Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 17(1): 67-73, Mar. 2019

Regular paper

# Comparison of Image Duplication Detection Using the Polar Coordinates System and Histogram of Oriented Gradients Methods

**Kartika Gunadi**[*], **Rudy Adipranata**, **and Ivan Suryajaya**

Informatics Department, Petra Christian University, Surabaya 60236, Indonesia

## Abstract

In the current era of digital technology, and with the help of existing software, digital photo manipulation is becoming easier and faster. One example of this is the development of powerful image processing software that makes it easy for a digital image to be manipulated and edited. It is therefore very important to protect and maintain public trust in digital images. Several methods have been developed to detect image manipulation. In this paper, we compare two methods for detecting image duplication due to copy-move actions, namely the polar coordinate system and the histogram of oriented gradients methods. The former is a method based on the transfer of a Cartesian image to a polar form, making it easy to tell whether there are objects that have undergone a copy/move in an image, while the latter is a method for retrieving information related to the distribution, which uses a target in the local area as a tool to represent the shape of the target. We compare the accuracy, speed and memory usage of these two methods.

**Index Terms**: Histogram of Oriented Gradients, Polar Coordinates System, Digital Image Manipulation

## I. INTRODUCTION

In the current era of digital technology, and with the use of existing software, the manipulation of digital images is becoming easier and faster. One example of this is the development of powerful image processing software, which can make it easy for a digital image to be manipulated and edited. Manipulation of images can lead to both positive and negative effects. Positive uses of image manipulation include touching up pictures, fixing damaged images, and others. Although the negative uses of image manipulation are not very different from the positive ones, they can be misused, for example to move the location of an object. In this case, the change in the location can create a misunderstanding over the meaning of an object in an image. It is therefore very important to protect and maintain public trust in digital imaging.

With further developments in this field, it will become ever easier to manipulate an image, and methods have therefore been developed to detect the authenticity of an image in terms of copy-move operations. Copy-move indications can be identified by various methods such as preprocessing procedures, hybrid features, a polar coordinates system (PCS) [1], or a histogram of oriented gradients (HOG) [2]. In this research, we compare two methods of detecting image fraud, namely the PCS and HOG methods. PCS is a method that relies on the transfer of a Cartesian image to a polar form, making it easy to identify whether there are objects that have undergone copy-move operations in an image [1]. HOG is a method for retrieving information related to the distribution,

**Received** 09 November 2018, **Revised** 10 January 2019, **Accepted** 15 January 2019
***Corresponding Author** Kartika Gunadi (E-mail: kgunadi@petra.ac.id, Tel: +62 31 8439040)
Informatics Department, Petra Christian University, Surabaya 60236, Indonesia

Open Access https://doi.org/10.6109/jicce.2019.17.1.67 print ISSN: 2234-8255 online ISSN: 2234-8883

using a target in the local area as a tool to represent the shape of target. It also provides a cluster of gradient information. Detection of copy-move operations depends on the shape and texture of the damaged image [2]. A comparison between these two methods is carried out based on the use of memory, the speed of performing a one-time process, and the accuracy of each method in terms of determining the copy-move operation that was applied to a picture. For testing, we use images from the CoMoFoD database [3].

## II. CARTESIAN AND POLAR COORDINATES

Cartesian coordinates are used to represent an object consisting of flat surfaces or straight lines. The shape of the elbow can be easily represented in both 2D and 3D Cartesian coordinates. 2D coordinates consist of two axes: a horizontal axis called the x-axis and a vertical axis called the y-axis. 2D Cartesian coordinates are used to describe 1D and 2D objects. Examples of 1D objects are lines (including both straight lines and curved lines). An example of a 2D object is a flat surface. Both 1D and 2D objects can be described in 3D coordinates, although 3D objects can only be described in 3D coordinates [5].

Polar coordinates can be obtained by converting Cartesian coordinates in an $(x, y)$ scheme into a polar system $(\theta, r)$, with the aim of allowing transformations of geometric images such as rotation, magnification or reflection. If two similar objects undergo different rotation, magnification, or reflection transformations, the objects still have the same polar shape, but different appearances [5].

## III. POLAR COORDINATES SYSTEM

The first step in the PCS method [1] converts the user input RGB image into a greyscale image using (1):

$$I = 0.299R + 0.587G + 0.114B. \quad (1)$$

Following this, the image is divided into small blocks with pixel displacements by as much as one pixel and along the specified $b$. The number of blocks can be calculated using (2) [6]:

$$B = (M - b + 1)x(N - b + 1) \quad (2)$$

where $M$ and $N$ are the width and height of image, respectively, $b$ is the size of the blocks (in pixels), which is specified by the user, and $B$ is the total number of blocks.

From Cartesian coordinates, the image is converted into polar coordinates in order to detect the existence of objects that have been copied and rotated. After this, a fast Fourier transform (FFT) is used [6, 7]. The FFT process is carried out on each column, and a sorting process is then applied to sort the column. Next, we perform a raster scan process in which the block containing a *[b, b]* (2D) array is changed into a *[b*b]* (1D) array by connecting each column. Following this, sorting is done on the rows and the initial place of each block is stored. A matching procedure is then carried out using the correlation (3) between *block[i]* and *block[i+1]* [1].

$$corr = \frac{\sum_{i=1}^{n}(Bx_i - \overline{Bx})(By_i - \overline{By})}{\sqrt{\sum_{i=1}^{n}(Bx_i - \overline{Bx})^2 \cdot \sum_{i=1}^{n}(By_i - \overline{By})^2}} \quad (3)$$

where $B_x$ represents *block[i]* and $B_y$ represents *block[i+1]*.

If the correlation results are greater than a given threshold $T_1$, then we proceed with calculating the spatial distance between the two blocks using (4) [1]:

$$dis = \sqrt{(As_i^x - As_{i+1}^x)^2 + (As_i^y - As_{i+1}^y)^2} \quad (4)$$

where $(As_i^x, As_i^y)$ represents *block[i]* and $(As_{i+1}^x, As_{i+1}^y)$ represents *block[i+1]*. If the distance value is greater than a given threshold $T_2$ then the image will be detected either as being the same or as having undergone a copy-move [1].

## IV. HISTOGRAM OF ORIENTED GRADIENTS

The HOG method is used to extract features from an object in an image. The initial process in the HOG method is to convert an RGB image to greyscale [4], and then to calculate the gradient value of each pixel [2]. After obtaining the gradient value, the next step is to determine the number of orientation bins that will be used to create the histogram. This process is called spatial orientation binning. At this stage, the image is divided into several cells, the size of which depends on the design and is usually a factor of the size of the image block used in the HOG process. In this paper, for block sizes of 8x8 we use cell sizes of $2 \times 2$.

These cells are then grouped into larger sizes called blocks to form the histogram; for example, one block consists of four cells, and these blocks are used in the normalisation process to perform HOG geometry calculations [2].

The step of the HOG method is to find the gradient value using (5) and (6) [2]:

$$G_x = [-1, 0, 1] * I(x, y) \quad (5)$$

$$G_y = [-1, 0, 1]^T * I(x, y). \quad (6)$$

After obtaining the values of $G_x$ and $G_y$, we can find the

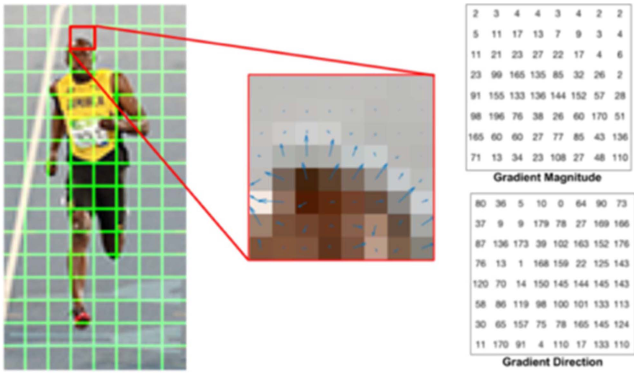


**Fig. 1.** Example of magnitude and theta of gradient [8].

values of the gradient and theta for each pixel using (7) and (8) [2]:

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \tag{7}$$

$$\theta(x,y) = \tan^{-1}\frac{G_y(x,y)}{G_x(x,y)}. \tag{8}$$

Fig. 1 shows examples of the magnitude and theta values. After obtaining the gradient and magnitude, each cell is represented as four bins, and normalisation is then carried out. A sorting process is then carried out on the vector, and a matching procedure is applied based on (9) [2]:

$$D(i, \sigma) = \min\{D(i; i-j) \ldots, D(i; i-1), D(i; i+1) \ldots, D(i; i+j)\}. \tag{9}$$

## V. SYSTEM DESIGN

### *A. Polar Coordinates System*

The first step in the PCS method is the conversion of an RGB image into a greyscale image. Following this, a blocking stage is applied, where the image is cut into smaller images with overlaps determined by the user. This blocking stage is carried out by shifting as much as one pixel right and down. The results from this blocking stage are entered into a 2D array. After this, we convert the image from Cartesian coordinates to polar coordinates, and a FFT process is carried out on each column in the 1D array. A sorting process is then carried out horizontally to compare the top values; if these values are the same, then the next values will be compared. When the sorting process is complete, a raster scan process is applied in which the 2D array is converted to a 1D array, and then combined into a column. The array is sorted and the value of the original pixel position is saved. The last step is a matching process [1]. A flowchart for the PCS method can be seen in Fig. 2.

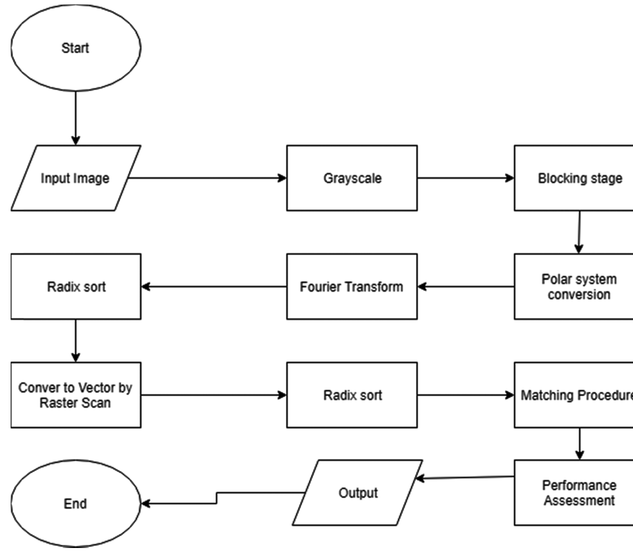


**Fig. 2.** PCS flowchart.

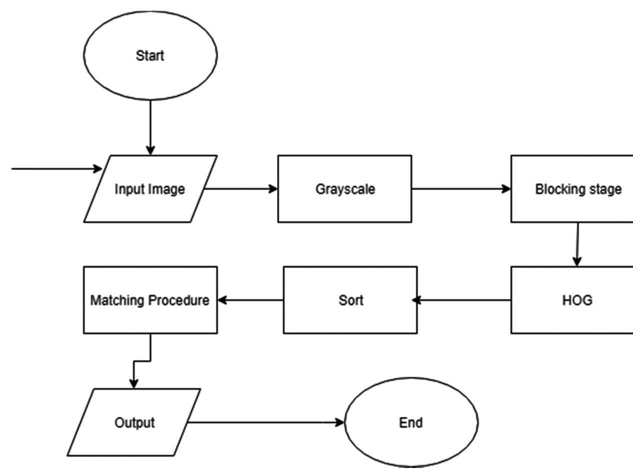


**Fig. 3.** HOG flowchart.

### *B. Histogram of Oriented Gradients*

The first step in the HOG method is to convert the input image to greyscale. After this, a blocking operation is carried out in which the greyscale image is cut into smaller images by overlapping a number of blocks determined by the user. This blocking stage is carried out by shifting as much as one pixel right and down. The results of this stage are entered into a 2D array. Next, we apply the HOG process, as explained in Section IV. Following this, the last step involves a matching procedure. A flowchart for the HOG method can be seen in Fig. 3.

## VI. EXPERIMENTAL RESULTS

### *A. Experimental Results Using the PCS Method*

In this experiment, we used block sizes of eight and 16, a similar threshold of 0.99-0.9995 and a threshold distance of 16. In the first experiment, we used the PCS method with a block size of eight for three input images. The results of test image 1 can be seen in Fig. 4. A total of 516 similar blocks were detected, and the operation took 1 min 5 s of processing time.

The results for test image 2 can be seen in Fig. 5. A total of 209 similar blocks were detected, requiring 1 min 10 s of processing time.

The results of test image 3 can be seen in Fig. 6. A total of 276 similar blocks were detected, requiring 1 min 41 s of processing time.

For the second experiment, we used a blocking size of 16. The experiment results of test image 1 can be seen in Fig. 7. A total of 403 similar blocks were detected, and the processing time was 3 min 14 s.

The results of test image 2 can be seen in Fig. 8. A total of 529 similar blocks were detected and the processing time was 3 min 15 s.

The results of test image 3 can be seen in Fig. 9. A total of 137 similar blocks were detected and the processing time was 5 min 2 s.

### *B. Experimental Results Using HOG Method*

Experiments using the HOG method were carried out using the same input images and the same parameters as the experiment using the PCS method. The block sizes were

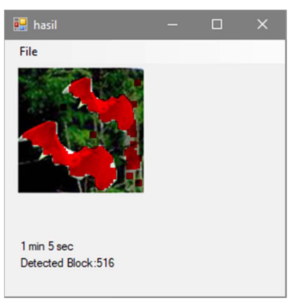


**Fig. 4.** Result of test image 1 using PCS with a block size of eight.

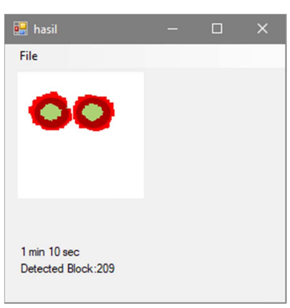


**Fig. 5.** Result of test image 2 using PCS with block size of eight.

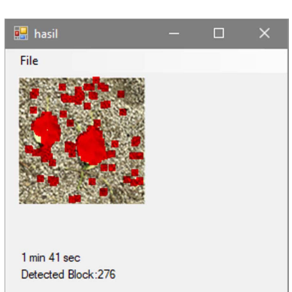


**Fig. 6.** Result of test image 3 using PCS with block size of eight.

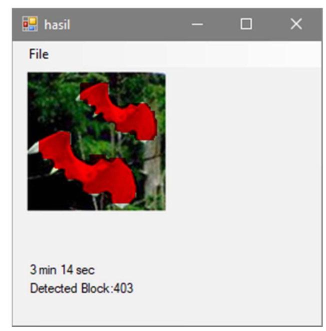


**Fig. 7.** Result of test image 1 using PCS with a block size of 16.

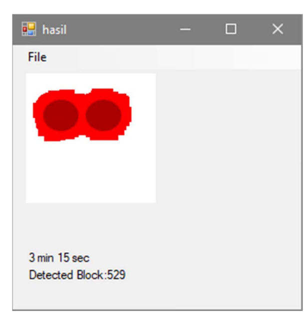


**Fig. 8.** Result of test image 2 using PCS with a block size of 16.

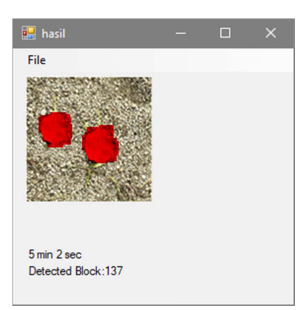


**Fig. 9.** Result of test image 3 using PCS with a block size of 16.

eight and 16, the similar threshold was 0.001 and the threshold distance was 16. The first experiments using HOG were carried out using a block size of 8.

The results of test image 1 can be seen in Fig. 10. A total of 58 similar blocks were detected, requiring 11 s of processing time.

The results of test image 2 can be seen in Fig. 11. Only four similar blocks were detected, requiring 11 s of processing time.

The results of test image 3 can be seen in Fig. 12. A total of 40 similar blocks were detected, requiring 11 s for this process.

In the second experiment involving HOG, we used a blocking size of 16. The results of test image 1 can be seen in Fig. 13. A total of 46 similar blocks were detected, requiring 10 s of processing time.

The results of test image 2 can be seen in Fig. 14. Eight similar blocks were detected, taking 10 s of processing time.

The results of test image 3 can be seen in Fig. 15. A total

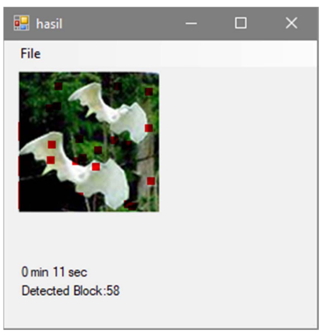


**Fig. 10.** Result of test image 1 using HOG with a block size of eight.

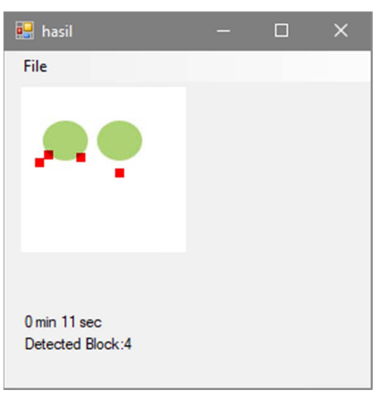


**Fig. 11.** Result of test image 2 using HOG with a block size of eight.

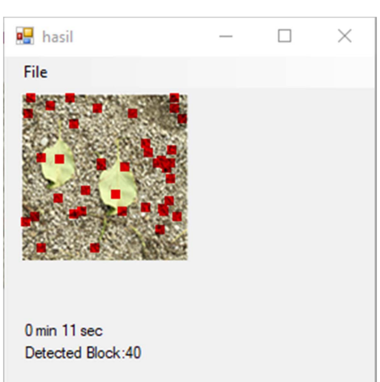


**Fig. 12.** Result of test image 3 using HOG with a block size of eight.

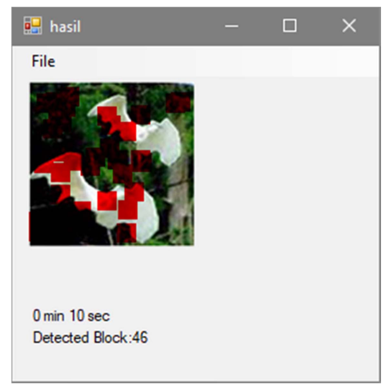


**Fig. 13.** Result of test image 1 using HOG with a block size of 16.

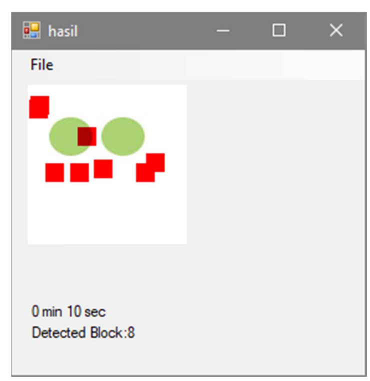


**Fig. 14.** Result of test image 2 using HOG with a block size of 16.

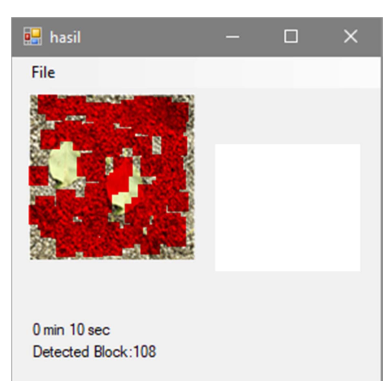


**Fig. 15.** Result of test image 3 using HOG with a block size of 16.

**Table 1.** Accuracy of the PCS method

| | 8×8 block size | 16×16 block size |
|---|---|---|
| Test Image 1 | (516-0)/516=95% | (403-0)/403=100% |
| Test Image 2 | (209-0) /209=100 % | (529-0)/529=100% |
| Test Image 3 | (276-70)/276=74% | (137-0)/137=100% |
| Average | 89.67% | 100% |

**Table 2.** Processing time for the PCS method

| | 8 × 8 block size | 16 × 16 block size |
|---|---|---|
| Test Image 1 | 1 m 5 s | 3 m 14 s |
| Test Image 2 | 1 m 10 s | 3 m 15 s |
| Test Image 3 | 1 m 41 s | 5 m 2 s |
| Average | 1 m 18 s | 3 m 50 s |

**Table 3.** Memory usage in the PCS method

| | 8 × 8 block size | 16 × 16 block size |
|---|---|---|
| Test Image 1 | 337.7 MB | 312.9 MB |
| Test Image 2 | 321.4 MB | 316.7 MB |
| Test Image 3 | 321.2 MB | 366.0 MB |
| Average | 327.8 MB | 331.9 MB |

**Table 4.** Accuracy of the HOG method

| | **8 × 8 block size** | **16 × 16 block size** |
|---|---|---|
| Test Image 1 | 5/58 = 8% | 9/46 = 19% |
| Test Image 2 | 2/4 = 50% | 1/8 = 12.5% |
| Test Image 3 | 4/40 = 1% | 9/108 =8.3% |
| Average | 20% | 13% |

**Table 5.** Processing time for the HOG method

| | **8 × 8 block size** | **16 × 16 block size** |
|---|---|---|
| Test Image 1 | 11 s | 10 s |
| Test Image 2 | 11 s | 10 s |
| Test Image 3 | 11 s | 10 s |
| Average | 11 s | 10 s |

**Table 6.** Memory usage in the HOG method

| | **8 × 8 block size** | **16 × 16 block size** |
|---|---|---|
| Test Image 1 | 350.6 MB | 370.3 MB |
| Test Image 2 | 349.7 MB | 371.3 MB |
| Test Image 3 | 351.0 MB | 368.6 MB |
| Average | 350.4 MB | 370.1 MB |

of 108 similar blocks were detected, requiring 10 s of processing time.

### *C. Comparison of Results*

The results for the accuracy of PCS can be seen in Table 1. This accuracy is calculated based on the number of correct blocks detected minus the number of wrong blocks detected, divided by the total number of blocks detected. In Fig. 8, not all of the balls were detected because the block is exactly the same; the PCS method therefore obtains a similarity result of one, and the object is not considered to have undergone copy-move.

The results for processing time can be seen in Table 2. The processing times for a block size of 16 are longer than for a size of eight, since each block will have a higher content, thus prolonging the process.

The results for memory usage can be seen in Table 3. Memory usage for a block size of 16 is more efficient, because the arrays required for a block size of 16 are less than the arrays required for a block size of eight.

The results for the accuracy of the HOG method can be seen in Table 4. In the same way as for the calculation for PCS, the results for the accuracy of HOG is calculated based on the number of correct blocks detected minus the number of wrong blocks detected, divided by the total number of blocks detected.

The results for processing time can be seen in Table 5. The processing time for a block of size eight is similar to that for a block size of 16.

The results for memory usage can be seen in Table 6. Memory usage for HOG using a block size of 16 is higher than for a block size of eight, due to the size of the array required by HOG for a block size of 16.

## VII. CONCLUSION

Based on the results of our analysis of the implementation and testing of these two methods, we can draw several conclusions. In terms of accuracy, the PCS method is far superior to the HOG method. Although there are some instances of false detection in the PCS method, the accuracy level is still relatively high, while the HOG method fails to detect many object areas. In terms of processing time, the HOG method is much faster than the PCS method. The memory usage of the two methods are almost the same, although PCS uses slightly more memory than HOG.

## REFERENCES

[1] S. M. Fadl and N. A. Semary, "Robust copy–move forgery revealing in digital images using polar coordinate system," *Neurocomputing*, vol 265, pp. 57-65, 2017. DOI: 10.1016/j.neucom.2016.11.091.

[2] J-C. Lee, C-P. Chang, and W-K. Chen, "Detection of copy–move image forgery using histogram of orientated gradients," *Information Sciences—Informatics and Computer Science, Intelligent Systems, Applications: An International Journal*, vol 321, pp. 250-262, 2015.

DOI: 10.1016/j.ins.2015.03.009.

[3] Video Communication Laboratory, "CoMoFoD - image database for copy-move forgery detection," [Internet], Available: http://www.vcl.fer.hr/comofod/examples.html.

[4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed, Pearson, 2017.

[5] S. Levy, "Relations between cartesian, cylindrical, and spherical coordinates," 1995, [Internet], Available: http://www.geom.uiuc.edu/docs/reference/CRC-formulas/node42.html.

[6] G. Lynch, F. Y. Shih, amd H. Y. M. Liao, "An efficient expanding block algorithm for image copy-move forgery detection," *Information Sciences*, vol 239, pp.253-265, 2013. DOI: 10.1016/j.ins.2013.03.028.

[7] W.Drongelen, *Signal Processing for Neuroscientists*, Elsevier, 2007.

[8] S. Mallick, "Histogram of oriented gradients," 2016, [Internet], Available: https://www.learnopencv.com/histogram-of-oriented-gradients/.

**Kartika Gunadi**

Kartika Gunadi is currently a senior lecturer in Informatics Department, Petra Christian University, Surabaya, He received his bachelor degree in Civil Engineering from Petra Christian University Surabaya, Indonesia and master degree in Information Technology from Sepuluh November Institute of Technology, Surabaya, Indonesia. His research interests are image processing and computer vision.

**Rudy Adipranata**

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 business information system.

**Ivan Suryajaya**

Ivan Suryajaya received his bachelor degree in Informatics from Petra Christian University, Surabaya, Indonesia. His research interests is image processing.