ITMSOC

Available online at http://itmsoc.org

Information Technology Management Society

ITMSOC Transactions on Information Technology Management 01 (2016) 19–25

ITMSOC–ITM

ITMSOC Transactions on Information Technology Management

http://www.itmsoc.org

Comparison between Shape-Based and Area-Based Features Extraction for Java Character Recognition

Rudy Adipranata*, Gregorius Satia Budhi, Liliana, Bondan Sebastian

Informatics Department, Petra Christian University Siwalankerto, Surabaya, Indonesia

Received 29 February 2016; Accepted 23 August 2016

Abstract

Java language is one of the local languages are widely used in Indonesia. Java language is widely used by resident of the island of Java. Java language has special character called Java character. In this research we compare features extraction which will be used to perform the recognition of Java character. The accuracy of recognition is greatly affected by accuracy of features extraction. Because if there are a lot of similar features between one character with other characters, may cause the system to recognize as the same characters. In this research, we compare between shape-based features and area-based features. Shape-based features consist of curves, lines, and loop composing a Java character. The number of curves, lines, and loop will vary between characters with other characters. For area-based features extraction, each character divide into 9×9 equal regions. In each region, the number of pixels will be calculated. From experimental results, area-based features extraction gives better result than shape-based features extraction. This experiment is done by using probabilistic neural network (PNN) as a method of recognition. By using shape-based features extraction, the system only has recognition accuracy below 20%, but using area-based features extraction, the recognition accuracy can achieve more than 60%.

Keywords: Shape-based Feature Extraction, Area-based Feature Extraction, Probabilistic Aeural Network.

1. Introduction

INDONESIA is a nation composed of many ethnic groups. Each ethnic has its own culture. One ethnic is Javanese who mostly lived on the island of Java. Javanese has a culture which covers language, writing, dancing, food, etc. In writing, Javanese has its own form of letters referred to the character of Java. Learning of Java characters has its own difficulty level, because Java character consist of so many symbols, categorized as basis characters, vowels, complementary, and so on. Because it is difficult to learn, then lately not many people can do the writing or reading of Java characters. In this research, we will try to preserve the Javanese character by developing a system to recognize Java character automatically.

One of the most important parts of the Java character recognition is feature extraction. The accuracy of recognition is greatly affected by accuracy of features extraction. Because if there are a lot of similar features between one character with other characters, may cause the system to recognize as the same characters. In this research, we compare between shape-based features and area-based features. Shape-based features consist of curves, lines and loop composing a Java character. The number of curves, lines, and loop will vary between characters with other characters [1]. To detect shape-based features, flood fill algorithm, Hough transform [2–4] will be used. For area based features extraction, each character divide into nine equal regions. In each region, the number of pixels will be calculated [5].

Before doing features extraction, several image preprocessing have been applied. The first one is skeletonization and followed by image segmentation. Skeletonizing is one of image processing that is used to reduce the pixels of an image while maintaining in-

*Corresponding author.
Email address: rudya@petra.ac.id (Rudy Adipranata)

formation, characteristic, and important pixels of the object. 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. This is implemented by changing the initial image in binary into skeletal representation of the image [2]. The next preprocessing is image segmentation. Image segmentation is done to get a piece of each Java character.

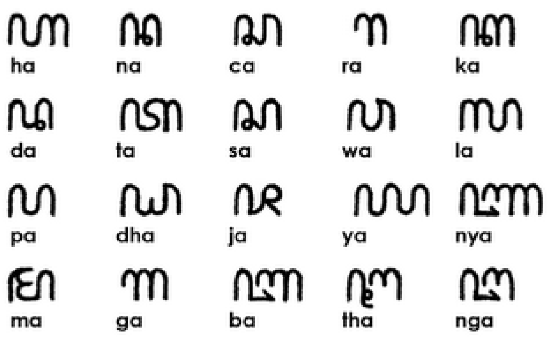


Fig. 1. Basis Java character.

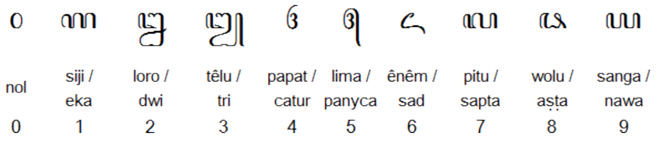


Fig. 2. Main Java characters.

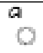
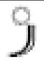
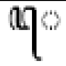
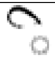
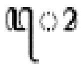
Java character	Description	*Sandhangan* name
	Vowel i	*Wulu*
	Vowel u	*Suku*
	Vowel é	*Taling*
	Vowel ê	*Pepet*
	Vowel o	*Taling tarung*

Fig. 3. Sandhangan character for vowels.

2. Java Character

Java characters have 20 basis characters, 20 characters who serve to close vowel (called pasangan), 8 main characters (called murda) used to write the beginning of sentences and words that show proper names, titles, cities, institutions, and other names, some complimentary (called sandhangan) as vowels, special characters, punctuation, etc. Basis character can be seen in Fig. 1 [6].

Main characters that used to write the beginning of sentences or names of person can be seen in Fig. Fig. 2 [7].

Some complementary especially for vowels can be seen in Fig. 3 [8].

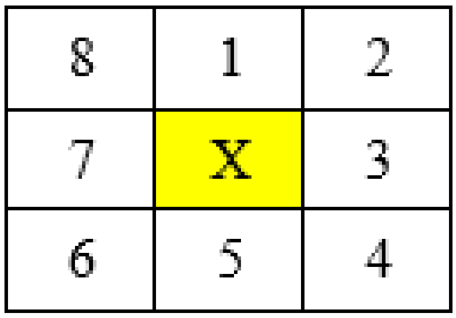


Fig. 4. Pixel and its eight neighbors.

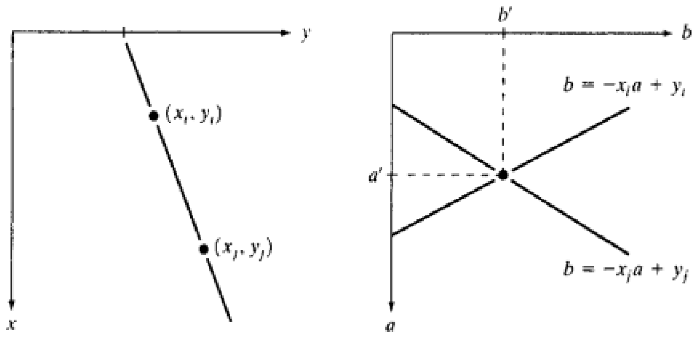


Fig. 5. a) xy area. b) Parameter space.

3. Skeletonizing

Skeletonizing is one of image processing that is useful for reducing pixel of an image (binary image), but may still retain the information, and the characteristics of the object that is on the image. There are several methods that can be used to implement skeletonizing, one of which is a method proposed by Zang-Suen [5]. The basic idea is to determine whether a pixel could be eroded just by looking at the eight neighbors of the pixel. The eight neighbors and the pixel can be seen in the Fig. 4.

To determine whether a pixel can be removed or not, there are two requirements. The first requirement is as follows [1]:

- If pixel has more than one and less than 7 neighbors, then it can be removed.
- If pixel has the only one connectivity, then it can be removed.

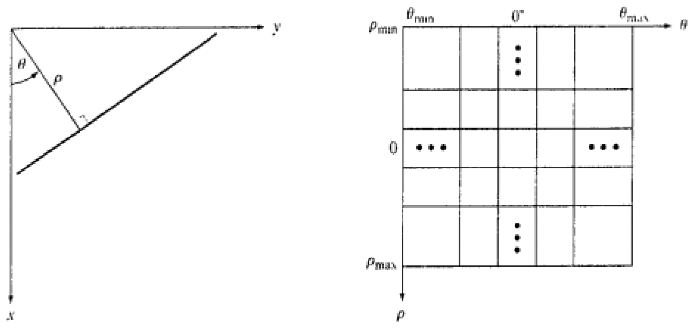


Fig. 6. a) Representasi normal suatu garis. b) Parameter space (ρ, θ).

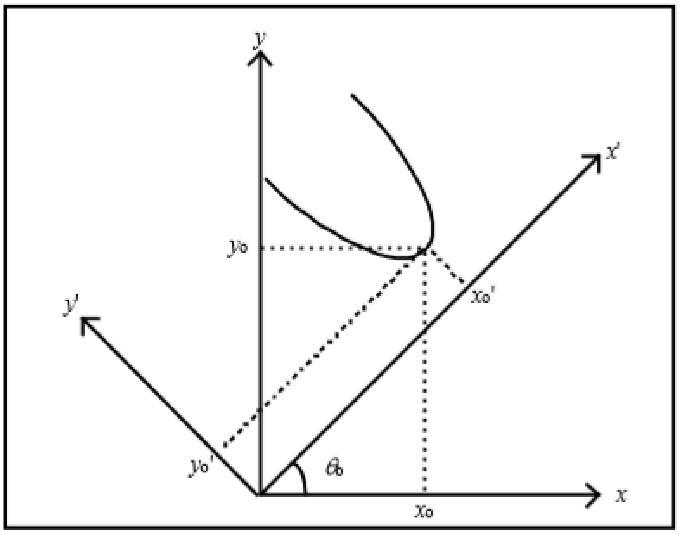


Fig. 7. Parabolic curve.

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

The second requirement differs from first requirement in the last two steps:

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

The first step on the above steps can be written as a logical expression in (1).

$$v(X) \wedge \left(\sim edge(X) \vee \left(v(d_3) \wedge v(d_5) \wedge (v(d_1) \vee v(d_7))\right)\right) \tag{1}$$

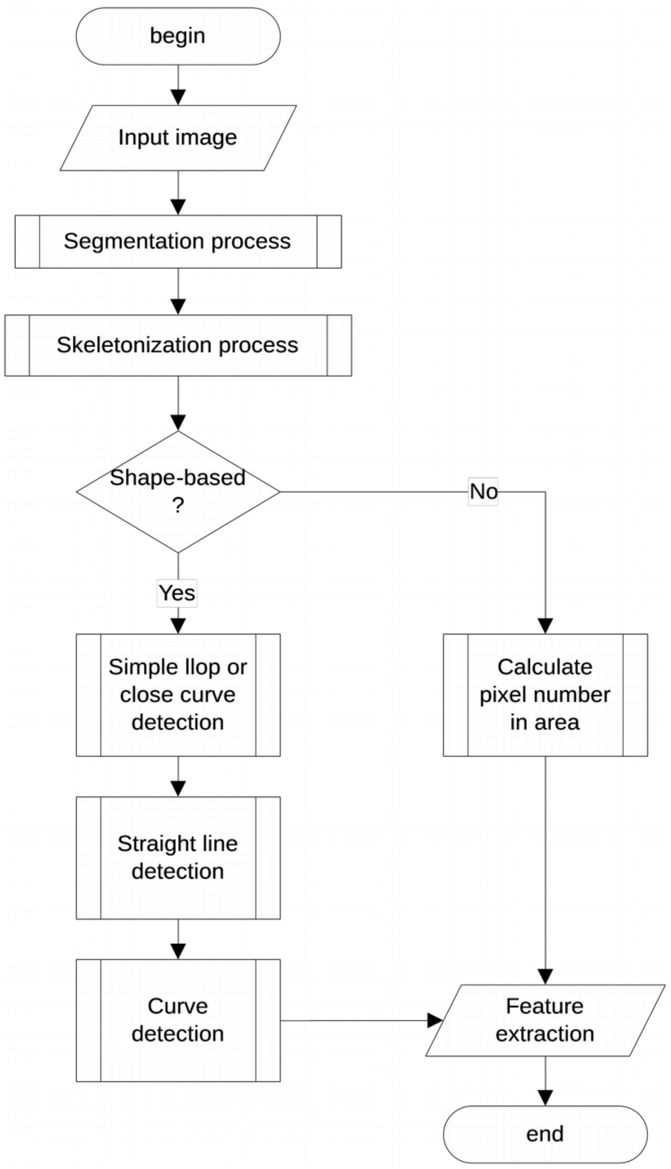
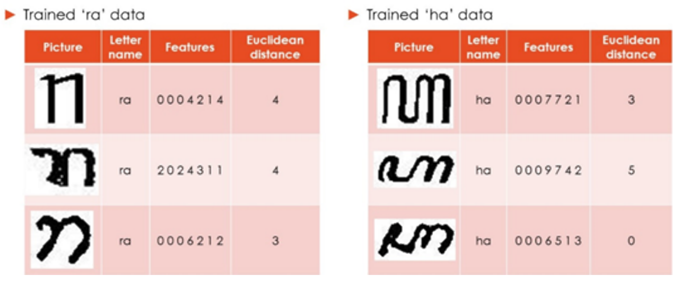


Fig. 8. System design.



▶ Trained 'ra' data

Picture	Letter name	Features	Euclidean distance
	ra	0004214	4
	ra	2024311	4
	ra	0006212	3

▶ Trained 'ha' data

Picture	Letter name	Features	Euclidean distance
	ha	0007721	3
	ha	0009742	5
	ha	0006513	0

Fig. 9. Example of shape-based feature extraction.

Table 1. Recognition Result of PNN Using Shape-based Feature Extraction.

No.	Image	Number of Java Character for Training	Number of Java Character for Classification	Accuracy (%)
1	Image 1	60	136	15.44
2	Image 2	46	110	16.36
3	Image 3	57	79	20.25
4	Image 4	61	222	23.42
5	Image 5	51	126	22.22
6	Image 6	28	27	29.63
7	Image 7	59	153	15.69
8	Image 8	43	87	16.09
9	Image 9	56	142	14.79
10	Image 10	15	10	30.00
Average				**18.77**

Table 2. Recognition Result of PNN Using Area-Based Feature Extraction.

No.	Image	Number of Java Character for Training	Number of Java Character for Classification	Accuracy (%)
1	Image 1	60	136	63.24
2	Image 2	46	110	49.09
3	Image 3	57	79	54.43
4	Image 4	61	222	56.76
5	Image 5	51	126	64.29
6	Image 6	28	27	55.56
7	Image 7	59	153	66.01
8	Image 8	43	87	75.86
9	Image 9	56	142	62.68
10	Image 10	15	10	60.00
Average				**61.08**

While the second step on the above steps can be written as a logical expression in (2).

$$v(X) \wedge \left(\sim edge(X) \vee \left(v(d_7) \wedge v(d_1) \wedge (v(d_5) \vee v(d_3))\right)\right) \quad (2)$$

X indicates the pixel being examined. V function generates the pixel value (1 = true for foreground pixels and 0 = false for pixel background). Edge function is true if X at the end of the object (referring to the number of neighbors of more than one and less than seven and the number of connectivity = 1). d_1, d_3, d_5, and d_7 referring to the neighbors pixels in a certain direction to the pixel X as seen in Fig. 4.

4. Feature Extraction

Feature extraction is the process of finding a mapping from original features into new features that is expected to give better result for class differences [9]. Feature extraction is an important topic in the classification, because the good features will be able to increase the rate of accuracy, while features that are not well tended will decrease the rate of accuracy.

In this research we will use and compare shape-based and area-based feature extraction. For shape-based, consist of curves, lines and loops that composed one Java character. While for area-based, image of Java character divided into small areas and the number of pixel in each area will be calculated.

4.1. Flood Fill Algorithm

Flood fill algorithm is used to detect a loop or closed curve. It has three main parameters, namely the start node, the target color and color replacement. Flood fill algorithm searches all nodes in the array which are connected to the start node through the path of the target color and then replace it with a replacement color. The following steps below are flood fill based algorithm by using recursion [5]:

1) Return if the node has not the same color as a target
2) Set the nodes color into a replacement color.
3) Run flood fill one step from the node to the west.
   Run flood fill one step from the node to the east.
   Run flood fill one step from the node to the north.
   Run flood fill one step from the node to the south.
4) Return.

4.2. Hough Transform

Hough Transform is used to detect shape in the image, e.g. line or curve. Hough Transform was first proposed by P.V.C Hough [3], and then Duda and Hart have implemented it to detect the lines in the image [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. Then the algorithm takes a vote on an array element called the accumulator array. The straight lines that will be detected by Hough Transform should satisfy (3) and (4).

$$y = a \cdot x + b, \quad (3)$$
$$b = -x_1 \cdot a + y_1. \quad (4)$$

By changing (3) and (4), 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 Fig. 5.

First, the value of the accumulator is initialized to 0. The edge of the object in the image, for each point (x, y), the value of b is calculated according to Eq. (4). The result is to be rounded to the nearest acceptable value in accumulator. Accumulator value will increase for each appropriate value a and value b according to (5).

$$A(a, b) = A(a, b) + 1, \quad (5)$$
$$\rho = x \cos\theta + y \sin\theta, \quad (6)$$
$$A(\rho, \theta) = A(\rho, \theta) + 1. \quad (7)$$

Each edge point has appropriate 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. A polar equation for a line with a parameter ρ and orientation θ has been proposed by Duda and Hart [4] as seen in (6).

Each point in the image is mapped into the accumulator for each value ρ and θ which satisfy (7).

The illustration of this mapping can be seen in Fig. 6.

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

Hough Transform can be used also to detect a parabolic curve. This was proposed by M.Z. Mat Jafri and F. Deravi [10]. There are four parameters involved, namely the point (x_0, y_0), orientation (θ), and the coefficient which contains information about the parabolic curvature in standard parabolic curve detection. But this algorithm can detect parabolic curve in any orientation using only three parameters [10]. The parameters are the point (x_0, y_0) and orientation θ. In this algorithm, by using 3D accumulator, all parabolic curves in various positions can be detected. The approach uses a point on the curve as a parameter which also shows the position of maximum curvature of the parabolic curve. For the gradient approach, Sobel operator is used. A coordinate transformation matrix is used to derive a new parabolic equations involving parabolic curve orientation to detect the parabolic curve in any orientation. Fig. 7 shows the graphic of parabolic curve with a specific orientation angle.

(x', y') coordinates is the (x, y) coordinates rotation by θ 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. Equation (8) is equation of the parabola in the (x', y') coordinates [10].

$$(y' - y'_0) = p \cdot (x' - x'_0)^2. \tag{8}$$

Equation (9) is standard two dimensional geometry matrix for counter-clockwise rotation with θ angle transformation.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \tag{9}$$

By substituting the value of x, y, x_0 and y_0 in (9) to (8), the parabolic (8) can be written as (10).

The value of differentiation of this equation is given in (11).

By substituting (11) into (10), a new relation to the parabola vertex and the orientation (x_0, y_0, θ) is shown in (12) [6].

Using the above relationship, parabola detection in various orientations can be done using only three dimensional accumulator arrays.

5. System Design

The system design is done by using a flowchart, which can be seen in Fig. 8.

Before doing feature extraction, image segmentation first performed to obtain an image of each Java character. Then, after the obtained images that contain only one Java character, skeletonizing is done in order to get the Java character with a thickness of one pixel. From the results of skeletonizing, then feature extraction will be done either based on shape-based or area-based.

For shape-based feature extraction, the process will detect loop, lines, and curves that form Java character. For each of Java character, is divided into two segment, upper segment and lower segment. For each segment, the number of detected loop, line, and curve is calculated. Those numbers are features that will be used as input to the Java character recognition. For each of Java character, there will be seven features from shape-based feature extraction: number of total loop, number of loop in upper segment, number of loop in lower segment, number of line in upper segment, number of line in lower segment, number of curve in upper segment, and number of curve in lower segment.

In the area-based feature extraction, the first step is to resize the image of each Java character to become 81×81 pixels. Then the image that has been resized, divided into 9×9 regions. So in total there are 81 regions with the size of each area is 81 pixels. The number of black pixels in each area is calculated, and the results of these calculations are features that will be used later in the process of Java character recognition. There are total 81 features from area-based feature extraction.

For recognition process, probabilistic neural network (PNN) algorithm is used. This method is used because PNN has a high accuracy in the classification of data, also has high speed when performing the process [11].

6. Experimental Results

Ten images of Java character are used in experiment. In each images, all Java characters divided into two, for training and for classification. All of them are processed with shape-based feature extraction and area-based feature extraction. The results of recognition rate are compared between them.

Example of shape-based feature extraction result can be seen in Fig. 9.

Experiment result of shape-based feature extraction for Java character recognition using PNN method can be seen in Table 1.

From experimental result in Table 1, it can be seen that accuracy rate is only 18.77, below 20%. For experiment using area-based feature extraction, can achieve accuracy rate 61.08%. The result can be seen in Table 2.

From the results of this experiment, although it can be concluded that the area-based feature extraction is better, but the results of the recognition is still not satisfactory, because it only reached slightly above 60%. Further research can be done using another recognition method to obtain better recognition results.

$$(-x\sin\theta + y\cos\theta) - (-x_0\sin\theta + y_0\cos\theta) = p\left[(x\cos\theta + y\sin\theta) - (x_0\cos\theta + y_0\sin\theta)\right]^2. \tag{10}$$

$$-\sin\theta + \frac{dy}{dx}\cos\theta = 2p\left[(x\cos\theta + y\sin\theta) - (x_0\cos\theta + y_0\sin\theta)\right]\cdot\left[\cos\theta + \frac{dy}{dx}\sin\theta\right]. \tag{11}$$

$$y_0 = \left[\frac{k_1(x\cos\theta + y\sin\theta) + (x\sin\theta - y\cos\theta)}{k_1(\sin\theta - \cos\theta)}\right] - x_0\left[\frac{k_1\cos\theta + \sin\theta}{k_1\sin\theta - \cos\theta}\right], \tag{12}$$

$$\text{where,}\quad k_1 = x_0\frac{-\sin\theta + \frac{dy}{dx}\cos\theta}{2(\cos\theta + \frac{dy}{dx}\sin\theta)}. \tag{13}$$

7. Conclusions

From the experimental results, it can be concluded that the use of area-based feature extraction give better results than the use of shape-based feature extraction. By using PNN as a Java character recognition method, the use of area-based feature extraction can result in accuracy rate of over 60%, while the use of shape-based feature extraction can only result accuracy rate less than 20% accuracy rate. This is because many of Java characters that have number of shape features that are similar to each other.

Acknowledgment

We thank to PT Coordination of PrivateHigher 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 no. 20/SP2H/PDSTRL_PEN/LPPM-UKP/IV/2015 entitled (in Bahasa Indonesia) "Aplikasi Pengenalan Aksara Jawa Guna Digitalisasi Dokumen Beraksara Jawa Untuk Mendukung Pelestarian Budaya Nasional".

References

1. Adipranata R, Liliana, Indrawijaya M, Budhi GS. Feature extraction for Java character recognition. In: Proc. of 4th International Conference on Soft Computing, Intelligent System and Information Technology. Bali, Indonesia; 2015. Available from: http://fportfolio.petra.ac.id/user_files/99-015/FeatureExtractionforJavaCharacterRecognition.pptx.
2. Gonzalez R, Woods R. Digital Image Processing. 3rd ed. New Jersey: Prentice Hall; 2008.
3. Hough PVC. Machine analysis of bubble chamber pictures. In: Proc. 2nd International Conference on High-Energy Accelerators and Instrumentation (HEACC 1959). vol. C590914. CERN, Geneva, Switzerland; 1959. p. 554–558. Available from: http://inspirehep.net/record/919922/files/HEACC59_598-602.pdf.
4. Duda RO, Hart PE. Use of the Hough transformation to detect lines and curves in pictures. Commun ACM. 1972 Jan;15(1):1115. Available from: http://dx.doi.org/10.1145/361237.361242.
5. Parker JR. Algorithm for Image Processing and Computer Vision. New York: John Wiley and Sons; 2010.
6. Daryanto. Kawruh Basa Jawa Pepak. Surabaya:Apollo Lestari; 1999. Available from: http://onesearch.id/Record/IOS1-INLIS000000000629660.
7. Javanese Alphabet (Carakan); 2015. Available from: http://www.omniglot.com/writing/javanese.htm.
8. Hastuti D. Learning Java Character; 2011. Wordpress Blog. Available from: https://dhenokhastuti.wordpress.com/2011/04/11/mari-belajar-lagi-menulis-aksara-jawa/.
9. Guo L, Rivero D, Dorado J, Munteanu CR, Pazos A. Automatic feature extraction using genetic programming: An application to epileptic EEG classification. Expert Systems with Applications. 2011 Aug;38(8):10425–10436. Available from: http://dx.doi.org/10.1016/j.eswa.2011.02.118.
10. Jafri MZM, Deravi F. Efficient algorithm for the detection of parabolic curves. In: Melter RA, Wu AY, editors. Proc. SPIE 2356, Vision Geometry III. SPIE-Intl Soc Optical Eng; 1995. p. 53–61. Available from: http://dx.doi.org/10.1117/12.198623.
11. Specht DF. Probabilistic neural networks. Neural Networks. 1990 Jan;3(1):109118. Available from: http://dx.doi.org/10.1016/0893-6080(90)90049-Q.

Biographies

Rudy Adipranata is currently a senior lecturer in Informatics Department, Petra Christian University, Surabaya, Indonesia.

He received his bachelor degree in Electrical Engineering from Petra Christian University, Surabaya, Indonesia, and master degree in Software Engineering from Graduate School of Software, Dongseo University, Busan, South Korea.

His research interests are image processing and computer vision.

Gregorius Satia Budhi is currently a senior lecturer in Informatics Department, Petra Christian University, Surabaya, Indonesia.

He received his bachelor degree in Electrical Engineering, minority on computer science from Adhi Tama Institute of Technology, Surabaya, Indonesia, and master degree in Informatics from Sepuluh November Institute of Technology, Surabaya, Indonesia.

His research interests are artificial intelligence and data mining.

Liliana is currently a senior lecturer in Informatics Department, Petra Christian University, Surabaya, Indonesia.

She received her bachelor degree in Informatics from University of Surabaya, Indonesia, and master degree in Visual Content from Graduate School of Software, Dongseo University, Busan, South Korea.

Her research interests are image processing and computer vision.

Bondan Sebastian is currently a web developer at Smart-IT, Surabaya, Indonesia.

Hereceived his bachelor degree in Informatics from Petra Christian University, Surabaya, Indonesia.