so far. The matrix $\boldsymbol{Q}_m$ designates the corresponding ellipse.
   # We stopped the iterations when $S \geq S_m$ for 10 consecutive times.

[d]In the statistical literature, "outliers" are the data generated by unmodeled random fluctuations[20]. Here, they are contiguous pixels on non-elliptic edge segments. In other words, "outlier removal" actually means "segmentation" of elliptic parts.

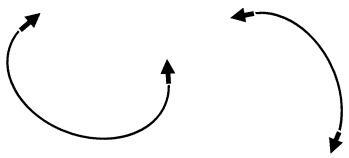


Fig. 4. Judging if the edge segment covers more than half of the ellipse.

After convergence, we selected inliers by the criterion

$$\frac{(\boldsymbol{x}_\alpha, \boldsymbol{Q}\boldsymbol{x}_\alpha)^2}{\|\boldsymbol{P}_k\boldsymbol{Q}\boldsymbol{x}_\alpha\|^2} < 10S_m, \tag{11}$$

which can be obtained by estimating the data standard deviation σ according to the formula $\hat{\sigma} \approx 1.4826\sqrt{S_m}$ given in Ref. 20 and setting the threshold to $(2.13\hat{\sigma})^2$. Finally, an ellipse $\boldsymbol{Q}$ is refitted to the detected inliers by renormalization, and the longest segment e that covers all the inliers is cut out.

2.6. *Search for another segment*

We now find ellipses consisting of multiple segments. First, we check if the detected segment e covers more than half of the entire ellipse. This is done by checking if the end parts of e are in converging orientations (Fig. 4). If so, we stop and return the ellipse $\boldsymbol{Q}$. Otherwise, we expand $\boldsymbol{Q}$ by $1+\gamma$. This is done by replacing Eqs. (6) by

$$\lambda_1^{(+)} = \frac{\lambda_1}{(1+\gamma)^2}, \qquad \lambda_2^{(+)} = \frac{\lambda_2}{(1+\gamma)^2}. \tag{12}$$

(We let $\gamma = 1.2$). We randomly choose a segment e' that is within the expanded ellipse and is not too short (we ignored those of less than 20 pixels). Randomly choosing four pixels from e and one from e', we fit an ellipse by LMedS as described earlier. This time, we evaluate, instead of Eq. (10), the sum of the medians computed for e and e'. We repeat this until the sum of the medians converges (we stopped if no update occurred 100 consecutive times). We return the initial $\boldsymbol{Q}$ if the resulting sum of medians is larger than four times the initial median for e (this criterion is heuristically introduced). Otherwise, we detect inliers by applying Eq. (11) to e and e' separately and fit an ellipse to the detected inliers by renormalization.

We could find more segments by repeating this procedure. However, the chance to pick out wrong segments will increase. Also, finding two segments is usually sufficient for robustly fitting an ellipse. So, we stop the search after one separate segment is found. Since it is difficult to derive a theoretically optimal strategy for searching and voting, the above procedure was introduced empirically.

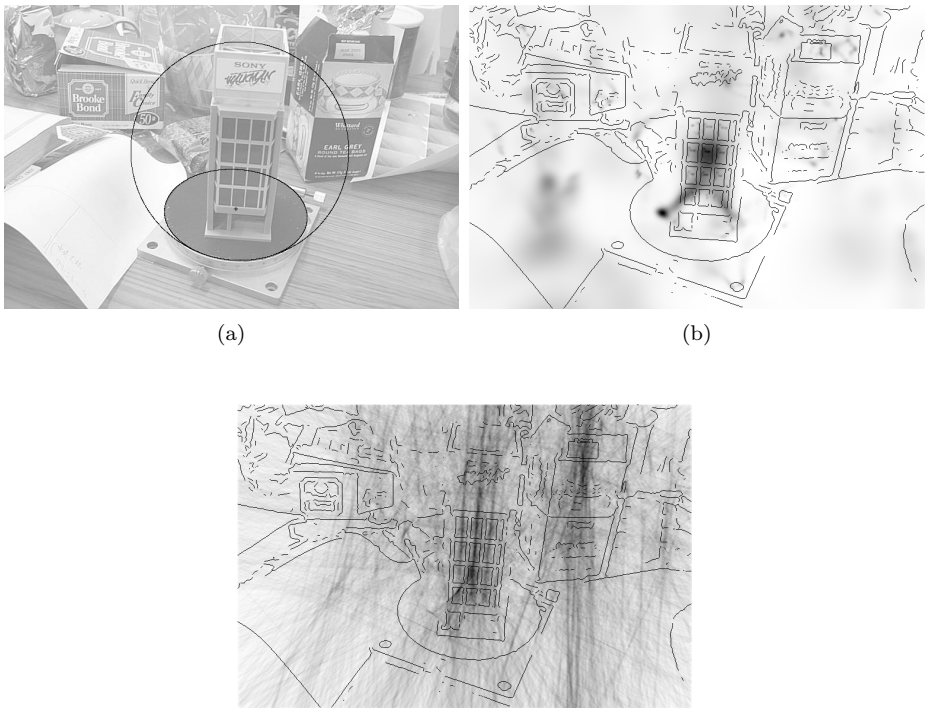
3. Real Image Examples

Figure 5(a) shows the detected osculating circle and the final ellipse superimposed onto the original image. Figure 5(b) is the initial edge image, onto which the votes for potential centers of osculating circles are superimposed in gray levels. For

comparison, Fig. 5(c) shows the result obtained by the standard Hough transform: we vote along the normal line to each edge pixel with Gaussian smoothing of one-pixel standard deviation on both sides[7]. As we can see, our scheme concentrates more votes along the evolute of the ellipse, in particular at its singularities.

Figure 6(a) plots the absolute value (recall that we vote positive and negative values) of the number of votes (ordinate) for the radius R (abscissa) from the detected center. For comparison, Fig. 6(b) shows the plot obtained by the standard Hough transform: the distance from the estimated center to each edge pixel is voted with Gaussian smoothing of one-pixel standard deviation[7]. We adjusted the scale so that the total number of votes is equal for both. As we can see, our scheme reduces the contributions from clustered edges to almost zero because of the cancellation of the signs, enhancing only the contributions from isolated arcs.

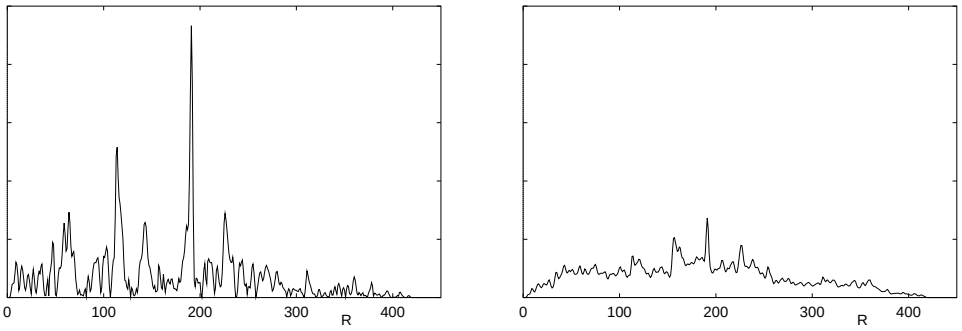
Figure 7(a) shows an edge image of a partially occluded ellipse. For this image, a hyperbola was fitted in the ring region along the initial circle even for $\delta = 10$. So, we fitted an ellipse by LMedS and went on to search for another segment. Figure 7(b) shows (1) the initial circle, (2) the hyperbola resulting from the ellipse growing,



(a) (b)

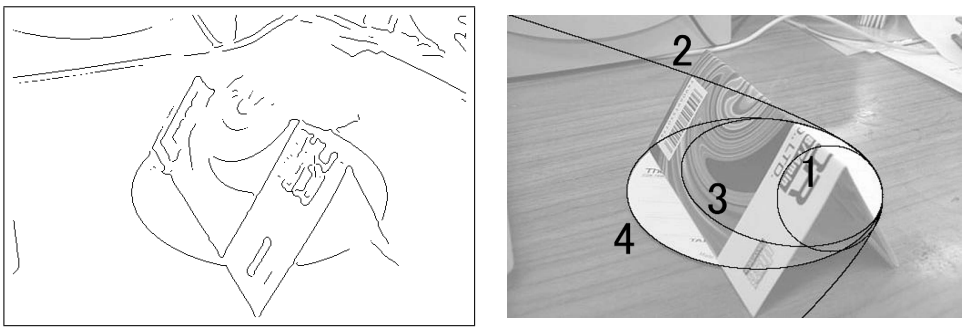
(c)

Fig. 5. Estimation of the center of the osculating circle. (a) The detected osculating circle and the final ellipse. (b) Voting for the candidates of the center. (c) Standard Hough transform.



(a) (b)

Fig. 6. The number of votes (ordinates) for the radius R of the initial circle (abscissa). (a) Proposed method. (b) Standard Hough transform.



(a) (b)

Fig. 7. (a) The edge image used. (b) (1) The initial circle, (2) the hyperbola resulting from the ellipse growing, (3) the ellipse fitted by LMedS, and (4) the ellipse fitted after detecting another segment.

(3) the ellipse fitted by LMedS, and (4) the ellipse fitted after detecting another segment.

Figure 8 shows various examples, showing the edge images (upper rows) and the initial circles and the final ellipses superimposed onto the original images (lower rows). In all cases, one or two iterations of the ellipse growing produced almost satisfactory shapes. The iterations terminated after four or five runs. As we see, we have obtained good fits even from very short edge segments.

We used Pentium III for the CPU and Linux for the OS. For 500×333-pixel images, the average computation time was 18 seconds, about half of which was spent on the preprocess of edges. As a result, computation time is nearly proportional to the number of edge segments. There is still room to increase efficiency by refining the program code.

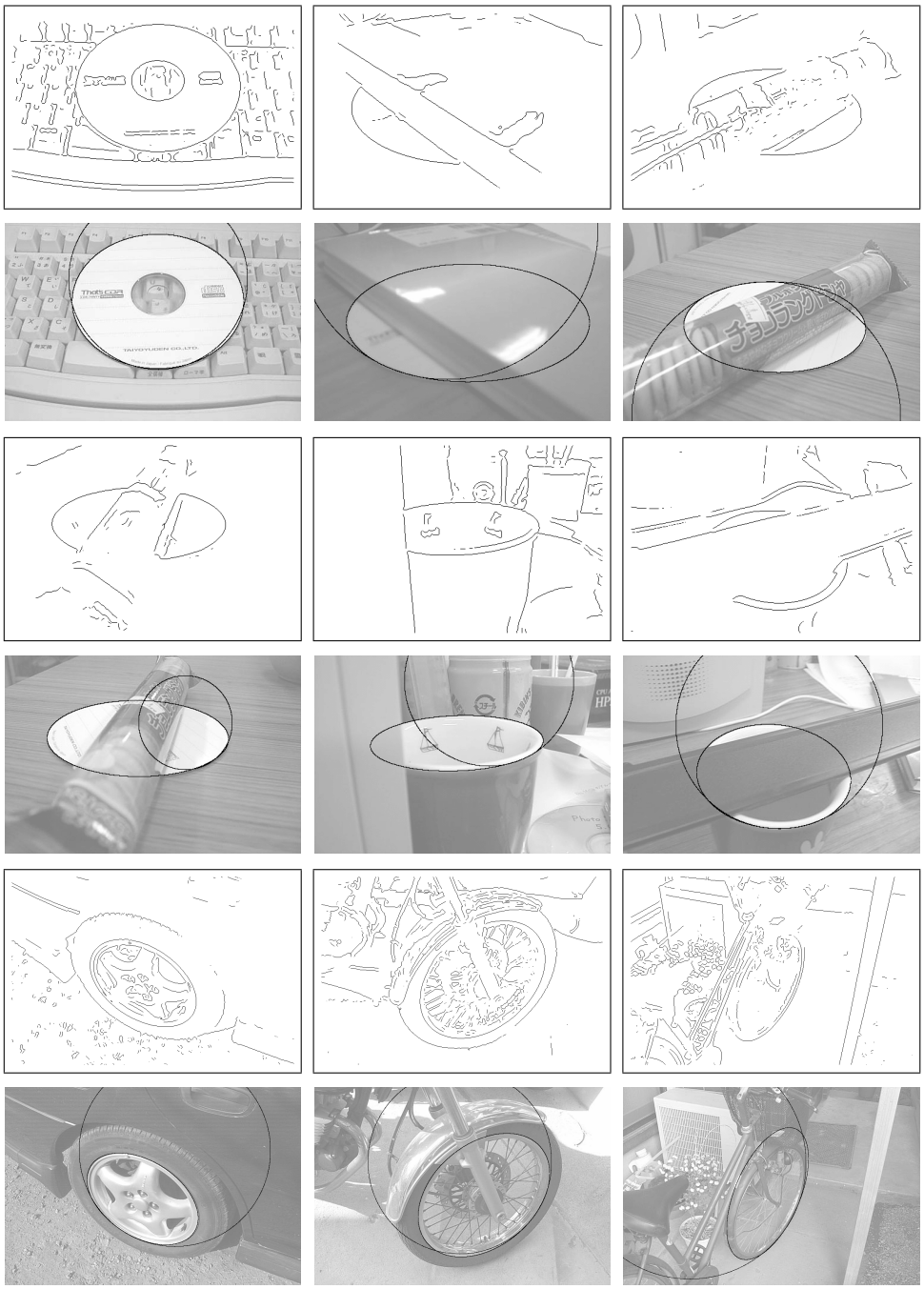
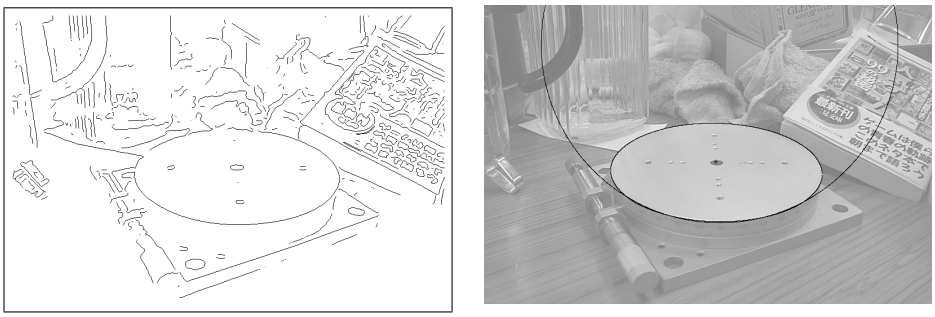


Fig. 8. Examples of ellipse fitting: the edge images (upper rows); the initial circles and the final ellipses superimposed onto the original images (lower rows).



(a) (b)

Fig. 9. Calibration of a turntable. (a) The edge image used. (b) The initial circle and the final ellipse superimposed onto the original image.

As a final example, we applied our method to the calibration of a turntable. One of the widely used methods for 3-D shape reconstruction is to place an object on a rotating turntable and take its images. This is equivalent to rotating the camera around the object, so the object shape can be reconstructed by triangulation. To do this, we need to calibrate the position of the turntable relative to the camera. For a circular turntable, this is easily done using its camera image alone if we measure its radius (Appendix B).

Figure 9 (a) is an edge image of a circular turntable. Fig 9(b) shows the initial circle and the final ellipse superimposed onto the original image (768×512 pixels). The focal length was calibrated to be 700 pixels. The radius of the turntable was measured to be 6.45cm. Using the procedure described in Appendix B, we found the angle between the camera optical axis and the turntable axis to be 59.4°, and the distance between the lens center to the turntable center to be 27.3cm. The accuracy of this computation depends on the accuracy of the camera parameters. Here, however, we focus only on image processing and do not go into the details of reliability analysis.

4. Concluding Remarks

With the motivation of obtaining accurate 3-D measurement of circular objects, we presented a new method for automatically detecting ellipses in images: we detect an osculating circle using a Hough transform, iteratively improve the fit, remove outlier pixels by LMedS, and search for separate arcs. We limited the Hough space to be one and two dimensions for efficiency and introduced special weighting schemes for enhancing accuracy. Using real images, we demonstrated that our method can detect partially occluded circular objects within a reasonable time. We also showed an example of turntable calibration for 3-D object shape reconstruction.

Our method relies on numerous empirically set parameters. This is inevitable in a sense: since it is impossible to cope with all possible scenes, some kind of tuning is unavoidable. In this respect, there is still room for further improvement.

In our real image experiments, we detected only one ellipse per image. If multiple ellipses exist in one image, our algorithm finds them one by one from larger to smaller, eliminating detected ones and searching for the remaining ones (our algorithm gives precedence to larger ellipses). However, our algorithm sometimes fails to distinguish two partly occluded concentric ellipses separated at a very short distance. Also, it is very difficult to find ellipses of which only tiny fractions are visible. These limitations seem inevitable as long as only edge segments are used. In order to resolve these, we need some contextual information about the scene.

Acknowledgments

The authors thank Mitsuo Okabe of Fujitsu Cadtech, Ltd., Japan, for real image experiments. This work was supported in part by the Ministry of Education, Culture, Sports, Science and Technology, Japan, under a Grant in Aid for Scientific Research C(2) (Nos. 14580404, 15500113), the Support Center for Advanced Telecommunications Technology Research, and Kayamori Foundation of Informational Science Advancement.

Appendix A. Canonical Form of a Quadratic Curve

The following is a brief summary of the canonical form of a quadratic curve (see Refs. 8 and 9 for the details).

Suppose Eq. (1), or Eq. (4) in matrix form, defines an ellipse. Using the matrix $\boldsymbol{S}$ and the vector $\boldsymbol{c}$ defined in Eqs. (5), we can write the matrix $\boldsymbol{Q}$ in Eqs. (3) as

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{S} & \boldsymbol{c} \\ \boldsymbol{c}^\top & F \end{pmatrix}. \tag{A.1}$$

Letting $\vec{x} = (x, y)^\top$, we can rewrite Eq. (4) in the form

$$(\vec{x}, \boldsymbol{S}\vec{x}) + 2f(\boldsymbol{c}, \vec{x}) + f^2 F = 0. \tag{A.2}$$

If the coordinate system is translated by $f\boldsymbol{S}^{-1}\boldsymbol{c}$, Eq. (A.2) is rewritten as

$$(\vec{x}', \boldsymbol{S}\vec{x}') = f^2 c, \tag{A.3}$$

where we let $\vec{x}' = \vec{x} - f\boldsymbol{S}^{-1}\boldsymbol{c}$ and c is the scalar defined in Eqs. (5). Let $\{\lambda_1, \lambda_2\}$ be the eigenvalues of $\boldsymbol{S}$, and $\boldsymbol{U}$ the orthogonal matrix having the corresponding unit eigenvectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ as its columns. Then,

$$\boldsymbol{S} = \boldsymbol{U} \begin{pmatrix} \lambda_1 & \\ & \lambda_2 \end{pmatrix} \boldsymbol{U}^\top. \tag{A.4}$$

If the coordinate system is rotated by $\boldsymbol{U}$, Eq. (A.2) has the form

$$\lambda_1 x''^2 + \lambda_1 y''^2 = f^2 c, \tag{A.5}$$

where we let $(x'', y'')^\top = \boldsymbol{U}^\top \vec{x}'$. Eq. (A.5) reduces to the following *canonical form*:

$$\frac{x''^2}{a^2} + \frac{y''^2}{b^2} = 1, \qquad a = f\sqrt{\frac{c}{\lambda_1}}, \quad b = f\sqrt{\frac{c}{\lambda_2}}. \tag{A.6}$$

The ring region of width δ around the ellipse of Eq. (1) can be defined if we let $a \leftarrow a \pm \delta$ and $b \leftarrow b \pm \delta$ and reconstruct the matrix $\boldsymbol{Q}^{(\pm)}$ by tracing back the above derivation.

From this analysis, we can see that Eq. (1) defines an ellipse if and only if λ_1, λ_2, and c have the same sign. Suppose the sign of the matrix $\boldsymbol{Q}$ is so chosen that $\det \boldsymbol{Q} < 0$. It is easily seen that λ_1, λ_2, and c have the same sign if and only if $AC - B^2 > 0$ and $A + C > 0$. If $AC - B^2 > 0$ but $A + C < 0$, no real points satisfy Eq. (1), so the condition $A + C > 0$ need not be considered as long as a real curve is observed. If $AC - B^2 = 0$, Eq. (1) defines a parabola; if $AC - B^2 < 0$, it defines a hyperbola. If $\det \boldsymbol{Q} = 0$, Eq. (1) defines two lines or their degeneracies (one double line, one point, or an empty set).

Appendix B. 3-D Interpretation of an Ellipse

The following is a well known procedure for computing the position and orientation of a circle in the scene from its projection image (see Refs. 8 and 9 for the details).

Let the constant f in Eqs. (1) and (3) be the focal length (in pixels) of the camera. We define an image xy coordinate system such that the origin is at the principal point (the point through which the optical axis passes). Taking the x-axis pointing up and the y-axis pointing right, we define the z-axis in the depth orientation. Let r be the radius of the turntable, and $\boldsymbol{Q}$ its observed image. In order to remove the scale indeterminacy of $\boldsymbol{Q}$, we normalize it to $\det \boldsymbol{Q} = -1$ (if $\det \boldsymbol{Q} = 0$, we have non-elliptic figures as mentioned in Appendix A).

Let λ_1, λ_2, and λ_3 be the eigenvalues of $\boldsymbol{Q}$, and $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3\}$ the orthonormal system of the corresponding eigenvectors. The eigenvalues are ordered in such a way that $\lambda_3 < 0 < \lambda_1 \leq \lambda_2$. The unit normal to the surface of the turntable is computed as follows:

$$\boldsymbol{n} = N[\sqrt{\frac{\lambda_2 - \lambda_1}{\lambda_2 - \lambda_3}}\boldsymbol{u}_2 \pm \sqrt{\frac{\lambda_1 - \lambda_3}{\lambda_2 - \lambda_3}}\boldsymbol{u}_3]. \tag{B.1}$$

Here, $N[\,\cdot\,]$ designates normalization to a unit vector. We obtain two solutions, because looking down on the turntable and looking up at it would result in the same ellipse image. In the usual setup, we pick out the looking-down solution, so we select the sign for which $n_1 n_3 \leq 0$. The orthogonal distance of the turntable surface from the lens center is given by

$$d = \sqrt{\lambda_1^3}r. \tag{B.2}$$

The center of the turntable is projected onto the image at the position

$$\boldsymbol{x}_C = Z[\boldsymbol{Q}^{-1}\boldsymbol{n}], \tag{B.3}$$

where $Z[\,\cdot\,]$ designates normalization to make the z component 1. The 3-D position of the center of the turntable is

$$\boldsymbol{r}_C = \frac{d\boldsymbol{x}_C}{(\boldsymbol{n}, \boldsymbol{x}_C)} \tag{B.4}$$

with respect to the camera coordinate system. The angle between the optical axis of the camera and the rotation axis of the turntable is give by

$$\theta = \cos^{-1} |n_3|. \tag{B.5}$$

References

1. Y. Asayama and M. Shiono, in *Proc. IAPR Workshop on Machine Vision Applications*, Nov. 1998, Makuhari, Chiba, Japan, pp. 494.
2. J. W. Bruce and P. J. Giblin, *Curves and Singularities*, (Cambridge University Press, Cambridge, 1984).
3. Y. C. Cheng and S. C. Lee, *Patt. Recog.*, **28**, 663 (1995).
4. W. Chojnacki, M. J. Brooks and A. van den Hengel, *J. Math. Imaging Vision*, **14**, 21 (2001).
5. W. Chojnacki, M. J. Brooks, A. van den Hengel and D. Gawley, *IEEE Trans. Patt. Anal. Mach. Intell.*, **22**, 1294 (2000).
6. D. B. Cooper and N. Yalabik, *IEEE Trans. Comp.*, **25**, 1020 (1976).
7. D. Ioannou, W. Huda and A. F. Laine, *Image Vision Comput.*, **17**, 15 (1999).
8. K. Kanatani, *Geometric Computation for Machine Vision*, (Oxford University Press, Oxford, 1993).
9. K. Kanatani, *Statistical Optimization for Geometric Computation: Theory and Practice*, (Elsevier, Amsterdam, 1996).
10. N. Guil and E. L. Zapata, *Patt. Recog.*, **30**, 1729 (1997).
11. C.-T. Ho and L.-H. Chen, *Patt. Recog.*, **28**, 117 (1995).
12. C. L. Huang, *Patt. Recog. Lett.*, **10**, 93 (1989).
13. Y. Leedan and P. Meer, *Int. J. Comput. Vision*, **37**, 127 (2000).
14. B. Matei and P. Meer, in *Proc. 15th Int. Conf. Patt. Recog.*, Sept. 2000, Barcelona, Spain, Vol. 3, pp. 801.
15. D. Pao, H. F. Li and R. Jayakumar, *Patt. Recog.*, **14**, 951 (1993).
16. P. L. Rosin and G. A. W. West, *IEEE Trans. Patt. Anal. Mach. Intell.*, **17**, 1140 (1995).
17. G. Roth and M. D. Levine, *CVGIP: Image Understand.*, **58**, 1 (1993).
18. G. Roth and M. D. Levine, *IEEE Trans. Patt. Anal. Mach. Intell.*, **16**, 901 (1994).
19. C. A. Rothwell, A. Zisserman, C. I. Marinos, D. A. Forsyth and J. L. Mundy, *Image Vision Comput.*, **10**, 250 (1992).
20. P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*, (Wiley, New York, 1987).
21. S. Tsuji and F. Matsumoto, *IEEE Trans. Comp.*, **27**, 777 (1978).
22. W.-Y. Wu and M.-J. J. Wang, *Patt. Recog.*, **26**, 1449 (1993).
23. H. K. Yuen, J. Illingworth and J. Kittler, *Image Vision Comput.*, **7**, 31 (1989).
24. J. H. Yoo and I. K. Seth, *Patt. Recog.*, **26**, 307 (1993).

Kenichi Kanatani received his M.S. and Ph.D. in applied mathematics from the University of Tokyo in 1974 and 1979, respectively. After serving as Professor of computer science at Gunma University, Gunma, Japan, he is currently Professor of information technology at Okayama University, Okayama, Japan. He is the author of *Group-Theoretical Methods in Image Understanding* (Springer, 1990), *Geometric Computation for Machine Vision* (Oxford, 1993) and *Statistical Optimization for Geometric Computation: Theory and Practice* (Elsevier, 1996). He is an IEEE Fellow.

Naoya Ohta received his ME degree in information science from the Tokyo Institute of Technology in 1985 and his Ph.D. in applied mathematics from the University of Tokyo in 1998. He engaged in research and development of image processing systems at the Pattern Recognition Research Laboratories of NEC, Japan. He was a research affiliate of the Media Laboratory in MIT, MA, U.S.A., from 1991 to 1992. He is currently Associate Professor of computer science at Gunma University, Gunma, Japan. His research interests include image processing and computer vision.