Computational Geometry 48 (2015) 180–188

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Curvature based shape detection

Tijana Šukilović

Faculty of Mathematics, University of Belgrade, Studentski Trg 16, 11001, Belgrade, Serbia

A R T I C L E I N F O

Article history:
Received 13 February 2013
Accepted 25 September 2014
Available online 2 October 2014

Keywords:
Curve detection
Shape recognition
Discrete curvature
Curvature fitting

A B S T R A C T

In this paper is defined a notion of discrete curvature associated with a discrete piecewise-smooth curve. Since every planar curve is, up to the orientation preserving isometry, uniquely determined by its curvature, shape recognition is reduced to fitting estimated discrete curvature function. Based on shape attributes, measures of similarity between two objects are introduced and used for full classification of detected geometric shapes in binary edge image.

1. Introduction

The problem of detecting geometric shapes such as lines, circles, circular arcs, ellipses etc. is a central problem in pattern recognition, computer vision, and robotics. Our ultimate goal in this direction is recognition of arbitrary geometric shapes.

In this paper we assume that the given raster image is black and white, i.e. the shapes are black pixels on white background. We define a notion of discrete curvature and apply it to the detection of piecewise-smooth curves.

The paper is organized as follows. In Section 2 we present related approaches addressing shape recognition problem. In Section 3 we recall some basic definitions and properties of plane curves.

In Section 4 we define a notion of discrete curvature associated with black pixel. It is analogous to the curvature of smooth plane curve. The curvature is a local property of smooth curve: it depends only on the shape of the curve near the feature point. Similarly, we use only the neighboring pixels to calculate the discrete curvature of the feature black pixel.

The process of assigning discrete curvature function to the discrete curve is presented in Section 5. We explain in detail how we trace points along the curve and calculate the corresponding curvature.

Section 6 is dedicated to the methods of discrete curvature matching and object recognition. Some measures of similarity between two objects based on their shape attributes are introduced and later used for distinguishing objects of different types. In addition, we can assess the attributes of similarity for objects from the same class, i.e. scaling factors and angles of rotation.

We discuss error estimations in Section 7. Calculation errors originate from both the discrete nature of image space and the numerical computations and we investigate the impact of each one on the outcome of the detection process.

Finally, some experimental results and conclusions are given in Section 8.

E-mail address: tijana@matf.bg.ac.rs.

http://dx.doi.org/10.1016/j.comgeo.2014.09.005
0925-7721/© 2014 Elsevier B.V. All rights reserved.

2. Related work

Hough Transform (HT) is one of the most widely used algorithms for line detection [1,2]. General Hough Transform (GHT) represents a modification of HT which is used to detect circles and even arbitrary curves [3]. The idea is that each black pixel in the image space votes in parameter space and the votes are accumulated in Hough space (HS) of parameters.

GHT is a very robust algorithm but it has some serious issues. The main problem in line detection using HT is line thickness, but recent work solves that problem efficiently [4].

Drawbacks of GHT for curve detection include high memory requirements, line thickness, and local maximum detection in the parameter space. Use of GHT for detection of more complex shapes is very limited, because of the large number of parameters which require high amounts of memory. Ellipse, for example, is determined by five parameters in general case (coordinates of center, size of two axes and rotation), hence, in principle, requires a 5-dimensional parameter array. The dimension of parameter space can be reduced by using some additional information [3,5], but the computational load is still high and some other difficulties arise, especially in arc detection process.

Besides Hough-based methods for shape recognition, there are several different approaches for solving this problem.

The first approach consists of converting the object to a set of idealized thin lines called the skeleton or medial axis. In this way the thinnest representation of the original object that preserves the homotopy aiding synthesis and understanding is obtained. In order to get structural description that captures the topological information embedded in the skeleton, one must detect end points, junction points and curve points of medial axis. The thin lines can be converted into a graph associating the curve points with the edges, the end and the junction points with the vertices. Such a skeletal graph can then be used as an input to graph matching algorithms (see [6–9]).

Shapes have also been represented by their outline curves. Matching typically involves finding a mapping from one curve to the other that minimizes an "elastic" performance functional, which penalizes "stretching" and "bending" [10,11]. The curve-based methods in general suffer from one or more of the following drawbacks: asymmetric treatment of the two curves, lack of rotation and scaling invariance, and sensitivity to articulations and deformations of parts.

The type of representation used in describing a shape can have a significant impact on the effectiveness of the recognition strategy. The comparative analysis of different shape representations can be found in [12].

Other approaches include approximation by special types of curves such as B-spline curves [13], Bézier curves [14], Hilbert curve [15], etc.

Previous approaches are mostly applied to general shape detection. For the detection of parameterized curves, the generalized Radon transform could be used [16].

The idea of curvature based shape representation for planar curves was introduced in [17,18]. There are many methods for estimating the curvature at point P (see [19–22]), majority of them use approximation and curvature fitting. Invariant descriptors under the affine, similarity and projective transformations were studied in [23]. Here, we use the similar approach.

3. Curvature of planar curves

Consider a parameterized curve $\alpha : (a, b) \to \mathbb{R}^2$ parameterized by its arc length s, i.e. distance along the curve measured from some fixed point on the curve:

$$s(t) = \int_c^t \|\alpha'(u)\| du, \quad c \le t \le b.$$

Then $\alpha'(s)$ is its tangent vector $T(s)$ and $\alpha''(s)$ is collinear to the unit normal vector $N(s)$. Therefore we have

$$\alpha''(s) = k(s)N(s), \tag{1}$$

where the function $k(s)$ is called **curvature** of the curve α at point $\alpha(s)$.

Our opinion is that the curvature approach can be used for detecting arbitrary shapes due to the fundamental theorem for plane curves (for mathematical details see [24]). The importance of the theorem is that the shape of the curve is determined only by its (signed) curvature, regardless of the position of the curve in the plane, i.e. translation and rotation.

To address the problem of scaling (i.e. detecting the same shape of different size) we suggest the following. Suppose that curve α_λ is obtained from curve $\alpha = \alpha(s)$, $s \in (0, b)$ by scaling with coefficient λ. Then the curvature of α_λ is scaled by $\frac{1}{\lambda}$ and arc length is scaled by λ, so we have

$$\bar{k}(\lambda s) = \frac{1}{\lambda} k(s), \quad s \in (0, b), \tag{2}$$

where $\bar{k}$ denotes the signed curvature of the curve α_λ. This allows us to easily match scaled shapes (details are found in Section 6).

The **turning number** of a closed curve $\alpha : \mathbb{R} \to \mathbb{R}^2$ is

$$\tau(\alpha) = \frac{1}{2\pi} \int_a^{a+c} k(t)\left\|\alpha'(t)\right\|dt,$$

where k denotes the curvature and c denotes the period of α. Although intuitively clear, the proof of the following theorem is due to H. Hopf.

Theorem 1. *(See [25].) The turning number of a simple closed plane curve α is ± 1.*

In the previous theorem the result $+1$ or -1 depends on the orientation of the curve. This means that by observing the curvature we are able to detect all simple closed curves. Additionally, we can determine if the curve is convex or not. Let us recall that a simple close regular plane curve α is convex if and only if its curvature k has constant sign; that is, k is either always nonpositive or always nonnegative.

4. Discrete curvature associated with a black pixel

Suppose that $P = \alpha(0)$ is a black pixel of the discrete curve, and $Q_i = \alpha(s_i)$, $i = 1, \dots, n$ are black pixels of the curve in the neighborhood of the point P. If P is not close to the end of the curve, we assume that pixels Q_i are **balanced**. Here balanced means that the number of pixels Q_i at "one side" of P (positive parameters s_i) is similar to the number of pixels Q_i at the other side of P (negative parameters s_i).

The curve α can be developed by the Taylor series in the neighborhood of point $\alpha(0)$

$$\alpha(s) = \alpha(0) + s\alpha'(0) + \frac{s^2}{2}\alpha''(0) + o\left(s^2\right). \tag{3}$$

We use this formula to get an approximation of curvature for discrete curves.

From formulas (1) and (3), for each Q_i we have

$$\overrightarrow{PQ_i} = \alpha(s_i) - \alpha(0) \approx s_i T(0) + \frac{s_i^2}{2} k(P) N(0),$$

where we denote $k(P) = k(0)$. Summing up for all i, since the points Q_i are balanced, the parameters s_i more or less cancel and we get

$$\sum_{i=1}^{n} \overrightarrow{PQ_i} \approx \frac{k(P)N(0)}{2} \sum_{i=1}^{n} s_i^2 = \frac{k(P)N(0)}{2} \sum_{i=1}^{n} \|\overrightarrow{PQ_i}\|^2. \tag{4}$$

We get an approximation of the curvature $k(P)$ of the curve α:

$$k(P) \approx \frac{2\|\sum_{i=1}^{n} \overrightarrow{PQ_i}\|}{\sum_{i=1}^{n} \|\overrightarrow{PQ_i}\|^2}. \tag{5}$$

In this way we can associate discrete curvature $k(P)$ to each black pixel on raster image.

Note that from Eq. (4) we can calculate (an approximation of) the unit normal vector $N(P) = N(0)$ of the curve α in point P. That normal is collinear to vector $\sum_{i=1}^{n} \overrightarrow{PQ_i}$.

Consider an arc ℓ and let $S = \{P_i = \alpha(s_i) \mid i = 1, \dots, n\}$ be the sampling of ℓ. Denote by δ the maximum arc-length between the sampling points and by K_i and K_i' respectively the maximum of the curvature and its derivative around the point P_i. The curvature estimator (5) is said to be convergent if

$$k(S) = \sum_{i=1}^{n} k(P_i) \longrightarrow \int_\ell k(s)ds, \quad \text{when } \delta \to 0.$$

Thus the conditions $K_i\delta \le \epsilon$, $K_i'\delta \le \epsilon$ must be satisfied for some small $\epsilon > 0$. This corresponds to the dense sampling in regions where curvature is high or it is changing rapidly.

5. Assigning discrete curvature function to the discrete curve

We assume that the image consists of straight lines or other curved lines, but without filled (black) regions (e.g. achieved by edge detection). We associate discrete curvature function to a curve α using the following algorithm.

We start from a "corrected" black pixel $P = P_0$ on the image which belongs to some curve α and incrementally trace some points along the curve α in such way that positive orientation is preserved. Denote them by $P_0, P_1, \dots, P_n$ with

$P_{n+1} = P_0$ if the curve is closed. All points P_i are processed by pixel correction method before further use. The reason we use the corrected pixel is that we seek for the best representative of curve α in neighborhood of P. This gives us much better results in estimating the curvature of the curve and an additional information about the curve thickness. Also, it makes process of sampling points more efficient. For each P_i we calculate discrete curvature $k_i = k(P_i)$ using formula (5). When all traced points are processed, depending on the outcome, we begin curve verification.

1. If the end of the curve was reached, we have detected an open curve. Information about start and end points is obtained.
2. If we have found a possible cross-section, the curve is either self-crossing or multiple curves are intersecting in that point. At this stage, we are not interested in this kind of cases, but further analysis can be performed once the detection process is over.
3. If we were to return to the initial point, depending on the value of the turning number, we have detected a simple closed curve ($\tau(\alpha) = 1$) or the curve is not verified. In the discrete case, the turning number of curve α is calculated by the formula

$$\tau(\alpha) = \frac{1}{2\pi}\sum_{i=0}^{n}\frac{1}{2}\big(k(P_{i+1}) + k(P_i)\big)\|P_{i+1} - P_i\|. \qquad (6)$$

Once we verify all detected objects/curves, we can proceed to further analysis. We can mark all open/closed curves, classify all objects, isolate only convex ones etc.

6. Curve analysis and curvature matching

After detecting all the objects in the image, it is natural to compute the measure of similarity between two objects based on their shape attributes.

If we fix some length L and scale all objects using (2) to have the same length L, as explained in Section 3, the curve will be uniquely determined only by its curvature. In this way, the shape detection reduces to discrete function fitting. The similar approach for comparing polygonal shapes was used by Arkin et al. [26].

The central question here is: given two sets of points in the plane, $P = \{(s_i, k_i),\ i = 1, \ldots, n_1\}$ and $Q = \{(s'_j, k'_j),\ j = 1, \ldots, n_2\}$, where k_i and k'_j are measured curvatures, and some $\delta > 0$, find a translation T such that $d(T(P), Q) < \delta$, where $d(\cdot,\cdot)$ is some distance measure. The translation is the only similarity transformation we are considering because the curvature stays unchanged under translation, rotation and scaling (since we re-scale all curves to have the same length) and the only difference comes from the point where we start measuring arc length. This is actually a special case of a matching problem already discussed in [27–29].

The Hausdorff distance (HD) between two point sets P and Q is defined as

$$H(P, Q) = \max\big(h(P, Q), h(Q, P)\big)$$

where $h(P, Q)$ is the directional Hausdorff distance from P to Q:

$$h(P, Q) = \max_{p\in P}\min_{q\in Q} d(p, q).$$

Here, $d(\cdot,\cdot)$ represents a more familiar metric on points; for instance, the standard Euclidean metric L_2 or the L_∞ metric. In [26] was presented the advantage of using metric L_2 instead of L_1.

Different distance metrics used for point sets and their use in object matching are discussed in [30,31]. Due to the specific arrangement of our points, we propose to use different distance measure:

$$D(P, Q) = \frac{1}{2}\left(\frac{1}{N_P}\sum_{p\in P}\min_{q\in Q} d(p, q) + \frac{1}{N_Q}\sum_{q\in Q}\min_{p\in P} d(p, q)\right),$$

where N_P and N_Q represent the number of elements in P and Q respectively. Note that, although $D(P, Q)$ is not a metric, it has very desirable behavior for object matching purposes.

We define the δ-neighborhood of a point q in the plane to be set of all points in $\mathbb{R}^2$ that are at distance $\le \delta$ from q.

Our goal is to find translation T given by the formula $x' = x + c,\ y' = y$, that brings every point in P to the δ-neighborhood of Q. If such a translation exists, the two objects match. Otherwise, a new object is found.

Additionally, if we match a pair of objects, we also get a complete information about similarity transformation between them, i.e. the scaling factor and the angle of rotation. For every two curves the corresponding scaling factor is the ratio between the scaling factors obtained by re-scaling them to length L. Rotation is a little bit more complicated.

Let α and β be two matched curves and c is the result of the optimal translation. If $\alpha(s_0)$ and $\beta(s_1)$ were the points we started measuring the length of curves α and β respectively, then the angle of rotation is the angle between the unit normal vectors calculated by formula (4) in points $\alpha(s_0)$ and $\beta(s_1 + c)$.

7. Error estimation

Our detection method depends heavily on rasterization of the curve. A natural question arises: how much does the discrete representation influence a correct curvature estimation?

Since every curve α at point $\alpha(t)$ is best approximated by an osculating circle, i.e. the circle of radius and center

$$r(t) = \frac{1}{|k(t)|}, \qquad C(t) = \alpha(t) + \frac{1}{k(t)} N(t),$$

we will discuss in detail error estimation for circles only.

For an arbitrary point p_0 on the circle, in order to calculate radius of the circle, we need to find two distinct points p_1 and p_2. If these three points are not collinear, then radius of the circle is calculated by the formula:

$$r = \frac{\|p_0 - p_1\| \cdot \|p_1 - p_2\| \cdot \|p_2 - p_0\|}{2P_\triangle}, \tag{7}$$

where $\|\cdot\|$ denotes the norm of the vector and $P_\triangle$ denotes the area of triangle $p_0p_1p_2$. The curvature of a circle of radius r is $k(s) \equiv \frac{1}{r}$.

It is obvious that the largest error in radius/curvature estimation will occur if the chosen points are very near, thus almost collinear. Hence, the question is how to find an appropriate measure to estimate what is *far enough*.

Theorem 2. *The longest black run of collinear pixels in a discretization of circle of radius r is less than*

a) $\sqrt{r}$ *(continuous approximation).*
b) $1 + 2\sqrt{r-2}$ *(Bresenhem's rasterization).*

Proof. Let us consider the circle centered at the origin of radius r (measured in pixels) and find the longest line through it contained in the circle. Since the result is invariant under rotation, we can consider the point $P = (r, 0)$ and the corresponding longest line will be the vertical line $x = r$.

a) By developing the circle in the Taylor series in a neighborhood of point P, we obtain that length s of the vertical line through p cannot be longer than $\sqrt{r}$.
b) Analyzing Bresenhem's algorithm [32], we see that error calculation for the pixel (x, y) is

$$err_{x,y} = x^2 + y^2 - r^2$$

and for the next vertical pixel

$$err_{x,y+1} = x^2 + (y+1)^2 - r^2 = err_{x,y} + (2y+1)$$

Summing up all errors, we get a different approximation for the length of the longest line containing pixel $P : s \leq 1 + 2\sqrt{r-2}$. Since the radius here is measured in pixels, condition $r \geq 2$ is not essential. $\square$

Specially, we can use this estimation to make some modifications in the circle detection process. To suppress accumulating errors, an estimation proposed in [33] can be used. Also, we can take into an account the fact that if X and Y are independent random variables with normal (Gaussian) distribution $N(0, \sigma^2)$, then variable $R \sim \sqrt{X^2 + Y^2}$ has Rayleigh distribution $R(\sigma)$ (for details see [34]).

Now we have the minimal distance required for accurate radius calculation. Performing numerous test, we concluded that the maximal experimental error for measuring the discrete curvature in an arbitrary point did not exceed 20%.

The other question that arises is how we can use the error of the calculated turning number τ_α in order to estimate error for matching closed curves. Is there any correlation? Using formula (6) we get an estimation for the turning number calculation error:

$$err_\tau = |\tau_\alpha - 1| \leq \frac{L}{2\pi} err^{max} = \frac{L}{2\pi} \max_i \big|k(s_i) - k_i\big|, \quad i = 0, \dots, n, \tag{8}$$

where L is the length of the curve α, $k(s_i)$ is the true curvature value in point $P = \alpha(s_i)$ and k_i is detected curvature in that point.

With the previous error estimation, we can proceed to discrete curvature fitting. We presume that two curves α and β belong to the same class if the error of matching process is:

$$err_{match} \leq \sqrt{n_\alpha(\tau_\alpha - 1)^2 + n_\beta(\tau_\beta - 1)^2}, \tag{9}$$

where n_α and n_β are numbers of sample points, and τ_α and τ_β calculated turning numbers of curves α and β, respectively.

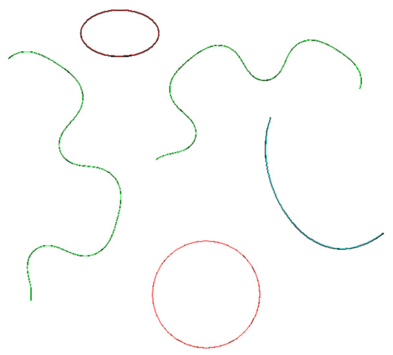

Fig. 1. Open and closed curves.

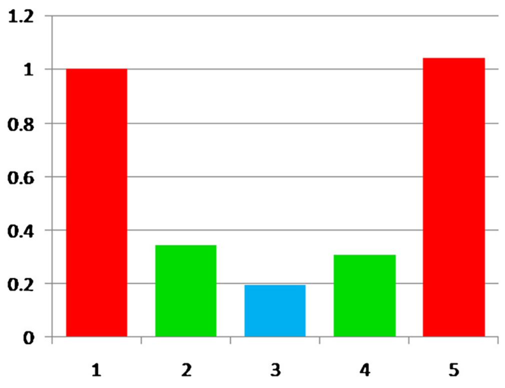

Fig. 2. Turning numbers for closed and open curves.

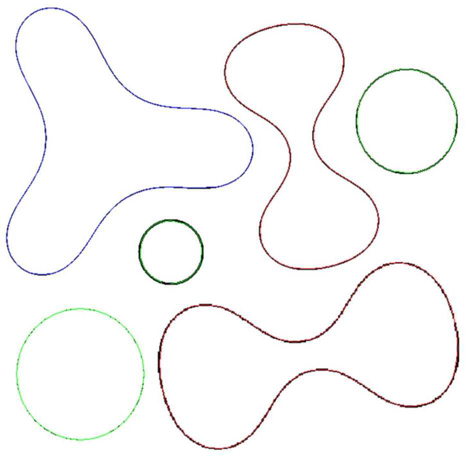

Fig. 3. Test image 1.

8. Experimental results

The main computational complexity of the algorithm is tracing points along the curve. Therefore, the execution speed depends heavily on the resolution. The pixel correction method plays the crucial part in making the sampling process more efficient, especially in cases when the thick lines are handled. In our implementation, it takes less than 2*s* to trace approximately 500 points, calculate the curvature and verify the detected curve. In the following examples, this procedure is repeated several times. Total time for detecting all of the objects and performing the complete classification is less than a minute.

In Fig. 1 we have drawn three open and two closed curves – a circle, an ellipse, a parabolic arc and two copies of an arbitrary curve. Previously determined error boundary (8) appears to be a good criterion for differentiating between open

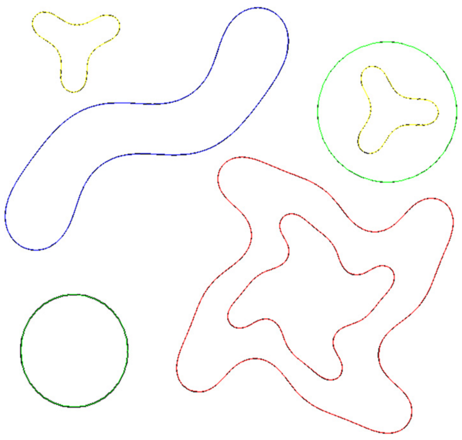
Fig. 4. Test image 2.

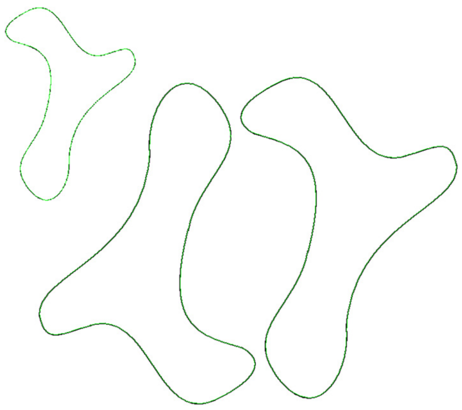
Fig. 5. Test image 3.

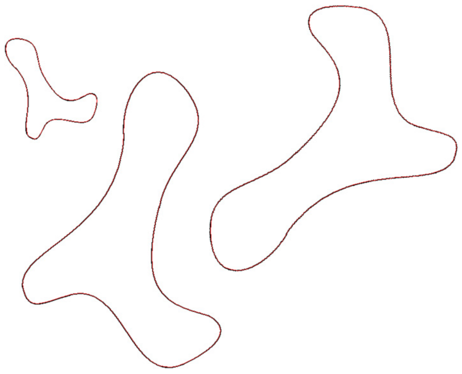
Fig. 6. Test image 4.

and closed curves. In Fig. 2 columns representing the turning number are colored in the same way as the corresponding curve.

We used (9) to determine whether the curves α and β belong to the same class or not. In our test images (see Figs. 1, 3–6) the matched curves are colored with the same color. Due to the discretization error, as seen in Fig. 1, our program fails to distinct between circles and ellipses without some additional analysis.

Furthermore, we can observe the impact of the similarity transformation on the detection outcome. In Figs. 5 and 6 we have transformed the middle curve – the right curve is just rotated, while the left one is both rotated and scaled. Results of true and detected parameters are shown in Table 1. Comparisons are made with respect to the middle curve, since that curve stays unchanged in both test images.

Table 1
Error estimation under similarity transformation.

	Left			Right		
	True	Detected	Error	True	Detected	Error
Fig. 5						
Scaling factor	0.6	0.6003	0.05%	1.0	0.9991	0.09%
Rotation angle	180	179.0800	0.51%	180	179.5550	0.25%
Fig. 6						
Scaling factor	0.4	0.3826	4.35%	1.0	1.0017	0.17%
Rotation angle	22	23.2543	5.70%	170	171.7510	1.03%

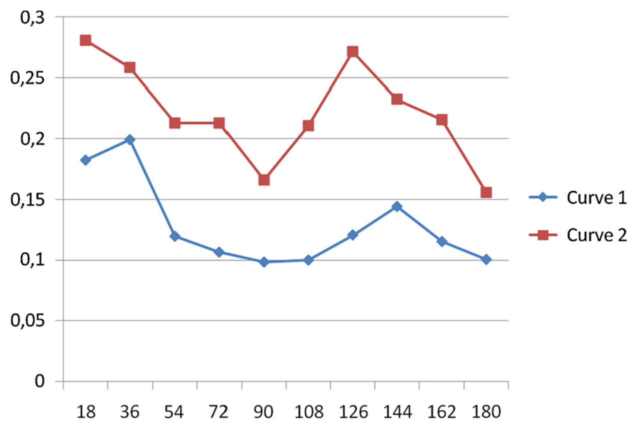

Fig. 7. The matching error of curves under the rotation.

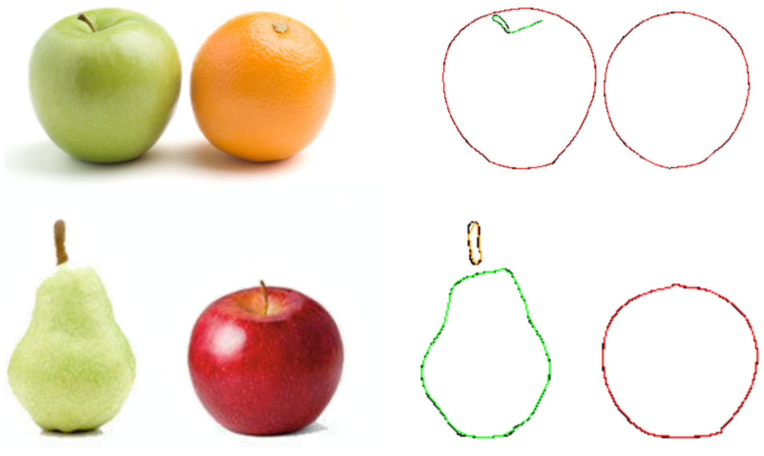

Fig. 8. Matching shapes extracted from the real images.

It is obvious that errors in estimation of the similarity transformation parameters are higher in Fig. 6 than in Fig. 5. The reason for this is purely technical – in the first image we rotated the curves by 180° and the rotation matrix is an integer matrix (similar is for angles 90° and 270°), while for the second one the angles are not so nice.

Fig. 7 shows the influence of the pixelization of the curves on the matching process. In order to suppress all other errors, we took two symmetric curves, and measured the matching error of the curves under the rotation for angle $n \cdot 18°$, $n = 0, \dots, 10$. The first one was convex (the curvature has constant sign) and the second one was not.

The main drawback of the algorithm is sensitivity to noise, so the result of matching objects extracted from the real images will depend on the quality of edge detection and noise removal process. In the final example, we can see that the shape of an apple match the shape of an orange, but not the shape of a pear (see Fig. 8).

8.1. Conclusion

The proposed method showed some real strengths in real time applications. Rough classification of geometrical shapes of the same type is performed well and by further analysis can be refined to distinguish between similar objects like circles and ellipses, for example. Also, for the curves from the same class, we obtained information about length, line thickness and similarity parameters (scaling factors and rotation angles).

Further modifications of the algorithm can address questions of cross-sections between objects.

Due to its ability to successfully handle line thickness, the proposed method might be generalized to the case of curve approximation of point clouds.

Acknowledgements

This research is partially supported within the project 174012 through the Ministry of Education and Science of the Republic of Serbia.

References

[1] P.V.C. Hough, Method and means for recognizing complex patterns, U.S. Patent 3069654, 1962.
[2] J. Illingworth, J. Kittler, A survey of the Hough transform, Comput. Vis. Graph. Image Process. 44 (1988) 87–166.
[3] D.H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, Pattern Recognit. 13 (1981) 111–112.
[4] J. Song, M.R. Lyu, A Hough transform based line recognition method utilizing both parameter space and image space, Pattern Recognit. 38 (2005) 539–552.
[5] R.K.K. Yip, P.K.S. Tam, D.N.K. Leung, Modification of Hough transform for circles and ellipses detection using 2-dimensional array, Pattern Recognit. 25 (1992) 1007–1022.
[6] D.T. Lee, Medial axis transformation of a planar shape, IEEE Trans. Pattern Anal. Mach. Intell. 4 (1982) 363–369.
[7] L.G. Shapiro, R.M. Haralick, Decomposition of two-dimensional shapes by graph theoretic clustering, IEEE Trans. Pattern Anal. Mach. Intell. 1 (1) (1979) 10–20.
[8] M. Pelillo, K. Siddiki, S.W. Zucker, Matching hierarchical structures using association graphs, IEEE Trans. Pattern Anal. Mach. Intell. 21 (11) (1999) 1105–1119.
[9] C. Di Ruberto, Recognition of shapes by attributed skeletal graphs, Pattern Recognit. 37 (1) (2004) 21–31.
[10] R. Basri, L. Costa, D. Geiger, D. Jacobs, Determining the similarity of deformable shapes, Vis. Res. 38 (1998) 2365–2385.
[11] L. Younes, Computable elastic distance between shapes, SIAM J. Appl. Math. 58 (1998) 565–586.
[12] T.B. Sebastian, B.B. Kimia, Curves vs skeletons in object recognition, in: Proceedings of the 2001 International Conference on Image Processing, vol. 3, IEEE, 2001, pp. 22–25.
[13] F.S. Cohen, Z. Huang, Z. Yang, Curve recognition using b-spline representation, in: Proceedings of the IEEE Workshop on Applications of Computer Vision, IEEE, 1992, pp. 213–220.
[14] L. Shao, H. Zhou, Curve fitting with Bezier cubics, Graph. Models Image Process. 58 (3) (1996) 223–232.
[15] Y. Ebrahim, M. Ahmed, W. Abdelsalam, S.C. Chau, Shape representation and description using the Hilbert curve, Pattern Recognit. Lett. 30 (4) (2009) 348–358.
[16] M. van Ginkel, M. Kraaijveld, L. van Vliet, E. Reding, P. Verbeek, H. Lammers, Robust curve detection using a Radon transform in orientation space, in: Image Analysis, 2003, pp. 163–174.
[17] F. Mokhtarian, A.K. Mackworth, A theory for multiscale, curvature based shape representation for planar curves, IEEE Trans. Pattern Anal. Mach. Intell. 14 (1992) 789–805.
[18] E. Calabi, P.J. Olver, C. Shakiban, A. Tannenbaum, S. Haker, Differential and numerically invariant signature curves applied to object recognition, Int. J. Comput. Vis. 26 (2) (1998) 107–135.
[19] L.F. Estrozi, et al., 1D and 2D Fourier-based approaches to numeric curvature estimation and their comparative performance assessment, Digit. Signal Process. 13 (2003) 172–197.
[20] T. Lewiner, J. Gomes Jr., H. Lopes, M. Craizer, Arc-length based curvature estimator, in: Proceedings of Sibgrapi, 2004, pp. 250–257.
[21] M. Craizer, T. Lewiner, J.-M. Morvan, Combining points and tangents into parabolic polygons, J. Math. Imaging Vis. 29 (2–3) (2007) 131–140.
[22] D. Coeurjolly, S. Svensson, Estimation of curvature along curves with application to fibres in 3D images of paper, in: Lect. Notes Comput. Sci., vol. 2749, Springer, 2003, pp. 247–254.
[23] A.M. Bruckstein, A.N. Netravali, On differential invariants of planar curves and recognizing partially occluded planar shapes, Ann. Math. Artif. Intell. 13 (3–4) (1995) 227–250.
[24] A. Gray, Modern Differential Geometry of Curves and Surfaces with Mathematica, Chapman & Hall/CRC, 2006.
[25] H. Hopf, Über die Drehung der Tangenten und Sehnen ebener Kurven, Compos. Math. 2 (1935) 50–62.
[26] E.M. Arkin, L.P. Chew, D.P. Huttenlocher, K. Kedem, J.S. Mitchell, An efficiently computable metric for comparing polygonal shapes, IEEE Trans. Pattern Anal. Mach. Intell. 13 (3) (1991) 209–216.
[27] D. Aiger, K. Kedem, Exact and approximate geometric pattern matching for point sets in the plane under similarity transformations, in: Canadian Conference on Computational Geometry, 2007, pp. 181–184.
[28] D. Aiger, K. Kedem, Approximate input sensitive algorithms for point pattern matching, Pattern Recognit. 43 (1) (2010) 153–159.
[29] M. Benkert, J. Gudmundsson, D. Merrick, T. Wolle, Approximate one-to-one point pattern matching, J. Discrete Algorithms (2012) 1–15.
[30] M.P. Dubuisson, A.K. Jain, Modified Hausdorff distance for object matching, in: 12th International Conference on Pattern Recognition, vol. 1, 1994, pp. 566–568.
[31] T. Eiter, H. Mannila, Distance measures for point sets and their computation, Acta Inform. 34 (2) (1997) 109–133.
[32] J. Bresenham, A linear algorithm for incremental digital display of circular arcs, Commun. ACM 20 (2) (1977) 100–106.
[33] S. Utcke, Error-bounds on curvature estimation, in: Lect. Notes Comput. Sci., vol. 2695, 2003, pp. 657–666.
[34] A. Papoulis, S.U. Pillai, Probability, Random Variables and Stochastic Processes, fourth edition, McGraw-Hill, 2002.