IPSJ Transactions on Computer Vision and Applications Vol.7 55–58 (July 2015)

[DOI: 10.2197/ipsjtcva.7.55]

Express Paper

Fast and Accurate Object Detection Based on Binary Co-occurrence Features

Mitsuru Ambai[1,a)] Taketo Kimura[1,b)] Chiori Sakai[1,c)]

Received: March 13, 2015, Accepted: April 20, 2015, Released: July 27, 2015

Abstract: In this paper, we propose a fast and accurate object detection algorithm based on binary co-occurrence features. In our method, co-occurrences of all the possible pairs of binary elements in a block of binarized HOG are enumerated by logical operations, i.e. circular shift and XOR. This resulted in extremely fast co-occurrence extraction. Our experiments revealed that our method can process a VGA-size image at 64.6 fps, that is two times faster than the camera frame rate (30 fps), on only a single core of CPU (Intel Core i7-3820 3.60 GHz), while at the same time achieving a higher classification accuracy than original (real-valued) HOG in the case of a pedestrian detection task.

Keywords: binary features, object detection, co-occurrence

1. Introduction

Nowadays a linear classifier is widely used in an object detection task. One of the representative works is HOG with SVM proposed by Dalal et al. [3], which is later extended to deal with part-based deformation by Felzenszwalb et al. [6]. In spite of its simplicity, the linear object detectors can achieve comparable accuracy to non-linear object detectors in certain kinds of tasks, e.g., pedestrian detection [5].

From a practical perspective, real-time processing is also very important. Recently it has been reported that using binary features instead of real-valued features (such as original HOG) can drastically accelerate the computation time of the linear classifier [1], [7], [8]. A key feature of this line of approach is that the weight vector of the linear classifier is decomposed into a weighted sum of a few integer basis vectors that only contain limited integer values, e.g., $\{-1, 0, +1\}$ in the case of Ref. [1]. By imposing such constraints on the basis vectors, the computation of the linear classifier is reduced to logical operations such as AND, XOR, and bit counts. The detail of this weight decomposition approach will be revisited in Section 2.

One drawback of binary feature representation is its insufficient discriminative power. Although it has been reported in Ref. [1] that binarized HOG with carefully designed thresholds can achieve almost the same classification accuracy as original (real-valued) HOG, such level of accuracy is outdated. In addition, there remains another problem, i.e. the feature extraction speed is still a bottleneck for the total performance, because a dense feature pyramid must be created to cope with different object sizes.

[1] Denso IT Laboratory, Inc., Shibuya, Tokyo 150–0002, Japan
[a)] manbai@d-itlab.co.jp
[b)] tkimura@d-itlab.co.jp
[c)] csakai@d-itlab.co.jp

1.1 Technical Contributions of This Work

In this paper, we propose a *fast* and *accurate* object detection algorithm based on binary features and a weight decomposition approach. Our method can process a VGA-size image at 64.6 fps, that is two times faster than the camera frame rate (30 fps), on only a single core of CPU (Intel Core i7-3820 3.60 GHz), while at the same time achieving a higher classification accuracy than HOG in the case of a pedestrian detection task. The contributions of our work are listed below.

- *Binary co-occurrence features:* We improve the discriminative power of binarized HOG by enumerating all pairwise co-occurrences in a block. We show that they can be enumerated by only using logical operations: circular shift and XOR, that results in extremely fast co-occurrence extraction.
- *Detection cascade:* Due to the high dimensionality of the binary co-occurrence features, the classification speed becomes slow even when the weight decomposition is applied. We introduce a two-stage cascade: while the first stage uses the original binary features (without co-occurrence), the second stage uses the binary co-occurrence features. In this framework, only a limited number of samples are passed to the 2nd stage. This results in a significant speed up.
- *Hybrid pyramid:* Instead of applying a single object model to a dense feature pyramid, we apply multiple object models with different sizes to a coarse feature pyramid (that consists of only a single layer per octave). This reduces the computation time of feature extraction, while classification is sufficiently accelerated by the weight decomposition approach.

The rest of this paper is organized as follows. Section 2 briefly reviews the weight decomposition approach. Section 3, 4, and 5 explain the above contributions: binary co-occurrence features, detection cascade, and hybrid pyramid, respectively. Section 6 shows experiments taking a pedestrian detection task as an example. We give concluding remarks in Section 7.

2. Fast Linear Classifier based on Weight Decomposition

In this section, we briefly review the weight decomposition method proposed in Refs. [1], [7], [8]. The linear classifier is defined as follows.

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b, \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^D$ is a weight vector, $\mathbf{x} \in \{-1, +1\}^D$ is a binary feature vector extracted from an image, D is a dimension, and $b \in \mathbb{R}$ is a bias term. To avoid a large number of floating point operations, the following approximated classifier is introduced.

$$f(\mathbf{x}) \approx f_{\text{approx}}(\mathbf{x}) = \sum_{i=1}^{M} c_i \mathbf{m}_i^\top \mathbf{x} + b, \tag{2}$$

where $\mathbf{m}_i$ is an i-th integer basis vector, $c_i \in \mathbb{R}$ is its coefficient, and M is the number of the basis vectors used for this approximation. In this way, $\mathbf{w}$ is decomposed into c_i and $\mathbf{m}_i$, which are determined by optimization so as to preserve the original score $f(\mathbf{x})$.

Since both $\mathbf{m}_i$ and $\mathbf{x}$ are discrete vectors, several techniques are available to quickly compute $\mathbf{m}_i^\top \mathbf{x}$. The state-of-the-art [1] recommended to impose ternary constraint on $\mathbf{m}_i \in \{-1, 0, +1\}^D$, from the viewpoint of a trade-off between approximation accuracy and computation time. If $\mathbf{m}_i$ is ternary, $\mathbf{m}_i^\top \mathbf{x}$ can be computed extremely fast by the following three logical operations: AND, XOR, and bit counts [*1]. Besides, the sign of the approximated classification score $f_{\text{approx}}(\mathbf{x})$ can be estimated by using only the first several inner products $\mathbf{m}_i^\top \mathbf{x}$ in most cases, where we should skip the rest of the inner product computation to reduce computation time. Throughout our work, we used the ternary decomposition and the early termination by the sign estimation proposed in Ref. [1].

3. Binary Co-occurrence Features

It is well known that co-occurrence features, e.g., Feature INteraction Descriptor (FIND) [2], XOR features [7], and Co-occurrence Histograms of Oriented Gradients (CoHOG) [9], significantly improve detection accuracy. Our work has a close relation to Refs. [2] and [7] as shown later. In the rest of this section, we begin with reviewing HOG and FIND. Next we revisit FIND in the case where the features are binary. Finally we give a fast co-occurrence extraction algorithm based on logical operation.

Here we briefly review HOG. An image is divided into lattice-shaped small regions called cells, each of which contains a histogram of oriented gradients. The edge orientation is quantized to 8 directions. 2×2 cells are grouped into a block. The histogram is normalized for each block. A HOG feature in a block is defined as $\mathbf{x} = (x_1, \cdots, x_{32})^\top \in \mathbb{R}^{32}$ (because $2 \times 2 \times 8 = 32$).

FIND is obtained by enumerating all pairwise products among elements in $\mathbf{x}$.

$$x_{i,j} = \frac{x_i x_j}{\mathcal{M}}, \tag{3}$$

[*1] If $\mathbf{m}_i$ is binary, it is well known that $\mathbf{m}_i^\top \mathbf{x}$ can be obtained by XOR followed by bit count operation. Even if $\mathbf{m}_i$ is ternary, $\mathbf{m}_i^\top \mathbf{x}$ can be obtained in a similar manner. The ternary case is well investigated in Ref. [1].

Table 1 Connection between pairwise product and XOR.

x_i	x_j	$x_i x_j$	$\text{XOR}(x_i, x_j)$	$-\text{XOR}(x_i, x_j)$
+1	+1	**+1**	−1	**+1**
−1	+1	**−1**	+1	**−1**
+1	−1	**−1**	+1	**−1**
−1	−1	**+1**	−1	**+1**

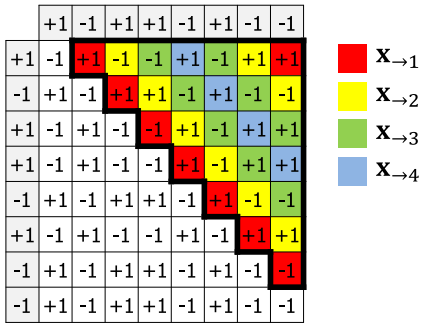


Fig. 1 This figure illustrates XOR values of all pairs by taking 8-bit string as an example. Since this figure is symmetric with respect to a diagonal line, we need elements surrounded by the black bold line only. This figure is best viewed in color.

where $x_{i,j}$ is a co-occurrence feature between x_i and x_j, and $\mathcal{M}$ is a scaling factor that is set to a mean value of x_i and x_j. The product $x_i x_j$ returns a strong response, if both x_i and x_j are large. In this way, co-occurrences are captured by the pairwise products.

Here we investigate the case where HOG is binarized by element-wise quantization as shown in Ref. [1]. In this case, the pairwise products of the binarized HOG $\mathbf{x} \in \{-1, +1\}^{32}$ can be substituted for XOR as follows.

$$x_i x_j = -\text{XOR}(x_i, x_j), \tag{4}$$

where $\text{XOR}(x_i, x_j)$ regards -1 and $+1$ as false and true bits, respectively. **Table 1** would help readers to understand Eq. (4). This relationship suggests that we can extract FIND-like features by XOR instead of floating point computations originally needed in Eq. (3). Goto et al. [7] also proposed to use XOR of binary features to capture their co-occurrences. However their method picked up limited pairs of binary elements in a block, i.e., co-occurrences of different edge orientations were not taken into account. In contrast, our method enumerates all binary co-occurrences in a similar manner to FIND.

XOR values of all possible pairs in a D-bit string are highlighted by a bold black line in **Fig. 1**. Although $D = 32$ in practice, this figure shows the case where $D = 8$ for ease of explanation. Interestingly, elements surrounded by the bold black line can be enumerated by only using logical operations. Circular shift followed by XOR can compute D binary co-occurrences at the same time as follows.

$$\mathbf{x}_{\to t} = \text{XOR}(\text{ROTATE}(\mathbf{x}, t), \mathbf{x}), \tag{5}$$

where $\text{ROTATE}(\mathbf{x}, t)$ is a function of circular shift, and t is the number of places to shift. The case where $t = 2$ is visualised in **Fig. 2**. It is obvious that $\mathbf{x}_{\to 1}$, $\mathbf{x}_{\to 2}$, $\mathbf{x}_{\to 3}$, and $\mathbf{x}_{\to 4}$ correspond to red, yellow, green, and blue elements in Fig. 1, respectively. Although half of $\mathbf{x}_{\to 4}$ is redundant (upper 4 bits are the same as lower 4 bits), we do not exclude the redundant bits for efficiency reason. Our binary co-occurrence features are defined as follows.

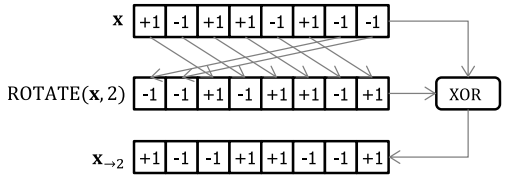


Fig. 2 Extracting co-occurrence features by using circular shift and XOR. The case where $t = 2$ is shown here.

$$\hat{\mathbf{x}} = (\mathbf{x}_{\rightarrow 1}, \cdots \mathbf{x}_{\rightarrow D/2}, \mathbf{x})^{\top}. \tag{6}$$

As shown above, $\hat{\mathbf{x}}$ also includes original features $\mathbf{x}$. This computation only needs $D/2$ circular shift operations and $D/2$ XOR operations, while Eq. (3) needs $O(D^2)$ floating point operations. In our experiment, our co-occurrence extraction was 327.3 times faster than FIND.

4. Detection Cascade

Although the binary co-occurrence representation enriches discriminative power, its high dimensionality may put an extra computational load on classification. To overcome this issue, we introduce a two-stage cascade: while the 1st stage uses the original binary features $\mathbf{x}$ (without co-occurrence), the 2nd stage uses the binary co-occurrence features $\hat{\mathbf{x}}$. The computational load of the 2nd classifier is sufficiently reduced, because most of the search windows placed on background patterns are rejected at the 1st stage.

For each level of an image pyramid, the following detection algorithm is applied.

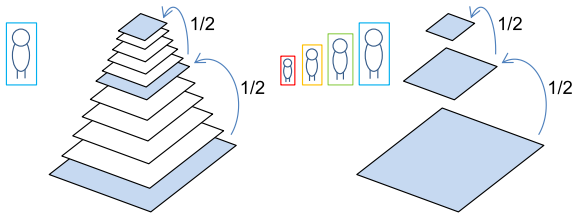
(1) Binary HOG $f(u, v) \in \{-1, +1\}^{32}$ is computed from a given image, where u and v are horizontal and vertical location in block-coordinate.
(2) Binary co-occurrence features $\hat{f}(u, v) \in \{-1, +1\}^{544}$ are computed from $f(u, v)$ by Eq. (6).
(3) The 1st stage classifier is applied to $f(u, v)$ in sliding window manner.
(4) For each location that gets through the 1st classifier, the 2nd stage classifier is applied to $\hat{f}(u, v)$.

The 1st and 2nd classifier are independently trained on the same training dataset. To control a number of passing samples from the 1st to the 2nd stage, a pre-defined positive value α is added to the bias of the 1st classifier. This parameter α is determined by a preliminary experiment as shown later in Section 6.1.

5. Hybrid Pyramid

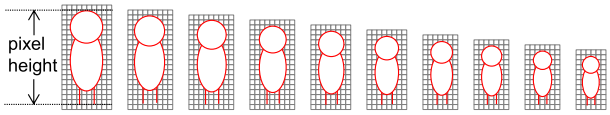
In conventional approaches, a single classifier is applied to a dense feature pyramid that contains many layers for each octave as illustrated in **Fig. 3** (a). However, if the computation time of classification is obviously faster than that of feature extraction, applying multiple classifiers with different sizes to a coarse feature pyramid is better for fast computation as suggested in Ref. [4]. We call this approach a hybrid pyramid which is illustrated in Fig. 3 (b). The hybrid pyramid is suited for our purpose, because our classifier is sufficiently accelerated by weight decomposition (reviewed in Section 2) and two-stage cascade (introduced in Section 4).

In the hybrid pyramid approach, the feature pyramid has only a single layer for each octave. Instead, 10 different classifiers are



(a) Conventional approach (b) Hybrid pyramid

Fig. 3 Two types of scale search. (a) A conventional approach trains only a single classifier and builds a dense feature pyramid. (b) A hybrid pyramid creates both a coarse feature pyramid and a classifier pyramid.



cells	10 × 21	9 × 20	9 × 19	9 × 18	8 × 17	8 × 16	7 × 15	7 × 14	7 × 13	6 × 12
pixel height	149.0	139.7	130.6	121.6	112.8	105.3	98.8	92.2	85.4	80.0

Fig. 4 Classifier pyramid. 10 classifiers are trained to cope with varied target sizes. From left to right, pixel heights are resized at a constant rate, $(1/2)^{(1/10)}$. Training patterns are placed so as to leave a small space on the top and bottom.

trained as shown in **Fig. 4**. All the classifiers consist of a different number of cell sizes in vertical and horizontal directions. Training patterns are placed at the center of them, and resized in such a way that the ratio of pixel heights from $(n-1)$-th to n-th level becomes $(1/2)^{(1/10)}$. This means that the largest object model is about two times taller than the smallest one. The cell sizes and the pixel heights of training patterns are summarized in Fig. 4.

6. Experiments and Discussion

Our method is evaluated on INRIA person dataset. As a classification performance indicator, Log-Average Miss Rate (LAMR) which is the average of miss rates at 9 false positive per image (FPPI) evenly sampled in log-space, is used as well as Ref. [5]. The following four detectors are compared in our experiments.

- *OrgHOG:* Original HOG proposed in Ref. [3]. 40 level feature pyramid (10 levels/octave) are created. The detector is formed as 6×12 cells which is the same as the smallest model in Fig. 4.
- *BinHOG:* 32-dimensional real-valued HOG is converted to 32-bit binary HOG. The hybrid pyramid approach is used to reduce the computational load of feature extraction.
- *co-occur:* The binary co-occurrence features are used in conjunction with the hybrid pyramid. The detection cascade introduced in Section 4 is not applied.
- *BinHOG+co-occur:* All of the methods explained in Section 2, 3, 4, and 5 are used to form an object detector.

6.1 Parameter Setting for Detection Cascade

The parameter α, that controls a number of passing samples from the 1st to the 2nd stage, must be chosen so as to balance both detection speed and accuracy. **Fig. 5** illustrates LAMR and detection speed with respect to α. When a large value was set to α, LAMR was improved while detection speed became slow. This is because more samples were passed from the 1st to the

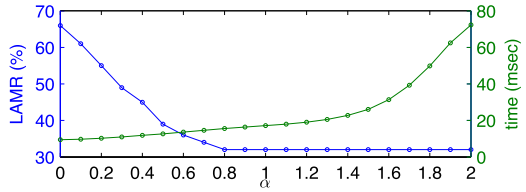


Fig. 5 Effect of parameter α. The blue plots are LAMR and the green plots are detection time. Intel Corei7-3820 processor (3.60 GHz) and a VGA image is used for measuring detection time. This figure is best viewed in color.

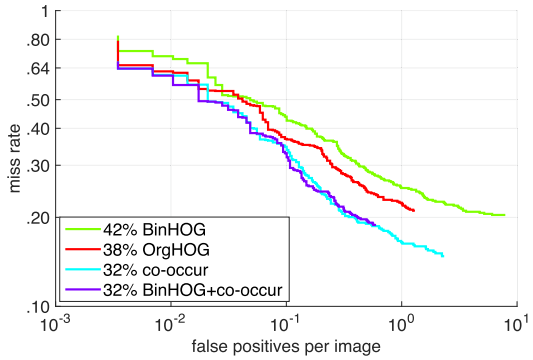


Fig. 6 Comparison of detection accuracies.

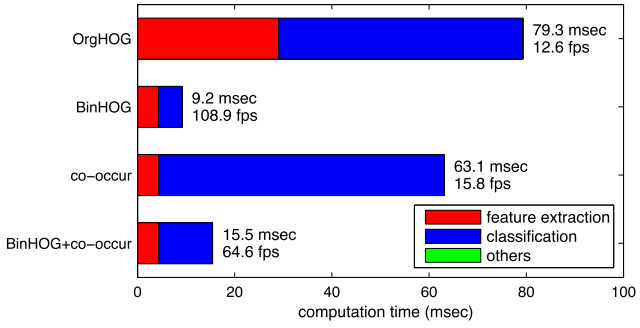


Fig. 7 Detection time on Intel processor.

2nd stage. In contrast, setting a small value to α resulted in fast detection and bad LAMR. As can be seen from Fig. 5, setting 0.8 to α can sufficiently accelerate detection speed without loss of accuracy. In the rest of experiments, we use this parameter.

6.2 Detection Accuracy

Fig. 6 shows the performance curves of miss rates versus FPPI. LAMR is also indicated for each method. It should be noted that *co-occur* outperformed not only *BinHOG* but also *OrgHOG* which is represented in a real-valued vector. Thus it was revealed that extracting binary co-occurrences can significantly improve detection accuracy. In addition, the LAMR of *BinHOG+co-occur* was the same as that of *co-occur*. The careful choice of the parameter α resulted in a balanced performance both in terms of accuracy and detection speed.

6.3 Detection Speed

The detection times are compared in **Fig. 7**. All the four methods were tested on Intel Core i7-3820 processor (3.60 GHz). Only a single core on the CPU was used. A *popcnt* instruction in SSE4.2 was used for counting true bits needed in classification. 500 VGA images were captured from a web camera to compute average detection time.

Table 2 Detection time on ARM processor.

CPU (product name)	Detection time
Cortex-A15 2.0 GHz (ODROID XU3)	42.5 msec (23.5 fps)
Cortex-A9 1.7 GHz (ODROID U3)	81.0 msec (12.3 fps)

We compared *OrgHOG* with *BinHOG* to observe that the hybrid pyramid combined with the weight decomposition [1] significantly accelerated both feature extraction and classification. *BinHOG* achieved 108.9 fps in exchange for 4% loss of LAMR compared to *OrgHOG*. It should be noted that the feature extraction speed of *co-occur* was almost the same as *BinHOG*. This result indicates that the computational load of binary co-occurrence extraction based on circular shift and XOR operations was extremely fast. The classification time of *BinHOG+co-occur* was 5.3 times faster than that of *co-occur* even without loss of accuracy. This was brought about by the detection cascade tuned by carefully chosen parameter α. *BinHOG+co-occur* run at 64.6 fps.

We also tested our *BinHOG+co-occur* on ARM-Cortex A15 and A9 processors with which mobile phones are commonly equipped. We used only a single core on the CPUs. The bit count operation was optimized by NEON instruction set. The test image size was VGA. The results shown in **Table 2** revealed that our method can run sufficiently fast even on low-end CPUs.

7. Conclusion

In this paper, we revealed that binary co-occurrence features can improve classification accuracy while maintaining real-time processing speed. Although we did not discuss memory consumption due to page limitation, our method requires little memory because features are represented in binary strings. This practical advantage made it easy to implement our method on low-end hardware.

References

[1] Ambai, M. and Sato, I.: SPADE: Scalar Product Accelerator by Integer Decomposition for Object Detection, *ECCV* (2014).
[2] Cao, H., Yamaguchi, K., Ohta, M., Naito, T. and Ninomiya, Y.: Feature Interaction Descriptor for Pedestrian Detection, *IEICE Trans. Inf. Syst.*, Vol.E93-D, pp.2656–2659 (2010).
[3] Dalal, N. and Triggs, B.: Histograms of oriented gradients for human detection, *CVPR*, pp.886–893 (2005).
[4] Dollar, P., Belongie, S. and Perona, P.: The Fastest Pedestrian Detector in the West, *BMVC* (2010).
[5] Dollár, P., Wojek, C., Schiele, B. and Perona, P.: Pedestrian Detection: An Evaluation of the State of the Art, *PAMI*, pp.743–761 (2012).
[6] Felzenszwalb, P.F., Girshick, R.B., McAllester, D. and Ramanan, D.: Object Detection with Discriminatively Trained Part Based Models, *PAMI*, pp.1627–1645 (2010).
[7] Goto, Y., Tsuchiya, M., Yamauchi, Y. and Fujiyoshi, H.: Fast Discrimination by Early Judgment Using Linear Classifier Based on Approximation Calculation, *IEICE-D* (*Japanese*), Vol.97, No.2, pp.294–302 (2014).
[8] Hare, S., Saffari, A. and Torr, P.H.S.: Efficient online structured output learning for keypoint-based object tracking, *CVPR*, pp.1894–1901 (2012).
[9] Watanabe, T., Ito, S. and Yokoi, K.: Co-occurrence Histograms of Oriented Gradients for Human Detection, *IPSJ Transactions on Computer Vision and Applications* (*CVA*), Vol.2, pp.39–47 (2010).

(Communicated by *Tatsuya Harada*)