Vol.2012-CVIM-182 No.35
2012/5/24

Binary code-based Human Detection

Yuji Yamauchi[1,a)] Hironobu Fujiyoshi[1,b)]

Abstract: HOG features are effective for object detection, but their focus on local regions makes them high-dimensional features. To reduce the memory required for the HOG features, this paper proposes a new feature, R-HOG, which creates binary codes from the HOG features extracted from two local regions. This approach enables the created binary codes to reflect the relationships between local regions. Converting feature values to binary, however, results in the loss of much information included in the features. In response to this problem, we have been focusing on " quantization residual " information that is lost at this time. In this study, we introduce a transition likelihood model into the classifier based on two ideas using quantization residuals to consider the possibility that a binary code observed from the image will make a transition to another binary codes. This enables classification that takes into account all binary codes including the originally desired binary codes even if an observed binary code differs from the truly desired binary codes due to some sort of effect from another binary codes. Experimental results show that a classifier equipped with transition prediction based on quantization residuals as proposed here achieves high-accuracy human detection compared to the same classifier without transition prediction.

1. Introduction

With the increasing use of digital cameras and vehicle-mounted cameras, the expectations for practical detection of humans in image for the purposes of improving image quality and assisting the drivers of vehicles are also rising. Research on the use of Field Programmable Gate Arrays (FPGA) or other such hardware implementations of that function has been done [3], [6], [12]. In hardware implementations, it is important that the detection method can operate with high accuracy, high-speed and low memory requirements.

Most detection methods proposed in recent years use combinations of local features of images and stochastic learning [2], [7], [16], [18], [19]. Local region gradients [1], [5], which are used as features in many proposals, can capture the shape of an object, but very many dimensions are required to obtain the features of each local region. The difficulty of accomplishing that with small-scale hardware that has limited memory is a major problem whose solution requires a reduction in the amount of feature data. Less data has two benefits. One is that less memory is needed and the other is that features can be categorical, each representing common properties.

Two approaches to reducing the amount of data can be considered: compressing the feature space to reduce the number of features and reducing the amount of data needed for each feature. The former approach includes methods such as vector quantization to reduce the number of features [9] and principle component analysis to compress the feature dimensions. These methods can retain the original amount of data while reducing the number of feature dimensions. Human detection, however, involves the processing of a huge number of detection windows, so these methods are very inefficient.

The latter approach involves quantizing features at a low bit rate. Scalar quantization, for example, can represent the feature data at a bit rate that fits the problem. Quantization is also an effective way to reduce the amount of data. In addition to representing the information with the minimum amount of data, it has the advantages of being robust against noise and easy to use. One method of quantization is threshold processing, which is simple and has the advantage of low computational cost. However, determining the optimum threshold for many samples is difficult. Another binarization method uses the size relationship. The Local Binary Pattern (LBP) [11], [13], [17] and a method that expands on that [4] have the advantage of not requiring a threshold, as binarization is based on comparison of the two values. Threshold binarization and size relationship binarization also differ in the data contained in one binary value. In threshold processing, the value represents only size, but when size relationship is used, the relation between two values is also included.

Our method focuses on binarization using the size relationship, which is one of the latter methods of reducing data quantity. To achieve highly accurate object detection while reducing the amount of feature data, we propose the Relational HOG (R-HOG) feature, a binarization method in which the size relationship is obtained by comparing HOG features from two local regions. Since R-HOG features uses the size relationship of two HOG features, they can represent

[1] Chubu University
1200 Matsumoto-cho, Kasugai, Aichi 487-8501, Japan
[a)] yuu@vision.cs.chubu.ac.jp
[b)] hf@cs.chubu.ac.jp

the relatedness of local regions.

However, the process of quantizing a features into binary form creates a problem in that a great deal of the information within the features drops out. If information divided into two classes is included in the missing information, the classification capability deteriorates when a binarized features are is used.

In this study, we focus on the information that drops out during the binarization of the feature. This missing information is called a quantization error that is defined as the difference in values before and after quantization, but in this study we define the missing information as " quantization residual "since the values after the quantization are " 0/1 ". A quantization residual indicates the difference between a real number value and a threshold value, when the values of a feature represented by real numbers are subjected to threshold value processing, by way of example. One characteristic of quantization residuals is that binary encoding that is stable and less likely to invert is obtained when the quantization residual is larger. If the quantization residual is small, on the other hand, unstable binary encoding in which inversions can easily occur are obtained. Binary codes that is expressed by combining a number of binary encodings is a discrete variable that forms another feature by simply inverting one binary code. For that reason, it can happen that even when features are similar when represented by real numbers, they are not observed to be the same binary code when binarized. Essentially, since it is necessary to extract elements that are common to the detection objects, in order to detect humans highly accurately, such representations of features are not suitable.

That is why, in this study, a transition likelihood model that represents the relationships between binary code based on quantization residuals is introduced into classifiers. A transition likelihood model represents the degree of likelihood of an observed binary code $\boldsymbol{x}$ transitioning to another binary code $\boldsymbol{x}$'. This study uses transition likelihood distributions created on the basis of binary code and quantization residuals obtained from training samples, as a transition likelihood model. This predicts transitions from an observed binary code to another binary code, in accordance with the transition likelihood distribution created during the classification. By introducing this binary code transition prediction into classifiers, it becomes possible to take into consideration transitions to the originally obtained binary code, even if the observed binary code differs from the actually desired binary code for some reason.

This paper is organized as follows. We summarize related works in Section 2. We describe HOG feature and binary code in Section 3, and report the experimental results in Section 4. We discuss the transition likelihood model and the classifiers that this model has been introduced into in Section 3, and we describe evaluation experiments that we performed in order to confirm the validity of the proposed method in Section 4. Finally, we summarize this paper in Section 5.

2. Related works

Local Binary Patterns (LBP) are a technique that is being applied in a variety of fields such as object detection, face recognition, and action recognition. The LBP features proposed by Ojala *et.al.* [13] represent the magnitude relations of a pixel to adjacent pixels with a code, thus allowing representations of fixed shapes such as edges. Liao *et.al.* proposed a method for application to face recognition in which LBP is extended to represent magnitude relations between the average luminance values in multi-resolution block areas [8]. Another technique applied in face recognition[15] and action recognition [21] is the Local Ternary Pattern (LTP), which extends LBP by using three values for the threshold. Although that method can capture the relations of nearby and adjacent pixels, it cannot represent other effective combinations that exist. Furthermore, the LTP approach requires optimum values for the three thresholds.

Methods that combine multiple feature quantities include the Joint Haar-like features proposed by Mita *et.al.*, which capture the relations of Haar-like features in different positions [10], and co-occurrence features that link the output values of weak classifiers with operators proposed by Yamauchi *et.al.* [20]. These methods combine feature quantities on the basis of recognition results, so the combination of features may be negatively affected when the results are in error or when the target of detection is obscured.

3. HOG features and binary code

This section describes HOG features and binarization as means of reducing the amount of HOG feature data.

3.1 HOG features

The Histograms of Oriented Gradients (HOG) feature proposed by Dalal *et.al.* [2] is a one-dimensional histogram of gradient orientations of intensity in local regions that can represent object shape. This feature is a histogram of adjacent pixel gradients for local regions, so it is not easily affected by local lighting conditions and is robust to changes in geometry.

To compute HOG features, first, the magnitude m and gradient orientation θ are calculated from the intensity I of the pixels. Next, using the calculated magnitude m and gradient orientation θ, the sum of the magnitudes of quantized gradient orientation θ' in cell region c ($p \times p$ pixels) are calculated. We represent the set of sums of magnitude in gradient orientation θ' as the N-orientation histogram $\boldsymbol{V}_c = \{v_c(1), v_c(2), \cdots, v_c(N)\}$. Finally, we use Eq. (1) to normalize the histogram by each block region ($q \times q$ cells) to extract the features.

$$v'_c(n) = \frac{v_c(n)}{\sqrt{\left(\sum_{k=1}^{q \times q \times N} v_c(k)^2\right) + \epsilon}} \quad (\epsilon = 1) \tag{1}$$

After normalization, the histogram $\boldsymbol{v}'_c$ is $\boldsymbol{v}'_c = \{v'_c(1), v'_c(2), \cdots, v'_c(B \times N)\}$. Here, B is the number

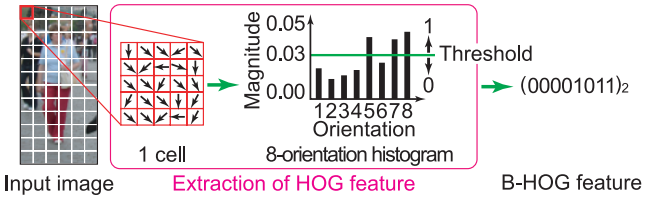

Fig. 1 B-HOG feature calculation method.

of cell regions that are contained in the block region.

3.2 Binarized HOG features (B-HOG)

Binarized HOG (B-HOG) features are obtained by thresholding. Those features can capture the relatedness of the gradient orientations within the cell region by observing the binary values for N orientations in the cell region as a single feature (binary pattern).

We use the eight-orientation histogram $\boldsymbol{v}'_c$ for the cell region subjected to threshold processing as shown in Eq. (2) to produce the B-HOG features $\boldsymbol{x}_c^B = \{x_c^B(1), x_c^B(2), \cdots, x_c^B(8)\}$. In reference [2], nine orientations are used, but in our work we choose eight orientations so that features can be represented as one byte.

$$x_c^B(n) = \begin{cases} 1 & \text{if } v'_c(n) \geq t \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Here, the t represents the threshold. For example, when extracting HOG features for an input image such as Fig. 1 and binarizing the features, we get $\boldsymbol{x}_c^B = (00001011)_2$.

3.3 Benefits and problems with B-HOG features

B-HOG features and HOG features vary with the amount of feature data. The HOG features obtained with Eq. (1) must usually be represented by double precision real numbers (8 bytes), but the B-HOG features can be represented by one unsigned character (1 byte). Thus, B-HOG features can reduce memory use to 1/8 that required by HOG features. However, the need to obtain the optimum binarization threshold values t for different human detection environments is a problem.

3.4 Relational HOG

Relational HOG(R-HOG) features are binarized by comparing the two values of HOG features obtained from two local regions as shown in Fig. 2, thus reducing data quantity by eliminating use of a threshold. While B-HOG features can represent gradient magnitudes only as binary values, R-HOG features can also represent the relationship between two features. Furthermore, R-HOG feature binarizes the size relation of HOG features, so processing to normalize the HOG is not needed. Because the normalization processing has the highest computational cost of the HOG feature processing, the proposed method can greatly reduce the processing cost.

R-HOG features are the binarized feature quantities $\boldsymbol{x}_{c_1c_2}^R = \{x_{c_1c_2}^R(1), x_{c_1c_2}^R(2), \cdots, x_{c_1c_2}^R(8)\}$ that result from comparing the size relationship of the eight-orientation histograms $\boldsymbol{v}_{c_1}$ and $\boldsymbol{v}_{c_2}$ obtained from two cell regions c_1 and c_2, as shown in Eq. (3).

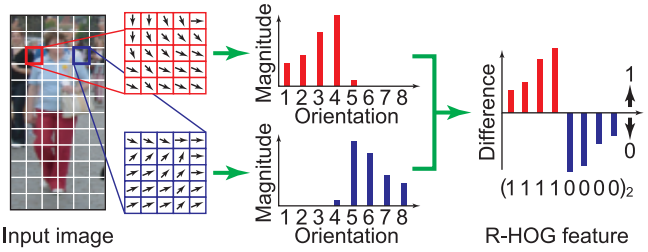

Fig. 2 Binarization using HOG features of two cell regions.

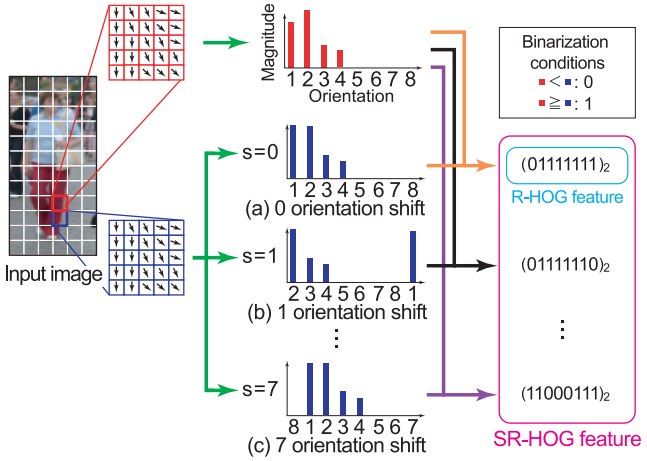

Fig. 3 Introducing a shift in the orientation.

$$x_{c_1c_2}^R(n) = \begin{cases} 1 & \text{if } v_{c_1}(n) \geq v_{c_2}(n) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Note that $v_c(n)$ does not require a normalization process. As we see in Fig. 2, we can create a binary pattern that captures the relatedness of local regions by using the size relationship of features in two cell regions. In doing so, the R-HOG features are computed from all combinations of cell regions. However, as shown in Fig. 3, if the extracted features are similar, their size relation is not distinct, and so is difficult to represent clearly as binary values.

3.5 Shifted Relational HOG features (SR-HOG)

To solve the problems associated with R-HOG features, as shown in Fig. 3 (b) and (c), we shift the orientation of the eight-orientation histogram $\boldsymbol{v}_{c_2}$ extracted from one of the cell regions by $s(s = 0, 1, 2, \cdots, 7)$ to create the eight histograms $\boldsymbol{v}_{c_2 s}$. Then, we use Eq. (4) in the same way as Eq. (3) to obtain the size relationship and calculate the eight binarized features, $\boldsymbol{x}_{c_1c_2s}^{SR}$.

$$x_{c_1c_2}^{SR}(n, s) = \begin{cases} 1 & \text{if } v_{c_1}(n) \geq v_{c_2}((n+s)\%8) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Here, % is the modulo operator. By calculating the size relationship with the orientation-shifted histograms, the size relationship can be represented clearly even if the extracted features are similar. In this paper, we refer to the R-HOG features extracted with orientation shifting as Shifted Relational HOG (SR-HOG) features.

Fig. 4 Examples of INRIA person dataset.

4. Evaluation Experiments

To evaluate the effectiveness of binary codes, we conducted experiment.

4.1 Dataset

We use the INRIA person dataset[2] as the dataset used in these evaluation experiments. The INRIA person dataset consists of 2,415 positive samples and 1,218 negative images for training, and 1,126 positive samples and 453 negative image for evaluation purposes. The positive images for training and evaluation are cropped to conform with regions containing, and are normalized to an image size of 64 × 128 pixels. The negative images for training and evaluation are images with no people in them. For the negative samples for training, we use ten sections taken at random from each image, giving a total of 12,180 samples. For the negative samples for evaluation, we perform a comprehensive raster scan on each evaluation image, giving approximately 2,000,000 samples.

Part of the INRIA person dataset is shown in Fig. 4. The images within the positive samples include people in various different poses and orientations, forming images with greatly differing viewpoints. This is an extremely challenging benchmark dataset.

4.2 Experiment outline

In our experiments, we compare HOG features and binarized HOG features. We use Real AdaBoost [14], which enables highly accurate and fast classifications, as the statistical training method. We use Detection Error Tradeoff (DET) curves in the comparison. A DET curve plots False Positives Per Window (FPPW) along the horizontal axis and miss rates along the vertical axis, with detection performance increasing with closeness to the origin at bottom left.

4.3 Experiment results

We tested the effectiveness of R-HOG and SR-HOG features. The DET curves that represent the results for the two datasets are presented in Fig. 5.

First, we compare the B-HOG features with the R-HOG features. Comparing the human detection rates for when the FPPW from Fig. 5 is 10^{-3}, the detection rate for the R-HOG features is about 7.5% higher than for the B-HOG features.

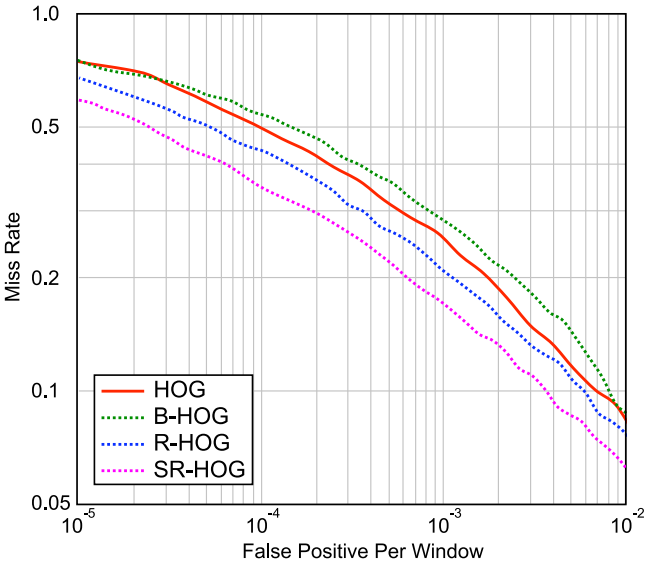

Fig. 5 DET curves of experiment.

Next, we compare R-HOG features and SR-HOG features. Comparing the people detection rates for when the FPPW from Fig. 5 is 10^{-3}, the detection rate for the SR-HOG features is about 6.8% higher than for the R-HOG features. From these results we know that R-HOG features, which are binary patterns obtained by size relationship, are superior to B-HOG features, which are binary patterns obtained by threshold processing, in capturing the relations between cell regions and thus provide a higher detection rate. Furthermore, shifting the gradient orientation of HOG features from one of the cell regions to obtain a binary pattern as is done for SR-HOG features clarifies the size relationship, so the performance is even higher than for R-HOG features.

Finally, we compare the R-HOG features and the SR-HOG features with the HOG features. For both human and vehicle detection, the detection rates for the R-HOG features are lower than for the HOG features. However, the detection rates for the SR-HOG features come closer to those for the HOG features, even though the feature data is reduced.

4.4 Comparison of memory and computational cost

Table 1 shows the memory use and computational cost required for feature extraction and classification in one detection window (64 × 128 pixels) for HOG features, B-HOG features, R-HOG features, and SR-HOG features, assuming 500 weak classifiers.

The R-HOG features used about 87.5% less memory than the HOG features, and the SR-HOG features used 75.0% less memory than the HOG features. Both the R-HOG features and the SR-HOG features reduced the computational cost by about 50.0% compared with the HOG features. This is because the R-HOG features and the SR-HOG features do not require normalization processing.

In general, binarization of features decreases the detection accuracy such as for B-HOG due to the reduction of

Table 1 Comparison of memory use and computational cost.

Feature	HOG	B-HOG	R-HOG	SR-HOG
Memory[KB]	3.91	0.50	0.49	0.98
Computational cost[ms]	5.39×10^{-7}	5.40×10^{-7}	2.70×10^{-7}	2.70×10^{-7}

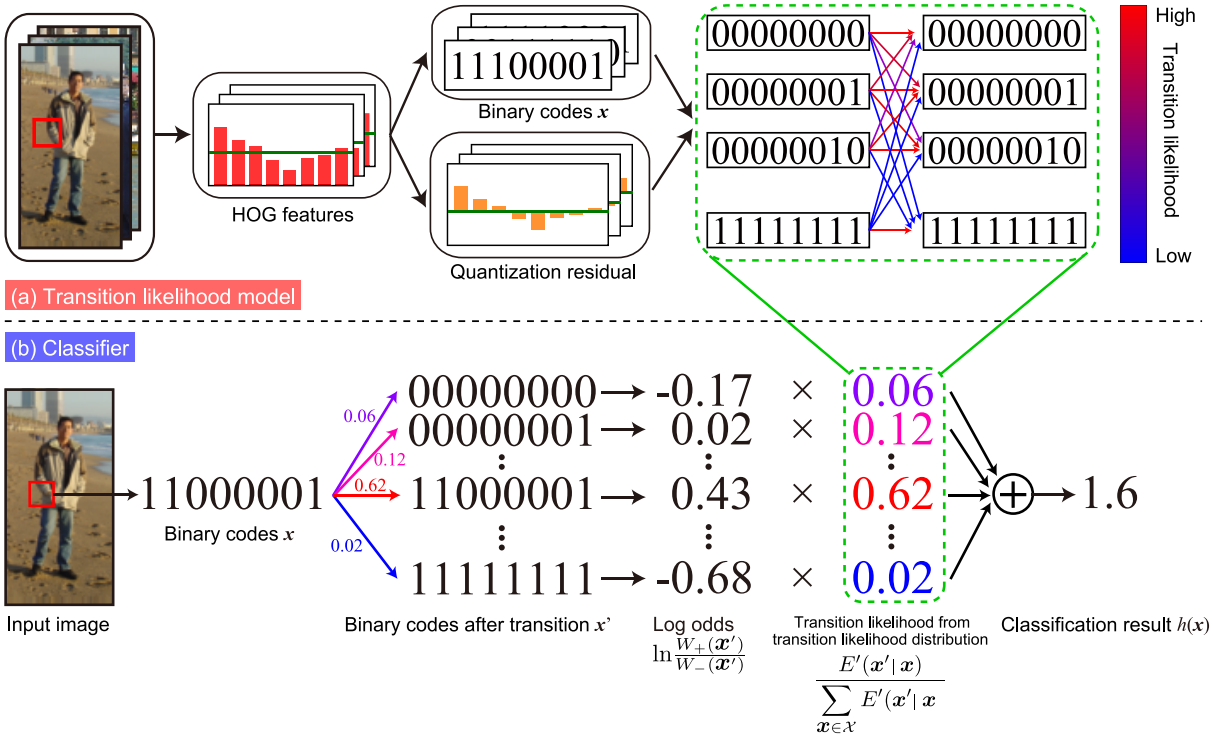

Fig. 6 Classification using transition prediction of a binary code.

information effective for classification. However, SR-HOG features are able to achieve higher discrimination accuracy than HOG features even though binarization is applied.

5. Classifier Introducing Transition Likelihood Model Based on Quantization Residual

Our approach introduces into classifiers a transition likelihood model created based on " quantization residuals ", which have not been used at all previously. The transition likelihood model outputs a transition likelihood that expresses the possibility of an observed binary code transitioning into another binary code. Since the proposed method makes it possible to represent relationships between binary code even though they are discrete variables, by considering these transition likelihoods during classification, it can output classification results that are even more reliable.

The flow of classification in accordance with the proposed method is shown in Fig. 6. The proposed method creates a transition likelihood model from binary code and quantization residuals obtained from training samples. In addition, human and non-human objects are classified by inputting into the classifiers based on the transition likelihood obtained from a binary code of an unknown input image and the transition likelihood model.

In this section, we first discuss problems of binary codes. And we define the quantization residual of a binary code and discuss the transition likelihood distribution created on the basis of quantization residuals as a transition likelihood model. We then discuss classifiers into which the binary code transition likelihood model has been introduced.

5.1 Problems of binary code

The amount of memory for representing a feature can be reduced to 1/64 by binarizing a feature represented by real numbers(double : 8 bytes). This also has other advantages such as it is completely unaffected by factors other than those that affect the characteristics during the binarization of the feature. However, the reduction in the amount of memory for representing the feature inevitably leads to a large reduction in the feature representation capability. This is because information that is valid for classification is comprised in the " quantization residual " (Fig. 7(a)) that is information that drops out during the binarization.

In addition, since the binary code that is used in previous human detection methods are discrete variables, a completely different feature is represented if just one code is different (Fig. 7(b)). For that reason, it can happen that the same binary code cannot be observed, even if features that are similar appear during the representation by real numbers. In essence, in order to perform highly accurate human detection, it is necessary for features to extract elements that are common to the detection objects, so such feature representations are not suitable.

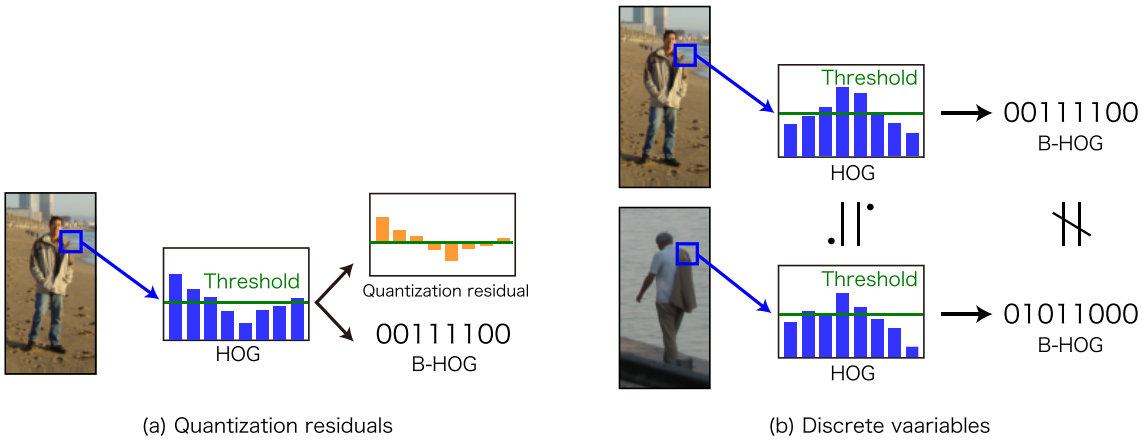

Fig. 7 Problems of binary code.

5.2 Quantization residuals

The proposed method focuses on quantization residuals, which are the amounts of information that drop out when a feature represented by real numbers is binarized. The calculations of the B-HOG feature and R-HOG feature used in this study are described below.

B-HOG feature

The quantization residual of a B-HOG feature is the difference between the gradient strength $v_c(n)$ of a gradient direction histogram in a direction n and a threshold value th, as shown by Equation (5):

$$q_c^B(n) = v_c(n) - th \tag{5}$$

We obtain quantization residuals $\boldsymbol{q}_c^B = \{q_c^B(1), q_c^B(2), \cdots, q_c^B(8)\}$ in all 8 directions from Equation (5).

R-HOG feature

The quantization residual of an R-HOG feature is the difference between two gradient direction histograms $v_{c_1}(n)$, $v_{c_2}(n)$, as shown by Equation (6):

$$q_{c1,c2}^R(n) = v_{c_1}(n) - v_{c_2}(n) \tag{6}$$

We obtain quantization residuals $\boldsymbol{q}_{c1,c2}^R = \{q_{c1,c2}^R(1), q_{c1,c2}^R(2), \cdots, q_{c1,c2}^R(8)\}$ in all 8 directions from Equation (6).

5.3 Transition likelihood distribution of binary code

We create a binary code transition likelihood distribution in order to represent how possible it is that a binary code will transition to another binary code. A transition likelihood distribution accumulates binary code transition scores that are calculated based on the quantization residuals of all training samples. The flow for calculating the transition scores of binary code are shown in Fig. 8. We consider the case in which an observed binary code, such as that shown in Fig. 8(a) transitions to every other binary code. First of all, we calculate the degree of non-inversion of each binary encoding, based on quantization residuals, as shown in Fig. 8(b). The degree of non-inversion of binary encoding represents the likelihood of a transition (bit inversion) occurring in the binary encoding, with it being more unlikely for binary code to transition as the degree of non-inversion increases. We also obtain a binary code transition score (Fig. 8(c)) from the degrees of non-inversion of these binary code. The calculations of transition scores for the binary code are described below.

5.3.1 Binary code transition score

To calculate the binary code transition score, we first obtain the degree of non-inversion z of the binary code. As shown in Fig. 8, when we have assumed that a binary code $\boldsymbol{x}$ that is observed from a certain training sample transitions to another binary code $\boldsymbol{x}'$, we determine the degree of non-inversion from whether or not there is a transition for each binary code $\boldsymbol{x}(n)$ that makes up the binary code $\boldsymbol{x}$ and that quantization residual. In this study, we consider the following two points on the calculation of the degree of non-inversion of binary code:

(1) Whether or not the binary code inverts
(2) Magnitude of the quantization residual

Taking into account the above two points, our considerations are divided into four patterns. First of all, if there is no inversion of the binary code and the quantization residual is large, the degree of non-inversion increases, but if the quantization residual is small, the degree of non-inversion decreases. If there is inversion of the binary code and the quantization residual is large, the degree of non-inversion decreases, but if the quantization residual is small, the degree of non-inversion increases. In the calculation of the degree of non-inversion of this study, we use a concave function $F(q)$ and a convex function $\bar{F}(q)$ in which one-dimensional functions are combined, as shown in Fig. 9. We calculate the degree of non-inversion of the binary code by using a concave function $F(q^i(n))$ and a convex function $\bar{F}(q^i(n))$, as shown in Equation (7), from a binary code $x^i(n)$ observed from a sample i and whether or not there is a transition

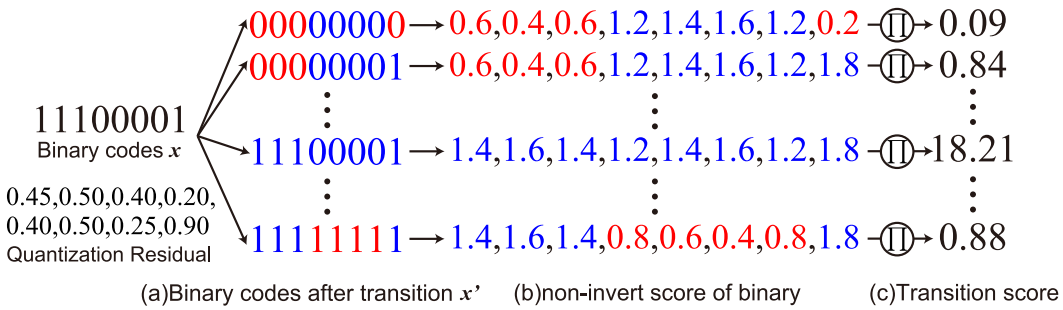

Fig. 8 Flow to compute the transition score of binary code based on quantization residuals. Blue represents that a binary code does not invert. Red represents to invert of a binary code.

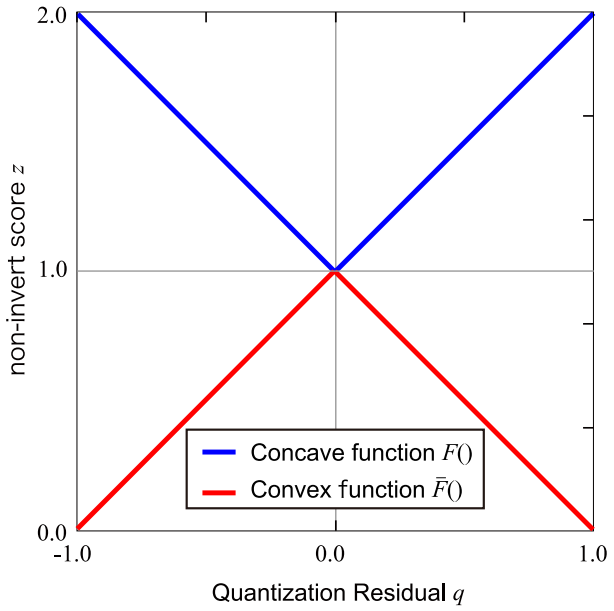

Fig. 9 Non-invert score of a binary code is computed from concave function $F()$ and convex function $\bar{F}()$.

in a binary code $x'(n)$ when we considered transitions to a certain binary code $\boldsymbol{x}'$.

$$z(x^i(n),x'(n),q^i(n)) = \begin{cases} F(q^i(n)) & \text{if } x^i(n)=x'(n) \\ \bar{F}(q^i(n)) & \text{otherwise} \end{cases} \quad (7)$$

The degree of non-inversion of binary code obtained from Equation ((7)) is high when there is no transition in the binary code (codes indicated by the blue characters in Fig. 8(b)) and low where there is a transition (codes indicated by the red characters). In addition to the presence or absence of transitions of the binary code, a transition score corresponding to the value of the quantization residual is output.

5.3.2 Transition scores of binary code

We obtain a transition score $e(\boldsymbol{x}'|\boldsymbol{x}^i)$ for the binary code on the assumption that the binary code $\boldsymbol{x}$ transitions to the binary code $\boldsymbol{x}'$, from the thus-obtained degree of non-inversion z of the binary code. The transition score of the binary code is obtained by taking the sum total of the degrees of non-inversion of the binary encoding, as shown in Equation (8), to take into account synchronism of the binary encoding.

$$e(\boldsymbol{x}'|\boldsymbol{x}^i,\boldsymbol{q}^i) = \prod_{n=1}^{8} z(x^i(n),x'(n),q^i(n)) \quad (8)$$

This is obtained from all of the training samples I, to create a transition likelihood distribution E for the binary code by summing them as shown by Equation (9):

$$E(\boldsymbol{x}'|\boldsymbol{x}) = \sum_{i=1}^{I} e(\boldsymbol{x}'|\boldsymbol{x}^i,\boldsymbol{q}^i)\delta[\boldsymbol{x}^i-\boldsymbol{x}'] \quad (9)$$

$\delta[\cdot]$ is the Kronecker delta function, which outputs 1 when $\boldsymbol{x}^i-\boldsymbol{x}'=0$ and 0 otherwise. We create a transition likelihood distribution E as described above for each feature. Thus 72 transition likelihood distributions are created for a B-HOG feature or 8,128 transition likelihood distributions for an R-HOG feature.

An example of transition likelihood distributions that we have created is shown in Fig. 10(a). This transition likelihood distribution has higher transition likelihoods as codes become more similar to the input binary code. For example, if the input binary code is {00000000}, the binary code after transition where the Hamming distance is 0 is naturally the most likely. After that, the next likely binary code are {00000001} and {10000000}. When the Hamming distance increases, on the other hand, transitions in the binary code are unlikely to occur, so it is clear that the transition likelihood is less. However, transition likelihoods assume values that are different from the Hamming distances because they are determined on the basis of the quantization residuals of the training samples. For that reason, when transitions of the binary code are likely to occur, in other words, when there are many samples with small quantization residuals within the training samples, the proposed method is better at representing transitions between binary code that is likely to occur in practice than with Hamming distance.

5.4 Transition prediction based on quantization residual

In this section, we discuss classifiers into which are introduced a transition likelihood model created on the basis of quantization residuals obtained from training samples (Fig. 6(b)).

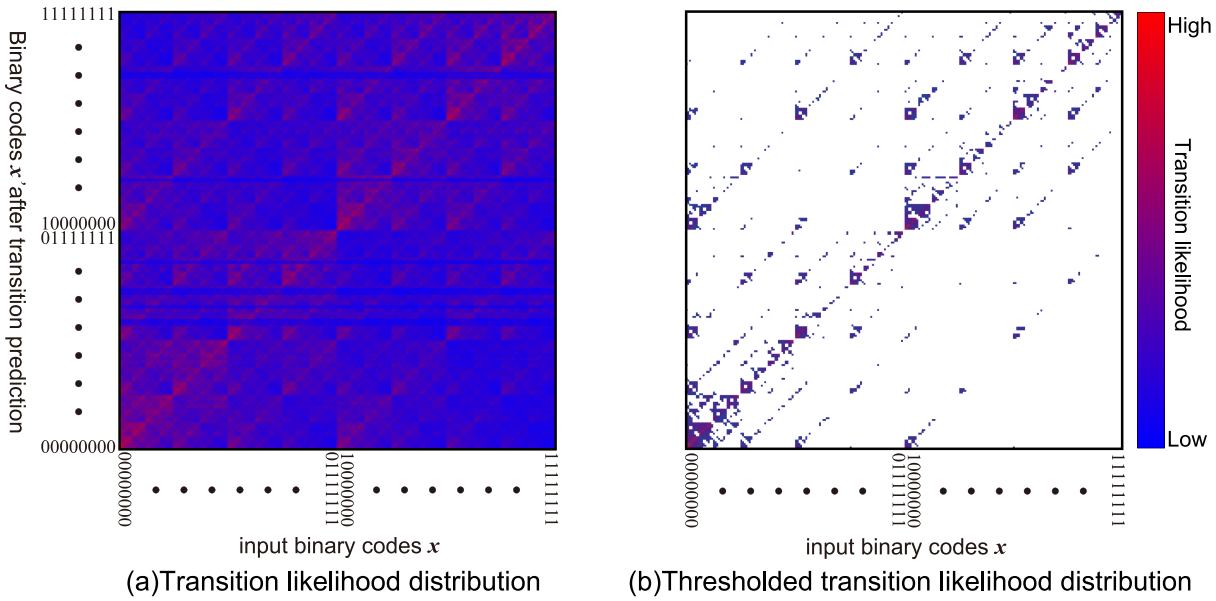

Fig. 10 Visualization of transition likelihood distribution $E(\boldsymbol{x}'|\boldsymbol{x})$ of binary code.

5.4.1 Classifiers using transition likelihood distributions

In our research, we use Real AdaBoost [14], which enables highly accurate and fast classifications, as the statistical training method. A strong classifier $H(\boldsymbol{x})$ that is trained by the Real AdaBoost algorithm is represented by linear linking of weak classifiers $h_t(\boldsymbol{x})$, as shown in Equation (10):

$$H(\boldsymbol{x}) = \sum_{t=1}^{T} h_t(\boldsymbol{x}) \tag{10}$$

where T is the total number of weak classifiers that are combined and t is the number of each weak classifier. Subsequently, since the number of the weak classifier is irrelevant, we express $h(\boldsymbol{x})$ as a general weak classifier. The weak classifiers $h(\boldsymbol{x})$ are determined by the log ratio of the probability of occurrence of positive class or negative class, as shown in Equation (11).

$$h(\boldsymbol{x}) = \frac{1}{2}\ln\frac{W_+(\boldsymbol{x})}{W_-(\boldsymbol{x})} \tag{11}$$

where the probability density function $W_\pm$ of the binary code is created by summing the weighting w_i of the training samples as shown by Equation (12) and Equation (13):

$$W_+(\boldsymbol{x}) = \sum_{\boldsymbol{x}^i=\boldsymbol{x}\wedge y_i=+1} w_i \tag{12}$$

$$W_-(\boldsymbol{x}) = \sum_{\boldsymbol{x}^i=\boldsymbol{x}\wedge y_i=-1} w_i \tag{13}$$

Since this study considers the transition from the observed binary code $\boldsymbol{x}$ to another binary code $\boldsymbol{x}'$, we introduce the transition likelihood model into Equation (14) and define each weak classifier $h(\boldsymbol{x})$ as shown by Equation (14).

$$h(\boldsymbol{x}) \triangleq \frac{1}{2}\sum_{\boldsymbol{x}'\in\mathcal{X}}\left(P(\boldsymbol{x}'|\boldsymbol{x})\ln\frac{W_+(\boldsymbol{x}')}{W_-(\boldsymbol{x}')}\right) \tag{14}$$

where $P(\boldsymbol{x}'|\boldsymbol{x})$ represents the probability of the binary code $\boldsymbol{x}$ transitioning to the binary code $\boldsymbol{x}$'. However, in practice the observed binary code does not transition to another binary code so $P(\boldsymbol{x}'|\boldsymbol{x})$ cannot be obtained. That is why in this study, we substitute a transition likelihood distribution $E(\boldsymbol{x}'|\boldsymbol{x})$ that represents the possibilities of transitions between binary code, as shown by Equation (15). The transition likelihood distribution $E(\boldsymbol{x}'|\boldsymbol{x})$ can simulate the representation of the transition probability $P(\boldsymbol{x}'|\boldsymbol{x})$ of binary code that cannot be observed in practice.

$$P(\boldsymbol{x}'|\boldsymbol{x}) \approx \frac{E(\boldsymbol{x}'|\boldsymbol{x})}{\sum_{\boldsymbol{x}'\in\mathcal{X}} E(\boldsymbol{x}'|\boldsymbol{x})} \tag{15}$$

If the binary code $\boldsymbol{x}$ is not observed during this time, we consider that there is also no transition between binary code. In such a case, both the denominator and numerator of Equation (15) are zero, so $P(\boldsymbol{x}'|\boldsymbol{x}) = 0$. As described above, classification can be done by taking into account the possibility of an observed binary code transitioning into another binary code, from Equations (14) and (15). However, since Equation (14) is considered for transitions to all of the binary codes, the log ratio of the probability of occurrence of a binary code that differs greatly from the observed binary codeg also has an effect on the weak classifiers. Since the log ratio is obtained during this process even when the probability of occurrence of the probability density function $W_\pm$ is equivalent to substantially zero, an extremely large or small value is added, which has an adverse effect on the value of the weak classifier $h(\boldsymbol{x})$. To resolve that problem, we subject the transition likelihoods to threshold value processing as shown in Equation (16), to suppress the effects on the classification results of binary- codes with low possibilities of transition from the observed binary code.

$$E'(\boldsymbol{x}'|\boldsymbol{x}) = \begin{cases} E(\boldsymbol{x}'|\boldsymbol{x}) & \text{if } E(\boldsymbol{x}'|\boldsymbol{x}) > \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

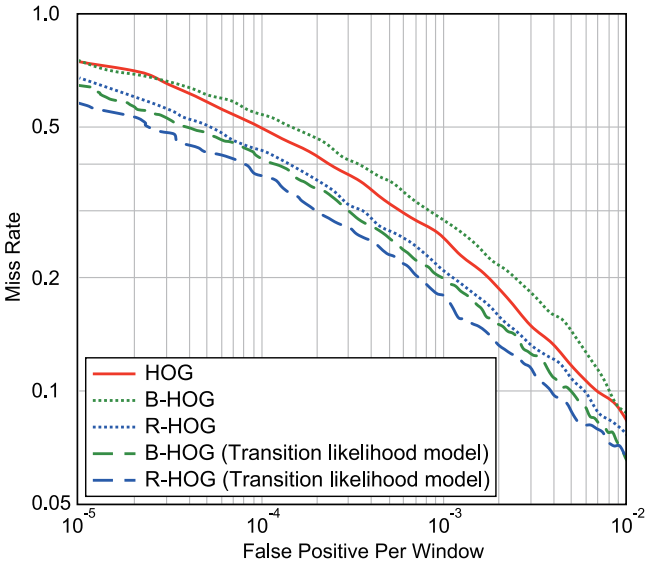

Fig. 11 DET curves of experiment.

This threshold value ϵ creates a model that does not take account of transitions of the binary code if it is set to an extremely large value. In this study, we set $\epsilon = 0.005$ from preliminary experiments. The transition likelihood distribution before the threshold value processing is shown in Fig.10(a) and the transition likelihood distribution after the threshold value processing is shown in Fig.10(b). Weak classifiers using the binary code transition likelihood distribution that we propose (Equation (14)) seem at first glance to have larger computational costs when compared with the Real AdaBoost weak classifiers (Equation (11)) that are generally used. However, the same computational costs are achieved by previously computing outputs for each binary code that is observed in practice, and saving them as a look-up table.

5.4.2 Classification

We discuss the flow when classifying an unknown input image using trained classifiers. We first calculate binary code from the unknown input image, as shown in Fig.6(b). We also obtain the degree of likelihood of an observed binary code transitioning into another binary encoding from the transition likelihood distribution, and take the summation of the product with the log odds to be a weak classifier. We calculate the final classification results by summing a number of weak classifiers $h(\boldsymbol{x})$ as shown in Equation (10).

6. Evaluation Experiments

We performed experiments to evaluate the validity of the proposed method.

6.1 Experiment results

A DET curve of the results of the experiments is shown in Fig.11. First of all, if the HOG feature , B-HOG feature , and the proposed method based on the B-HOG feature are compared, the proposed method, the HOG feature , and the B-HOG feature are shown to be in decreasing order of detection performance. The detection rate of the B-HOG feature , which is a HOG feature that has been subjected to threshold value processing where the FPPW is 10^{-3}, deteriorated by approximately 2.9% in comparison with the HOG feature. When the proposed method based on the B-HOG feature where the FPPW is 10^{-3} was compared with the HOG feature and B-HOG feature, the detection rate rose by approximately 5.4% and 8.4% respectively. From the above, we see that the detection performance deteriorated when the HOG feature was simply binarized by threshold value processing, but we were able to obtain detection performances that exceeded those of the HOG feature by introducing binary code predictions into the classifiers, in accordance with the proposed method.

In addition, this tendency produced similar results even when the binarization method was different. A comparison of the R-HOG feature where FPPW was 10^{-3} and the proposed method based on the R-HOG feature demonstrated an increase in detection performance of approximately 3.0%.

From the above results, we have confirmed that the proposed method enables human detection that is more accurate than that of previous methods. With previous methods that use binary code, binary code that is discrete variables are handled as mutually independent values. For that reason, if the observed binary code has been observed to be another binary code for some reason, only the observed binary code will be considered during the classification, so it is possible that that classification results will vary widely. The proposed method, on the other hand, forms a framework which enables predictions of how likely the observed binary code will transition into all the other binary codes. Since the binary code transitions are based on the transition likelihood distribution created from the binary code obtained from training samples and the quantization residuals, the possibilites of binary code transitions that occur readily in practice are considered. For that reason, even if a binary code has been observed to be another binary code for some reason or other, that adverse effect can be restrained.

6.2 Discussion

We have confirmed from the results of the evaluation experiments that weak classifiers into which a binary code transition likelihood model has been introduced contribute to an increase in detection performance, irrespective of the method used to binarize the feature. In this section, we discuss to what degree is the detection performance of each weak classifier raised by the introduction of binary code transition predictions.

The results of miss-classification rates of general Real AdaBoost weak classifiers (Equation (11)) and the miss-classification rates of weak classifiers into which transition predictions have been introduced (Equation (14)) are plotted on two-dimensional graphs in Fig. 12. All of the plotted points represent miss-classification rates when the same feature is used. If there is no change in the performance of a weak classifier, it is plotted on the red line indicating $y = x$. If the detection performance is higher with the proposed

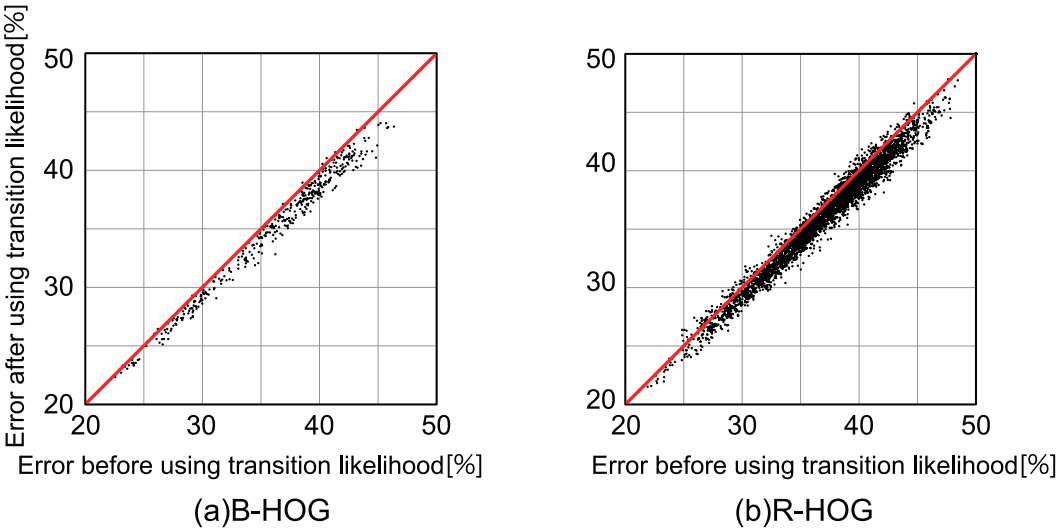

Fig. 12 The graph of error computed from equation (11) and equation (14). The ploted points are represented equal error rate. (a) B-HOG feature base. (b) R-HOG feature base.

method, the point is plotted to the right and below the red line.

From Fig. 12, it is clear that the detection performance of a large number of weak classifiers is increased by the proposed method. When based on the B-HOG feature, the detection performance for approximately 96.2% of the weak classifiers can be raised, and when based on the R-HOG feature, the detection performance for approximately 88.3% of the weak classifiers can be raised. The classification rate for each individual weak classifier increases by approximately 5.0% at a maximum. However, since Real AdaBoost, which is an ensemble training method in which large numbers of weak classifiers are combined, is used as the training method, the final classifiers trained by the proposed method can achieve an even higher detection performance.

7. Conclusions

In this paper, we proposed binary codes from HOG feature and classifier introducing transition likelihood model based on quantization residual. Binary codes were reduced the memory used for representing features by subjecting HOG features to threshold value processing then binarizing in accordance with the magnitude relationships of two histograms. However, the process of quantizing a features into binary form creates a problem in that a great deal of the information within the features drops out. Then, we proposed a method of effective utilization of quantization residuals, which have not been used previously in the pattern recognition field. The proposed method introduces a transition likelihood model into classifiers, in order to consider the possibility of a binary code that has been observed from an image transitioning into another binary code. We used a transition likelihood distribution created from binary code and quantization residuals obtained from training samples, as the transition likelihood model. The proposed method is capable of outputting highly reliable classification results since it can consider the possibility of an observed binary code transitioning into another binary code. The results of experiments show that the proposed method enables an increase in detection performance while maintaining the same levels of memory and computing costs as those for previous methods of binarizing features.

The approach that makes effective use of quantization residuals in accordance with the proposed method is implemented by Boosting-based classifiers in this study, but we consider it will be possible to expand it into other classifiers such as Random Forest and SVM. We will examine expansion into such training methods in the future.

References

[1] Bosch, A., Zisserman, A. and Munoz, X.: Representing shape with a spatial pyramid kernel, *international conference on Image and video retrieval* (2007).
[2] Dalal, N. and Triggs, B.: Histograms of Oriented Gradients for Human Detection, *Computer Vision and Pattern Recognition*, Vol. 1, pp. 886–893 (2005).
[3] Ess, A., Leibe, B., Schindler, K. and Gool, L. V.: A Mobile Vision System for Robust Multi-Person Tracking, *IEEE Conf. on CVPR*, pp. 1–8 (2008).
[4] Hadid, A., Pietikainen, M. and Ahonen, T.: A Discriminative Feature Space for Detecting and Recognizing Faces, *IEEE Conf. on CVPR*, Vol. 2, pp. 797–804 (2004).
[5] Hou, C., Ai, H. Z. and Lao, S. H.: Multiview Pedestrian Detection Based on Vector Boosting, *Asian Conference on Computer Vision*, pp. 210–219 (2007).
[6] Khattab, K., Dubois, J. and Miteran, J.: Cascade Boosting-Based Object Detection from High-Level Description to Hardware Implementation, *EURASIP Journal on Embedded Systems*, Vol. 2009, pp. 1–12 (2009).
[7] Leibe, B., Seemann, E. and Schiele, B.: Pedestrian detection in crowded scenes, *Computer Vision and Pattern Recognition*, Vol. 1, pp. 878—-885 vol. 1 (2005).
[8] Liao, S., Zhu, X., Lei, Z., Zhang, L. and Li, S.: Learning Multi-scale Block Local Binary Patterns for Face Recognition, *in Advances in Biometrics*, pp. 828–837 (2007).
[9] Linde, Y., Buzo, A. and Gray, R.: An Algorithm for Vector Quantizer Design, *IEEE Trans. on Communications*, Vol. 28, pp. 84–95 (1980).
[10] Mita, T., Kaneko, T. and Hori, O.: Joint Haar-like Features Based on Feature Co-occurrence for Face Detection (in Japanese), *IEICE Trans.*, Vol. J89-D, No. 8, pp. 1791–1801 (2006).

[11] Mu, Y. D., Yan, S. C., Liu, Y., Huang, T. and Zhou, B. F.: Discriminative local binary patterns for human detection in personal album, *Computer Vision and Pattern Recognition*, pp. 1–8 (2008).
[12] Nair, V., Laprise, P. O. and Clark, J. J.: An FPGA-Based People Detection System, *EURASIP Journal on Applied Signal Processing*, Vol. 2005, pp. 1047–1061 (2005).
[13] Ojala, T., ainen, M. P. and Harwood, D.: A comparative study of texture measures with classification based on featured distributions, *Pattern Recognition*, Vol. 29, pp. 51–59 (1996).
[14] Schapire, R. E. and Singer, Y.: Improved boosting algorithms using confidence-rated predictions, *Machine Learning*, Vol. 37, No. 3, pp. 297–336 (1999).
[15] Tan, X. and Triggs, B.: Enhanced Local Texture Feature Sets for Face Recognition under Difficult Lighting Conditions, *IEEE Transactions on Image Processing*, Vol. 19, pp. 1635–1650 (2010).
[16] Tuzel, O., Porikli, F. M. and Meer, P.: Human Detection via Classification on Riemannian Manifolds, *Computer Vision and Pattern Recognition*, pp. 1–8 (2007).
[17] Wang, X., Han, T. X. and Yan, S.: An HOG-LBP Human Detector with Partial Occlusion Handling, *International Conference on Computer Vision* (2009).
[18] Watanabe, T., Ito, S. and Yokoi, K.: Co-occurrence Histograms of Oriented Gradients for Human Detection, *Information Processing Society of Japan Transactions on Computer Vision and Applications*, Vol. 2, pp. 39–47 (2010).
[19] Wu, B. and Nevatia, R.: Detection of Multiple, Partially Occluded Humans in a Single Image by Bayesian Combination of Edgelet Part Detectors, *International Conference on Computer Vision*, pp. 90–97 (2005).
[20] Yamauchi, Y., Takaki, M., Yamashita, T. and Fujiyoshi, H.: Feature Co-occurrence Representation Based on Boosting for Object Detection, *International Workshop on Socially Intelligent Surveillance and Monitoring(in conjunction with CVPR)*, pp. 31–38 (2010).
[21] Yeffet, L. and Wolf, L.: Local Trinary Patterns for human action recognition, *Proc. of IEEE International Conference on Computer Vision* (2009).