International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 17 (2015) pp 37731-37736
© Research India Publications. http://www.ripublication.com

Multi Scale Performance of Feature Extraction for Human Head Recognition

Panca Mudjirahardjo, M. Fauzan Edy Purnomo, Rini Nur Hasanah, Hadi Suyono
Department of Electrical Engineering, Faculty of Engineering, Universitas Brawijaya.
Jl. MT. Haryono 167, Malang, 65145, Indonesia.
e-mail: {panca, mfauzanep, rini.hasanah, hadis}@ub.ac.id

Abstract
Feature extraction plays an important role in head recognition. It transforms an original image into a specific vector to be fed into a classifier. An original image cannot be further processed directly. Raw information in an original image does not represent a specific pattern and a machine cannot understand that information. In this paper, we evaluate the multi scale performance of feature extraction method for human head recognition. We perform a comparison of the existing image features extraction and our novel methods using a static image. The existing features are HOG and LBP, and the novel feature is *a histogram of transition*. To extract the image feature, we use a normal size of image weight and height of 20 and 30 pixels, respectively. Then we evaluate the image with size of less than and greater than the normal size. We employ an algorithm to increase or reduce the image size into the normal size. The recognition rates using the proposed feature are that the head recognition rate is 91% and the non-head recognition rate is 99.7%. The execution time is 0.077 ms. These performances show that the proposed feature can be used for real time application.

Keywords: Head recognition; transition feature; histogram of transition; HOG; LBP.

Introduction

Head detection and recognition have been an important research in the last few years. Many applications use this research, such as robotics, automated room monitoring, people counting, person tracking, etc. Many new methods are introduced in this field, to improve the computation time and the recognition rate. One of them is the method based on feature extraction.
Feature extraction plays an important role in head recognition. It transforms an original image into a specific vector to be fed into a classifier. An original image cannot be further processed directly. Raw information in an original image does not represent a specific pattern and a machine cannot understand that information.
In an image, there are foreground and background patterns. In a simple image, foreground and background pattern can be separated clearly. In a complex image, however, foreground and background pattern cannot be separated clearly. There are many texture patterns both on foreground and background. Sometimes, both foreground and background contain similar texture and color on them. This is a difficult task in a head detection and recognition system. The system has to recognize a foreground pattern as a head or a non-head. Correct choice of a foreground extraction method will increase the recognition rate.
A feature is assumed to be able to distinguish a foreground and a background pattern. All of features distinguish a foreground pattern over the background from the edge pattern of the foreground, since a foreground has a specific edge pattern over the background.
Currently the most commonly used feature extraction methods are Histogram of Oriented Gradients (HOG)[1] and Linear Binary Pattern (LBP)[2]. The new feature extraction is a histogram of transition as our novel method [3]. This feature is relied on a background extraction. The simple method to extract a foreground is by using a difference function. Where we define some pixels as foreground pixels, then we calculate all pixel intensity with respect to the foreground pixels. If the difference result is less than or equal to the threshold, then the pixel is consider as foreground, otherwise is as background.
The overview of this experiment is shown in **Fig. 1**. The structure of this paper is as follows. Section 2 explains the preprocessing. Three feature extraction methods are overviewed in section 3. Experimental results are shown in section 4. Finally the paper is concluded in section 5.

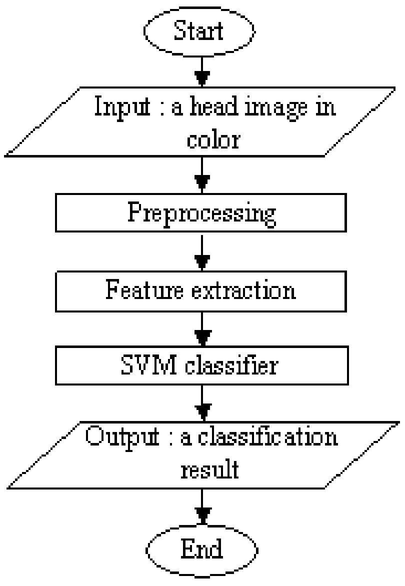
Start
Input : a head image in color
Preprocessing
Feature extraction
SVM classifier
Output : a classification result
End

Fig. 1. Overview of the proposed method.

Preprocessing

In this section, we normalize the image size into the normal size 20×30 pixels. For HOG and LBP feature, the input

image is grayscale. For Histogram of Transition feature, we convert the RGB image into HSV image, extract the foreground then compute the image feature.
Algorithm to increase the image size 10×15 pixels into 20×30 pixels (twice scaling) is depicted in **Fig 2**, as follows:

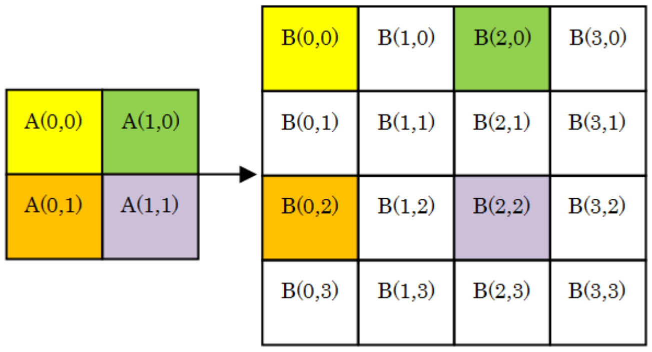

Fig. 2. Illustration to increase the image size

Initially for all (colB = even and rowB = even), we set:

$$I_{B(\text{colB, rowB})} = I_{A(\text{colB}/2,\ \text{rowB}/2)} \quad \text{(1-a)}$$

if (colB = odd and rowB = even), then

$$I_{B(colB,rowB)} = \frac{I_{B(colB-1,rowB)} + I_{B(colB+1,rowB)}}{2} \quad \text{(1-b)}$$

if (colB = even and rowB = odd), then

$$I_{B(colB,rowB)} = \frac{I_{B(colB,rowB-1)} + I_{B(colB,rowB+1)}}{2} \quad \text{(1-c)}$$

if (colB = odd and rowB = odd), then

$$I_{B(colB,rowB)} = \frac{I_{B(colB-1,rowB-1)} + I_{B(colB+1,rowB+1)}}{2} \quad \text{(1-d)}$$

if (colB = end of colB),

then
$I_{B(\text{colB, rowB})} = I_{B(\text{colB-1, rowB})}$

if (rowB = end of rowB),

then
$I_{B(\text{colB, rowB})} = I_{B(\text{colB, rowB-1})}$

where $I_{B(x,\,y)}$: pixel intensity of image B at pixel location (x, y).
The algorithm to reduce the image size 40×60 pixels into 20×30 pixels (half scaling) is depicted in **Fig. 3**, as follows:

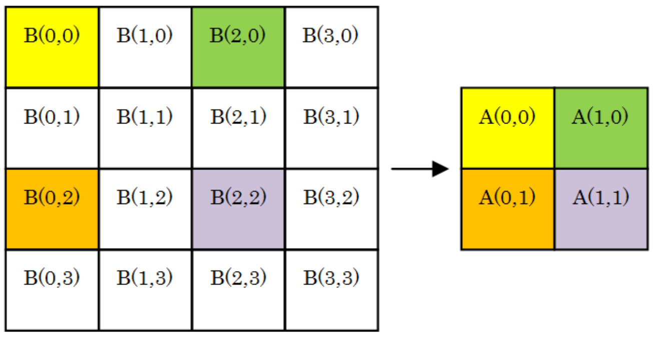

Fig. 3. Illustration to reduce the image size

$$I_{A(n,\,m)} = \max(I_{B(n,\,m)}, I_{B(n+1,\,m)}, I_{B(n,\,m+1)}, I_{B(n+1,\,m+1)},). \quad (2)$$

In **Fig. 2** and **Fig. 3**, the same color means the same pixel intensity.

Feature Extraction

A. *Histogram of Oriented Gradients [1][2]*

This feature is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. The basic idea is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions. In practice this is implemented by dividing the image window into small spatial regions ("*cells*"), for each cell accumulating a local 1-D histogram of gradient directions or edge orientations over the pixels of the cell. The combined histogram entries form the representation. For better invariance to illumination, shadowing, *etc*., it is also useful to contrast-normalize the local responses before using them. This can be done by accumulating a measure of local histogram "energy" over somewhat larger spatial regions ("*blocks*") and using the results to normalize all of the cells in the block. This will refer to the normalized descriptor blocks as *Histogram of Oriented Gradient (HOG)* descriptors. The overview of HOG feature extraction is depicted in **Fig. 4**.

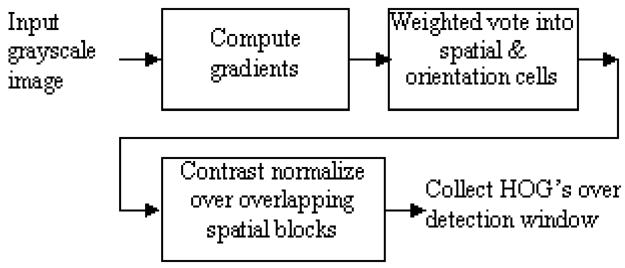
Input grayscale image
Compute gradients
Weighted vote into spatial & orientation cells
Contrast normalize over overlapping spatial blocks
Collect HOG's over detection window

Fig. 4. The overview of HOG feature extraction

We use a simple 1-D [-1, 0, 1] mask to compute the image gradients and no pre-smoothing of the image is applied, both in horizontal and vertical directions. The gradient in horizontal direction, I_x, and the gradient in vertical direction, I_y, are defined as follows,

$$\begin{bmatrix} I_x \\ I_y \end{bmatrix} = \begin{bmatrix} I(x-1,y) & I(x,y) & I(x+1,y) \\ I(x,y-1) & I(x,y) & I(x,y+1) \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \tag{3}$$

$$|I| = \sqrt{I_x^2 + I_y^2} \tag{4}$$

$$\theta = \tan^{-1} \frac{I_y}{I_x} \tag{5}$$

where I (x, y) is the intensity of a grayscale image at location x-th column and y-th row. $|I|$ is a gradient module magnitude and θ is a gradient module orientation.

At the next step, a clustering process based on the gradient module orientation is done. In practice, the gradient image is split into 8 image bins, each one representing an orientation with a certain range within 0^o to 360^o. For every pixel, the corresponding module of the gradient is stored in the appropriate orientation image bin. The orientation image bin is shown in **Fig. 5**.

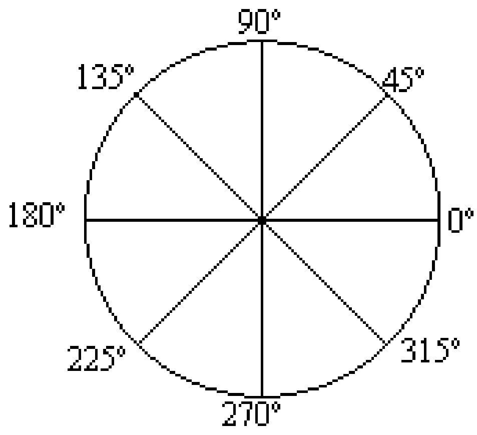

Fig. 5. The orientation image bin

At this point, we calculate the histogram of oriented gradient in the square-cells. The cell size is 5×5 pixels. Each cell is calculated directly from the gradient image and they are not overlapping.

Then we perform contrast normalization over overlapping spatial blocks. Block size is 15×15 pixels or 3×3cells. The overlapped pixels are 5 pixels. We use normalization as follows;

$$L2-norm: \boldsymbol{v} = \frac{\boldsymbol{v}}{\sqrt{\|\boldsymbol{v}\|_2^2 + \varepsilon^2}} \tag{6}$$

Here ε is a small constant larger than zero.

*B. **Local Binary Pattern [3]***

The local binary pattern (LBP) is a non-parametric operator which describes the local spatial structure of an image. Ojala et al. [4] first introduced this operator and showed its high discriminative power for texture classification. At a given pixel position (x, y), LBP is defined as an ordered set of binary comparisons of pixel intensities between the center pixel and its eight surrounding pixels, as shown in **Fig.6**. The decimal form of the resulting 8-bit word (LBP code) can be expressed as follows;

$$LBP(x,y) = \sum_{n=0}^{7} s(i_n - i_c)2^n \tag{7}$$

where i_c corresponds to the grey value of the center pixel (x, y), i_n to the grey values of the 8 surrounding pixels, and function $s(x)$ is defined as

$$s(x) = \begin{cases} 1 & \text{if} \quad x \geq 0 \\ 0 & \text{if} \quad x < 0 \end{cases} \tag{8}$$

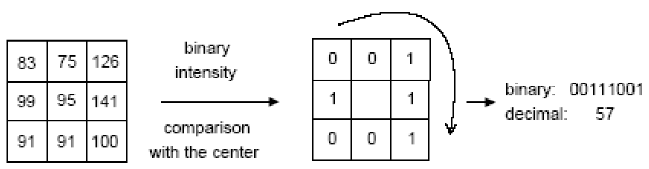

Fig. 6. The LBP operator

By definition, the LBP operator is unaffected by any monotonic gray-scale transformation which preserves the pixel intensity order in a local neighborhood. Note that each bit of the LBP code has the same significance level and that two successive bit values may have a totally different meaning. Actually, The LBP code may be interpreted as a kernel structure index.

An image is usually divided into small regions. For each region, a cumulative histogram of LBP codes, computed at each pixel location within the region, is used as a feature vector.

*C. **Histogram of transition***

Another feature is the novel method, a histogram of transition [5][6]. As the first step to create a histogram of transition, we calculate a transition feature. A transition feature is to compute the location and the number of transitions from background to foreground along horizontal and vertical lines. So, this transition feature relies on foreground extraction. **Fig. 7** shows the overview of creating a histogram of transition.

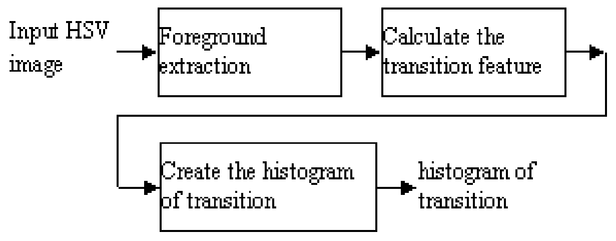
Input HSV image
Foreground extraction
Calculate the transition feature
Create the histogram of transition
histogram of transition

Fig. 7. The overview of histogram of transition feature extraction

Fig. 7. The overview of histogram of transition feature extraction

In this paper, we consider to use a simple foreground extraction. A simple algorithm to extract foreground is that we determine some reference pixel coordinates as foreground [5][6]. Then we compare another pixel's intensity (I_x) to the reference pixel's intensity (I_R). Since we extract foreground in an RGB image, we have pixel's intensity in red, green and blue. We determine a pixel at coordinate (x, y) as foreground or background by equation (9),

$$I(x,y)=\begin{cases}\text{foreground} & \delta(I_x,I_R)<th\\ \text{background} & \text{otherwise}\end{cases} \qquad (9)$$

where $\delta(.)$ is a distance function. In this paper, we use Euclidean distance.

After we get the foregrounds, then we extract the feature of these foregrounds. Our feature refers to [7][8]. Transition feature has been used successfully in handwritten characters recognition, but it hasn't been used in head detection yet. Due to a simple calculation to create a feature vector, we apply the idea to head recognition. We do some modifications on it to be able to be used for head recognition.

The idea is to compute the location and the number of transitions from background to foreground along horizontal and vertical lines. This transition calculation is performed from right to left, left to right, top to bottom, and bottom to top. Since a constant dimensional feature is required as input to the SVM classifier, an encoding scheme is developed.

In the first stage of feature extraction, the transition in each direction is calculated. Each transition is represented as a fraction of the distance across the image in the direction under consideration. These fractions are computed in the increasing order, differed from [8] in decreasing order. For example, when calculating the location of transitions from left-to-right, a transition close to the left edge would have a low value and a transition far from the left edge would have a high value as illustrated in **Fig 8.**

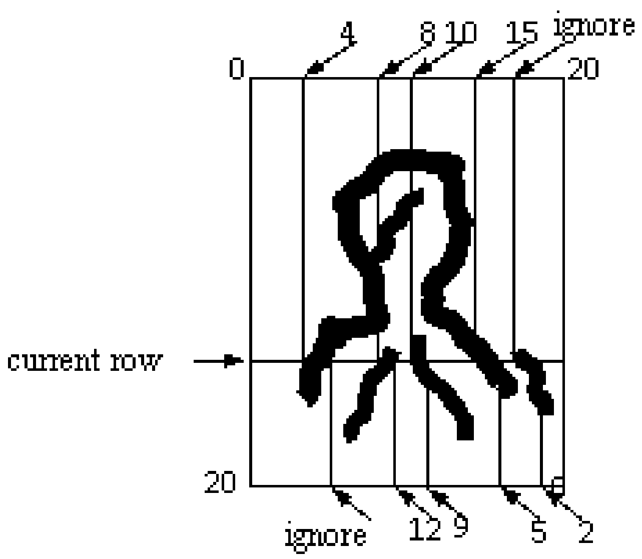

Fig.8. The first stage of transition feature extraction shown for transitions from the left and from the right on one row of the image, with M = 4.

The maximum number of transitions, M, are counted on each line. If there are more than M transitions in a line, only the first M are counted and the rest are ignored. M is set to 4 in the experiment. If there are less than M transitions on a line, then the "nonexistent" transitions are assigned as a value of 0. More precisely, by a line we mean a row or a column of the head image. Let h be the height of the image and w be the width of the image. We assign exactly M values to each line, say t_1, t_2, …, t_M. We assume that there are n transitions on a line located at (x_i, y_i) for i = 1, 2, … n. The algorithm for calculating the transition feature can be represented as follows: It doesn't require normalization as in [8]:

```
for $i$ = 1 to min ($n$, $M$)
    if the line is a row then
        $t_i = x_i$;
    else
        $t_i = y_i$;
    end if;
end for;
if $n < M$ then
    For $i = n+1$ to $M$
        $t_i = 0$;
    end for;
end if;
```

The transitions are resampled to a 4-point sequence for each direction and assembled into a feature vector. The four transitions for each row (column) are represented as two-dimensional (2-D) array, $t = [t_{ij}]$ for i = 1, …, h(w) and j = 1, …, 4.

The second stage is generating a histogram of transition. It is different from [8] where they calculated local averaging on the columns of t. Histogram of transition shows how often the location of the transition occurs at each transition. An example of generating a histogram of transition for transition left-to-right is shown in **Fig. 9.**

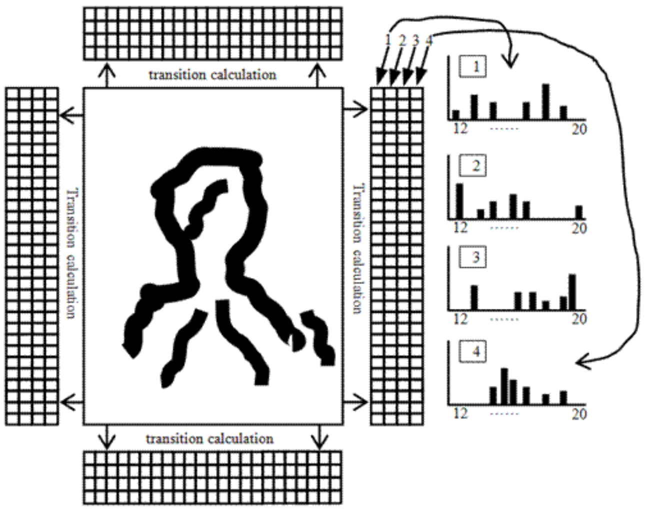

Fig. 9. The second stage of transition feature calculation consisting of generating a histogram of transition

This histogram of transition creates a feature vector to be fed into the input of a SVM classifier [9].
The HOG feature contains gradient information of a pixel among its neighbor. Thus they give a high magnitude at the edge. On the other hand, the LBP feature gives a binary pattern with a pixel among its neighbor. The histogram of transition feature looks like the HOG feature: It gives the edge position from right, left, top and bottom side. In contrast to the HOG feature, the calculation of the histogram of transition feature is simpler.

Experimental Results
The experimental environment is as follows: Operating system is Windows 7 professional; the processor is Intel® core™ i7 CPU 870 @2.93GHz and the used software is Microsoft Visual Studio 2010.
For robust detection, we use backgrounds and negative samples at outdoor scenery. We use INRIA data [1][10] for training and testing images. For training, we use image size 20×30 pixels, positive sample of 2, 000 images, negative sample of 4, 500 images. For testing, we use image size of 10×15 pixels, 20×30 pixels and 40×60 pixels. Positive sample of 100 images, negative sample of 300 images are employed. **Fig. 10** shows some positive and negative samples.

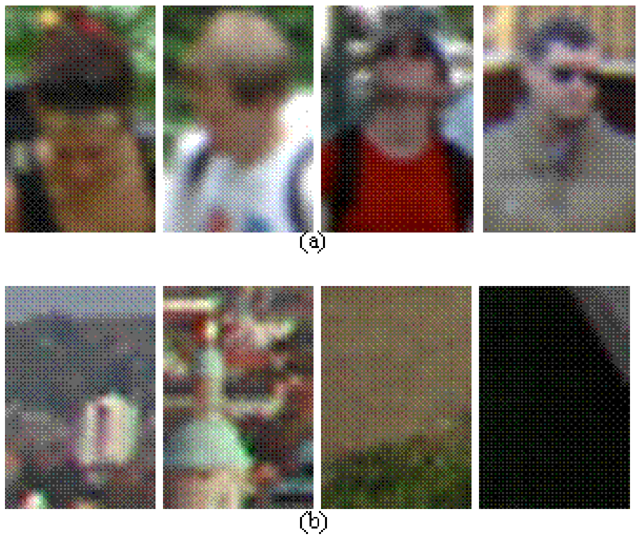

Fig. 10. Samples for SVM training: (a) positive samples, (b) negative samples

In this research, we use 9 bins to cluster the gradient image. One is for value I_x and I_y equals zero, the other 8 bins for representing the orientation from 0^o to 360^o. For image size 20×30, cell size 5×5, and block size 15×15 with overlapped pixels of 5 pixels, the number of feature dimensions equals

the_number_of_bins × *the_number_of_cells_in_a_block* × *the_number_of_blocks*, which amounts to 648 dimensions.

In using the LBP operator, we divide an image into four non-overlapping regions. The number of feature dimensions equals

the_number_of_region × *max_number_of_decimal _value*,

which amounts to 1020 dimensions.
For the image preprocessing to extract transition feature, we extract the foreground using a difference function Eq. (9). First, we determine 32 reference pixel coordinates as foreground. These coordinates are fixed for all the training and the test data. The coordinates should represent position of head and shoulder. Then, we check all pixels' intensity to the five reference pixel's intensity with Euclidean distance, by Eq. (9). If a pixel's intensity distance to the one or more of five reference pixel's intensity is less than a threshold, the pixel should be a foreground pixel, otherwise a background pixel.
In this research, the maximum number of transition, *M*, is 4. The number of feature dimension,

2×((*M*×*width_of_image*)+ (*M*×*height_of_image*)), are 400 dimensions.

The result of head recognition is summarized in **Table I**.

TABLE I. **THE RESULT OF HEAD RECOGNITION**

Feature	**The number of array**	**Recognition rate (%)**						**Execution time (ms)**
		Small size (10×15)		**Normal size (20×30)**		**Big size (40×60)**		
		positive	**negative**	**Positive**	**negative**	**positive**	**negative**	
HOG	648	83	98.00	84	98.30	84	98.67	0.353
LBP	1020	92	61.00	**95**	80.00	**96**	61.33	0.261
Histogram of transition	400	**92**	**99.70**	92	**99.70**	92	**99.67**	0.087

Conclusion

In this paper, we evaluate the performance of multi scale of input image, then we extract the feature and perform a comparison of the existing image features extraction methods using a static image. The existing features are HOG and LBP, and the proposed feature is a histogram of transition.

In design, the proposed feature is robust for multi scale image and has the acceptable recognition rate compared to the existing features.

The proposed feature has the number of array less than the existing features, and the computation of feature transition is simpler than the existing features. These conditions give the computation of the proposed feature faster than the computation of existing features. This performance shows that the proposed feature can be used for real time application.

As future work, we are going to conduct experiments to improve foreground extraction.

References

[1] N. Dalal, B. Triggs. "Histograms of oriented gradients for human detection", Proceeding of Computer Vision and Pattern Recognition, Vol. 1, pp. 886-893, 2005.

[2] M. Perdersoli, J. Gonzalez, B. Chakraborty, J. Villanueva. "Boosting histograms of oriented gradients for human detection", In: Proc. 2nd Computer Vision: Advances in Research and Development (CVCRD), pp. 1–6, 2007.

[3] G. Heusch, Y. Rodriguez, S. Marcel. "Local binary patterns as an image preprocessing for face authentication", Proceeding of Automatic face and Gesture Recognition (FGR 2006), pp. 9-14, 2006.

[4] T. Ojala, M. Pietik¨ainen, D. Harwood. "A comparative study of texture measures with classification based on featured distributions", Pattern Recognition, 29(1), pp. 51–59, 1996.

[5] P. Mudjirahardjo, J.K. Tan, H. Kim, S. Ishikawa. "Head detection and tracking for an intelligent room", Proc. SICE Annual Conference 2014, pp. 353-358, 2014.

[6] P. Mudjirahardjo, J.K. Tan, S. Ishikawa. "A study on human motion detection---Toward abnormal motion identification", Ph.D. Thesis, Kyushu Institute of Technology, Japan, pp. 35-55, 2015.

[7] P. Mudjirahardjo. "Penerapan jaringan perambatan balik untuk pengenalan kode pos tulisan tangan" (The implementation of back-propagation network for recognition of handwritten post code), Master Thesis, Universitas Gadjah Mada, pp. 1-100, 2001.

[8] P.D. Gader, M. Mohamed, J.H. Chiang. "Handwritten word recognition with character and inter-character neural networks", IEEE Trans. On Systems, Man, and Cybernetics – Part B: Cybernetics, Vol. 27, No. 1. pp. 158-164, 1997.

[9] C.W. Hsu, C.C. Chang, C.J. Lin. "A practical guide to support vector classification", Proceeding of IEEE Intelligent Vehicles Symposium, Vol. 1, pp.1-6, 2010.

[10] http://pascal.inrialpes.fr/data/human/