

# Feature Extraction for Java Character Recognition

*by* Gregorius Satia Budhi

---

**Submission date:** 27-Nov-2019 02:37PM (UTC+0700)

**Submission ID:** 1222721109

**File name:** 11.\_Feature\_Extraction\_for\_Java\_Character\_Recognition.pdf (455.39K)

**Word count:** 3210

**Character count:** 15545

## Feature Extraction for Java Character Recognition

Rudy Adipranata<sup>1)</sup>, Liliana<sup>2)</sup>, Meiliana Indrawijaya<sup>3)</sup>, Gregorius Satia Budhi<sup>4)</sup>

Informatics Department, Petra Christian University

Siwalankerto 121-131, Surabaya, Indonesia

rudya@petra.ac.id<sup>1)</sup>, lilian@petra.ac.id<sup>2)</sup>,  
meiliindrawijaya@gmail.com<sup>3)</sup>, greg@petra.ac.id<sup>4)</sup>

**Abstract.** Feature extraction is very important in the process of character recognition. A good feature of the character will increase the level of accuracy for the character recognition. In this research, the feature extraction experiment of Java characters is done, where those features could be used for Java character recognition later. Before performing the process of feature extraction, segmentation is performed to get each Java character in an image, and followed skeletonization process. After skeletonization process, feature extraction is done including simple closed curve, straight lines and curve. Several experiments was done using various parameters and Java characters in order to obtain the optimal parameters. The experiment results of simple closed curve and straight line feature extraction are quite good, respectively reached 82.85% and 74.28%. However, the result of the curve detection is still not good, only reached 51.42%

**Keywords:** feature extraction, Java character, skeletonization

## 1 Introduction

One of the ethnic groups found in Indonesia is the Javanese with a culture that is known as the Javanese culture, which has many aspects. One is the use of Java characters on writing. To preserve the culture in the form of Java character, conservation efforts need to be done so that the culture is not extinct, one is to digitize documents bearing Java characters especially old documents that have high historical values. Java characters consist of basic characters, numbers, complementary characters and others. In this paper, we conduct research on feature extraction of Java characters. The features obtained later will be used for the Java character recognition in order to digitize the documents. These experiments of feature extractions need to be done in order to find different features for each Java character, because some of the Java characters have similar form to each other. Input to the system is an image that contains handwritten Java characters. Image segmentation is performed to separate the existing characters, and then we do the skeletonizing before feature extraction. The feature extraction was conducted on the number of lines, simple close curves and open curves.

## 2 Skeletonizing

Skeletonizing is one of image processing that is used to reduce the pixels of an image while maintaining information, characteristics and important pixels of the object. This is implemented by changing the initial image in binary into skeletal representation of the image. The purpose of skeletonizing is to make simpler image so that the image can be analyzed further in terms of shape and suitability for comparison with other images. Problems encountered while doing skeletonizing is how to determine the redundant pixels and to maintain important pixel. This process is more similar to erosion process where erosion can lead to an unexpected region deleted, which is not expected in the process of thinning. Skeleton must remain intact and have some basic properties such as [1]:

- Must be composed of several thin regions, with a width of 1 pixel.
- Pixels that form the skeleton should be near the middle of the cross section area of the region.
- Skeletal pixel must be connected to each other to form several regions which are equal in number to the original image.

There are two requirements that are used to determine whether a pixel can be removed or not. The first requirement is as follows [1]:

- A pixel can be removed only if it has more than one and less than 7 neighbors.
- A pixel can be removed only if it has the same one connectivity.
- A pixel can be removed only if at least one of the neighbors who are in the direction of 1, 3, or 5 is a background pixel.
- A pixel can be removed only if one of the neighbors who are in the direction of 3, 5, or 7 is a background pixel.

The second requirement is similar to the first but differ in the last two steps:

- A pixel can be removed only if at least one of the neighbors who are in the direction of 7, 1, or 3 is a background pixel.
- A pixel can be removed only if one of the neighbors who are in the direction of 1, 5, or 7 is a background pixel.

## 3 Simple Closed Curve

Simple closed curve is a simple curve that both endpoints coincide [2]. Examples of simple closed curve can be seen in Figure 1.



Fig. 1. Simple close curve

To detect a closed curve can be done by using the flood fill algorithm. Flood fill algorithm has three parameters, namely the start node, the target color and color replacement. This algorithm searches all nodes in the array are connected to the start node through the path of the target color and then replace it with a replacement color. Flood fill based algorithm by using recursion can be written as follows:

1. If the node is not the same color with the color of the target, the return
2. Set the color of nodes into a replacement color.
3. Run Flood-fill (one step to the west of the node).  
Run Flood-fill (one step to the east of the node).  
Run Flood-fill (one step to the north of the node).  
Run Flood-fill (one step to the south of the node).
4. Return.

#### 4 Hough Transform

Hough Transform is a technique for determining the location of a shape in the image. Hough Transform was first proposed by P.V.C Hough [3], and implemented to detect the lines in the image by Duda and Hart [4].

Hough Transform maps the points in the image into the parameter space (Hough Transform space) based on a function that defines the shape that wants to be detected, and then takes a vote on an array element called the accumulator array. Hough Transform is generally used to perform the extraction of lines, circles or ellipses in the image, but in its development, Hough Transform can also be used to the extraction of more complex shapes. Hough Transform is used to detect the straight lines that satisfy the Equation 1 and 2:

$$y = ax + b \quad (1)$$

$$b = -x_1 a + y_1 \quad (2)$$

By changing the Equation 1 to 2, each edge point (x, y) on an image will result in single line equation parameters (a, b). The points on the same line will have the value of the parameter that cross at a point (a, b) in the parameter space as shown in Figure 2.

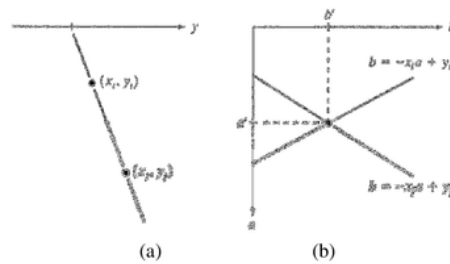


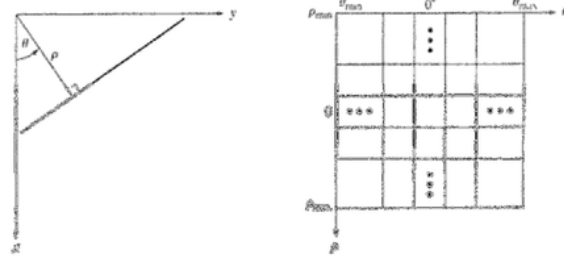
Fig. 2. (a) xy Area (b) Parameter space

At first, the value of the accumulator is initialized with zero. For each point  $(x, y)$  which is edge of the object in the image, the value of  $b$  will be calculated according to the Equation 2 and will be rounded to the nearest value that is allowed in the accumulator. Accumulator value will be incremented for each point that satisfies the equation with the values of  $a$  and  $b$  given in accordance with the Equation 3:

$$A(a, b) = A(a, b) + 1 \quad (3)$$

Each edge point has line parameter mapped in the accumulator. The higher the value in the accumulator, the greater the likelihood of a line is detected in the image. Duda and Hart proposed a polar equation for a line with a parameter  $\rho$  and orientation  $\theta$  [4] (Equation 4). Illustration can be seen in Figure 3.

$$\rho = x \cos \theta + y \sin \theta \quad (4)$$



**Fig. 3.** (a) Normal representation of a line. (b) Parameter space  $(\rho, \theta)$

In the same way with the standard Hough transform, each point in the image is mapped into the accumulator for each value  $\rho$  and  $\theta$  which satisfy the Equation 5.

$$A(\rho, \theta) = A(\rho, \theta) + 1 \quad (5)$$

The range of values for the angle  $\theta$  is  $\pm 90$  as measured by the  $x$ -axis. While the range of values of  $\rho$  is  $\pm\sqrt{2} D$ , where  $D$  is the distance between the vertex on the image [5].

In general, the Hough Transform method consists of three basic steps:

- Each pixel of an image is transformed into a curve parameter of the parameter space.
- Accumulator with cell arrays placed on the parameter space and each pixel images provide a value to the cells in the transformation curve.
- Pixel with a local maximum value is selected, and the coordinates of the parameters used to represent a segment of the curve in image space.

#### 4.1 Parabolic curve detection using Hough Transform

In the image of the actual object, the curve can be in any orientation. Parabolic curve with rotation can be detected by using an algorithm based on coordinate transformation of parabolic equations. In standard parabolic curve detection, there are four parameters involved, namely the point  $(x_0, y_0)$ , orientation  $(\theta)$ , and the coefficient which contains information about the parabolic curvature. However, Jafri and Deravi proposed an algorithm to detect parabolic curve in any orientation using only three parameters [6]. The parameters are the point  $(x_0, y_0)$  and orientation  $\theta$ . Using this algorithm, all parabolic curves in various positions can be detected by using 3D accumulator. This approach uses a point on the curve as a parameter which also shows the position of maximum curvature of the parabolic curve. Sobel operator is used for the gradient approach. To detect the parabolic curve in any orientation, a coordinate transformation matrix is used to derive a new parabolic equations involving parabolic curve orientation.

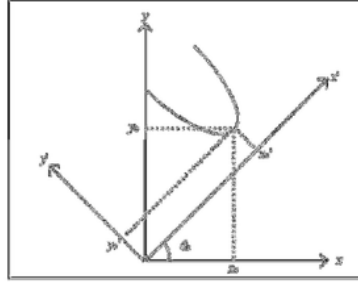


Fig. 4. Parabolic curve

Parabolic curve with a specific orientation angle is shown in Figure 4.  $(x', y')$  coordinates is the  $(x, y)$  coordinates rotation by  $\theta$  degrees with the center coordinate system as the axis of rotation. The vertex of parabola is  $(x'_0, y'_0)$  at the  $(x', y')$  coordinates or  $(x_0, y_0)$  in the  $(x, y)$  coordinates. The equation of the parabola in the  $(x', y')$  coordinates can be written in Equation 6 [6].

$$(y' - y'_0) = p (x' - x'_0)^2 \quad (6)$$

Standard two dimensional geometry matrix for counter-clockwise rotation with  $\theta$  angle transformation is shown in Equation 7.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (7)$$

By substituting the value of  $x, y, x_0$  and  $y_0$  in Equation 7 to Equation 6, the parabolic Equation 6 can be written as:

$$(-x \sin \theta + y \cos \theta) - (-x_0 \sin \theta + y_0 \cos \theta)$$

$$= p [(x \cos \theta + y \sin \theta) - (x_o \cos \theta + y_o \sin \theta)]^2 \quad (8)$$

And the value of differentiation of this equation is:

$$\begin{aligned} & -\sin \theta + \frac{dy}{dx} \cos \theta \\ & = 2p [(x \cos \theta + y \sin \theta) - (x_o \cos \theta + y_o \sin \theta)] \cdot \left[ \cos \theta + \frac{dy}{dx} \sin \theta \right] \end{aligned} \quad (9)$$

By substituting Equation 9 into 8, a new relation to the parabola vertex and the orientation  $(x_o, y_o, \theta)$  is [6]:

$$\begin{aligned} y_o &= \left[ \frac{k_1(x \cos \theta + y \sin \theta) + (x \sin \theta - y \cos \theta)}{k_1(\sin \theta - \cos \theta)} \right] \\ & \quad \frac{(k_1 \cos \theta + \sin \theta)}{(k_1 \sin \theta - \cos \theta)} x_o \end{aligned} \quad (10)$$

where  $k_1$  is

$$k_1 = \frac{-\sin \theta + \frac{dy}{dx} \cos \theta}{2(\cos \theta + \frac{dy}{dx} \sin \theta)} x_o \quad (11)$$

From the above relationship, it can be seen that the use of three dimensional accumulator arrays is sufficient to detect a parabola in various orientations.

## 5 Experimental Results

Several types of experiments has been performed, such as simple close curve detection, line detection and curve detection on several sample scanned document images of Java characters. The overall process can be seen in Figure 5. At first, system will accept input images that contain Java characters. Then segmentation process will be done to separate each character. For each character, skeletonization process will be done before detection process.

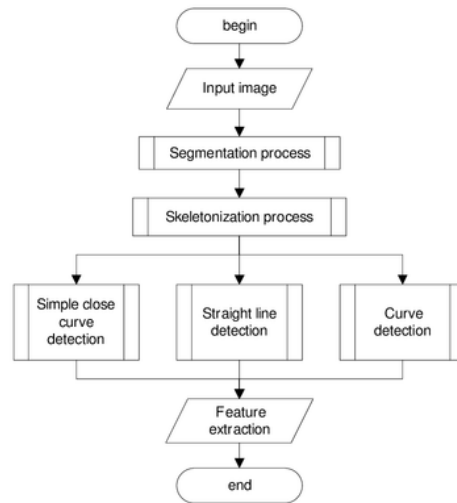















Fig. 5. Flowchart of the system

The following are the results of the simple close curve detection. The number of simple close curve contained in several sample images can be seen in Table 1.

Table 1. Experiments results of simple close curve detection

No	Java Character	Number of simple close curve detected
1	ꦲ	0
2	ꦲꦏ	2
3	ꦲꦏꦸ	3
4	ꦲꦏꦸꦏ	3
5	ꦲꦏꦸꦏꦸ	0
6	ꦲꦏꦸꦏꦸꦏ	2
7	ꦲꦏꦸꦏꦸꦏꦸ	1
8	ꦲꦏꦸꦏꦸꦏꦸꦏ	0
9	ꦲꦏꦸꦏꦸꦏꦸꦏꦸ	0
10	ꦲꦏꦸꦏꦸꦏꦸꦏꦸꦏ	4
11	ꦲꦏꦸꦏꦸꦏꦸꦏꦸꦏꦸ	0
12	ꦲꦏꦸꦏꦸꦏꦸꦏꦸꦏꦸꦏ	2
13	ꦲꦏꦸꦏꦸꦏꦸꦏꦸꦏꦸꦏꦸꦏ	1



14		1
15		2
16		1
17		1
18		2
19		0
20		0
21		2
22		0
23		1
24		1
25		1
27		1

From the experiment results in Table 1, there are incorrect calculations of the number of close curve, that is the sample number 2, 3, 4, 6, 10, 12, 15, 16, 18, 21, 24, and 25. This error rate reaches 44.44%. Some existing problems are:

1. Two parallel lines are too close together, so it becomes a closed curve after the thinning process.
2. There are areas that are too close together, so it becomes a closed curve after the thinning process.
3. The presence of noise in the form of a closed curve with a small size.

To solve the problem number three, we add minimum area parameter which used to limit the value of the minimum area of a closed area that can be considered as the actual close curve, not just as a noise. To overcome the problem of numbers 1 and 2, we divide the image into two equal segments (top and bottom). By using minimum area parameter and dividing image, we got better experiment result. Closed curve is considered as noise if the area less than or equal to 2 pixels.

The following is experiment of line detection by using several parameters. The parameters are:

- Theta resolution: Resolution angle in degrees.
- Threshold: The minimum value limit value in the accumulator array can be expressed as a line. Threshold consists of two types, namely by percentage and by value.
- Min line length: The value of the minimum length of a line segment can be considered to be a line.
- Max line gap: The minimum distance between the lines can be detected as a line.

Table 2. Experiment results of line detection

No	Parameter	Image	Thinned image	Result image	Union of line detected and thinned image	Number of line detected
1	Theta resolution: 1 Neighbor: 4 Threshold: 5 Min line length: 4 Max line gap: 4					4
						5
2	Theta resolution: 2 Neighbor: 4 Threshold: 5 Min line length: 4 Max line gap: 4					8
						8
3	Theta resolution: 3 Neighbor: 4 Threshold: 5 Min line length: 4 Max line gap: 4					7
						7
4	Theta resolution: 5 Neighbor: 4 Threshold: 5 Min line length: 4 Max line gap: 4					2
						5
5	Theta resolution: 7 Neighbor: 4 Threshold: 5 Min line length: 4 Max line gap: 4					3
						3
6	Theta resolution: 1 Neighbor: 4 Threshold: 5 Min line length: 4 Max line gap: 4					5
						2
7	Theta resolution: 3 Neighbor: 4 Threshold: 5 Min line length: 4 Max line gap: 4					4
						3
8	Theta resolution: 5 Neighbor: 4 Threshold: 5 Min line length: 4 Max line gap: 4					4
						2

From the experiment results in Table 2, we can conclude that the smaller the value of theta resolution, more lines can be detected, and also we can see that theta resolution value of 1 gives the best results.










We also do experiments on changing of threshold parameter. From experiment results, it can be concluded that if the threshold is too small, will causes noise be detected. However, if the threshold is too large causing a short line will not be detected. The threshold value for 7 pixels provide relatively good results and stable for line detection.

From other experiment on changing of line length, we get results that the optimum line length ranges between 4 - 5 pixels.

From experiments on changing maximum line gap, it can be concluded that if the max line gap is too small will causes many lines are detected. And from the results, maximum line gap of 4 pixels will give good results.

The following experiment of curve detection by using several parameter values. The experiment result can be seen in Table 3.

**Table 3.** Experiment results of curve detection

No	Parameter	Image	Result Image	Number of curve detected
1	Theta Step: 4 Neighbor: 7 Threshold: 9			4
				0
2	Theta Step: 5 Neighbor: 7 Threshold: 9			3
				0
3	Theta Step: 6 Neighbor: 7 Threshold: 9			4
				0

The parameters tested in the detection curve are theta step, neighbor and threshold value. Theta step value affects the increment value of angle (theta), the line will be detected every n-degree value of angle. From the experiment results, it can be concluded that the smaller value of theta step, the detail lines can be detected, and the theta step value 1 gives the best results.

Neighbor parameters affect the width of the window for detecting local maxima. From experiment results, the smaller the value of the neighbor, more lines can be detected because of the larger peak, and the optimum value of neighbor is 7, it will get the best result.

The threshold parameter affects the number of crossovers in the parameter space. From experiment results, it can be concluded that if the threshold is too small, will cause noise is detected. However, if the threshold is too large causing a short line is not detected. The optimum threshold value of 7 pixels gives relatively good results.

By using the parameters obtained from each experiment, we conduct experiment on a number of Java characters document images to get the number of simple closed curve, lines and curves as well as a comparison with the calculation of the number manually. The results of the simple closed curve detection reaches 82.85%, and the result of line detection reached 74.28%. While the result of curve detection is still low, reaching only 51.42%.

## 6 Conclusion

In this paper we have conducted experiment on feature extraction of Java character document images that later will be used for the detection of Java character. Based on the experiment results, it can be concluded that the optimum parameters for the detection of simple closed curve is a minimum area of 2 pixels, while the optimum parameters for detecting the line is the theta resolution value of 1 pixel, threshold 7 pixels, line length ranges between 4 - 5 pixels, and maximum line gap 4 pixels. For detection of curves, the best parameters are theta step 1, neighbor 7 pixels and the threshold 7 pixels. From experiment on the samples of Java character document images, the results of simple closed curve and line feature extraction are quite good, respectively reached 82.85% and 74.28%. However, the result of the curve detection is still not good, only reached 51.42%.

## Acknowledgment

We thank to PT Coordination of Private Higher Education Region VII, East Java, Indonesia for funding this research by Research Competitive Grant DIPA-PT Coordination of Private Higher Education Region VII, East Java, Indonesia fiscal year 2015 entitled (in Bahasa Indonesia) "Aplikasi Pengenalan Aksara Jawa Guna Digitalisasi Dokumen Beraksara Jawa Untuk Mendukung Pelestarian Budaya Nasional".

## References.

1. Parker, J. R.: Algorithm for image processing and computer vision. New York, NY, USA: John Wiley & Sons, Inc. (1997)
2. Adjie, N., & Maulana.: Mathematics Problem Solving. Bandung: UPI Press (2006)
3. P.V.C. Hough.: Machine Analysis of Bubble Chamber Pictures. Proc. Int. Conf. High Energy Accelerators and Instrumentation (1959).
4. Duda, R. O. and P. E. Hart.: Use of the Hough Transformation to Detect Lines and Curves in Pictures," Comm. ACM, Vol. 15, pp. 11-15 (1972)
5. Gonzalez, R., & Woods, R.: Digital Image Processing, Third Edition. New Jersey: Prentice Hall (2008)
6. Jafri, M., & Deravi, F.: Efficient algorithm for the detection of parabolic curves. Vision Geometry III, 53-62 (1994)

# Feature Extraction for Java Character Recognition

## ORIGINALITY REPORT

9%

SIMILARITY INDEX

4%

INTERNET SOURCES

9%

PUBLICATIONS

2%

STUDENT PAPERS

## PRIMARY SOURCES

- |       |   |    |
|-------|---|----|
| 1     | <p>Gregorius Satia Budhi, Rudy Adipranata.<br/>"Comparison of bidirectional associative memory, counterpropagation and evolutionary neural network for Java characters recognition", 2014 International Conference of Advanced Informatics: Concept, Theory and Application (ICAICTA), 2014<br/>Publication</p> | 3% |
| <hr/> |   |    |
| 2     | <p><a href="http://www.sersc.org">www.sersc.org</a><br/>Internet Source</p>   | 1% |
| <hr/> |   |    |
| 3     | <p>Mat Jafri, Mohammad Z., Farzin Deravi, and Angela Y. Wu. "", Vision Geometry III, 1995.<br/>Publication</p>  | 1% |
| <hr/> |   |    |
| 4     | <p>Submitted to Southampton Solent University<br/>Student Paper</p>   | 1% |
| <hr/> |   |    |
| 5     | <p>Muhammad Labiyb Afakh, Anhar Risnumawan, Martianda Erste Anggraeni, Mohamad Nasyir Tamara, Endah Suryawati Ningrum. "Aksara jawa text detection in scene images using convolutional neural network", 2017</p>  | 1% |

# International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC), 2017

Publication

6

Rudy Adipranata, Yulia, Liliana, Gregorius Satia Budhi. "Implementation of Javanese Text to Speech using MaryTTS Engine", Proceedings of the 2018 International Conference on Machine Learning and Machine Intelligence - MLMI2018, 2018

Publication

1%

7

Submitted to University of Southampton

Student Paper

1%

Exclude quotes On

Exclude bibliography On

Exclude matches < 1%