International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:5, No:7, 2011

A Fast Object Detection Method with Rotation Invariant Features

Zilong He and Yuesheng Zhu

***Abstract*—**Based on the combined shape feature and texture feature, a fast object detection method with rotation invariant features is proposed in this paper. A quick template matching scheme based online learning designed for online applications is also introduced in this paper. The experimental results have shown that the proposed approach has the features of lower computation complexity and higher detection rate, while keeping almost the same performance compared to the HOG-based method, and can be more suitable for run time applications.

***Keywords*—**gradient feature, online learning, rotation invariance, template feature

I. Introduction

REAL-TIME accurate object detection has widespread applications, such as video surveillance, driving assistance system, and content-based image retrieval. Currently, the general object recognition approaches are based on single shape feature or single texture feature. The typical dominant methods like HOG [1] require a significant training time to build a classifier offline. These methods will therefore need to be constantly retrained given situations in which they have to continuously learn about new objects online.

In this paper, a fast object detection method with rotation invariant features and a quick template match scheme based online learning [2] is designed to enhance the performance in terms of robustness, computation complexity and detection accuracy for online applications. The new method and scheme will be described in details below.

II. Template Feature Extraction

The proposed approach is based on template matching and the gradient is used as the shape feature and LBP [3] feature as texture feature. One of the advantages of our algorithm is that every single bit of an 8-bit integer can be used to represent the template features. Its processing is therefore simple and it can run faster than other object recognition algorithms, such as HOG. Additionally, this approach is also rotation invariant.

Zilong He and Yuesheng Zhu are with Communication & Information Security Lab, Shenzhen Graduate School, Peking University, Shenzhen, CHINA. Corresponding author e-mail add:, zhuys@szpku.edu.cn.

A. Template Feature Computation Method

The regular steps of computing the gradient of an image is as follows. First, the template is divided into big blocks, and then every big block is divided into small cells [4]. Each feature can be defined by its cell position C(x_c , y_c , w_c , h_c), the parent block position B(x_b, y_b, w_b, h_b) and the orientation bin number k, so each feature f is denoted by f(C, B, K). The gradients at the point (x, y) of image *I* can be found by convolving gradient operator with the image [5]:

$$G_x(x,y) = [-101] * I(x,y) \tag{1}$$

and

$$G_y(x,y) = [-101]^T * I(x,y) \tag{2}$$

The magnitude of the gradient at the point (x, y) is

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \tag{3}$$

The orientation of the edge at the point (x, y) is

$$\theta(x,y) = \arctan\left(\frac{G_y(x,y)}{G_x(x,y)}\right) \tag{4}$$

Then the orientation range [0, π] is divided into K bins and the value of k_{th} bin to be denoted as:

$$\Psi_k(x,y) = \begin{cases} G(x,y) & if \quad \theta(x,y) \in bin_k \\ 0 & otherwise \end{cases} \tag{5}$$

Now our gradient feature is not the same as HOG because the difference of the gradient feature is used instead of the gradient feature itself. In this paper, use the present pixel's gradient orientation angle value to minus the front pixel's gradient orientation value to get a gradient orientation difference value. This value will then be divided into k bins. Fig. 1 shows the procedures of the template feature extraction

of our approach.

Proposed Algorithm

Input: template image.

1: image pyramids and rotation translation;
2: multiple resolution division into blocks;
3: **For** i=1:N
4: get one block and divide bolck into cells;
5: **For** i=1:M
6: get one cell;
7: calculate gradient orientation difference of the dominant pixels;
8: calculate LBP feature;
9: binary representation;
10: **end for**;
11: cyclic coding and add to list;
12: **end for**

Output: template feature.

Fig. 1 Procedures of the template feature extraction

B. Template Feature Extaction of the Dominant Pixels

To make our algorithm robust to small deformation and faster to compute, every gradient of every pixel is not considered like DOT [6]. The template image is divided into small rectangle grids called blocks, and then the blocks are divided into rectangular grids called cells [4] as illustrated in Fig. 2. Only the gradient orientation of the dominant pixel that has a large magnitude is considered. Another benefit of this is that it can simplify the extraction of the gradient orientation difference. Now, only the differences between the dominant gradient orientation in the current rectangle region and the most dominant gradient orientation of the previous rectangle region need to be calculated. Fig. 3 shows the orientation and magnitude of the dominant pixels in the current cell.

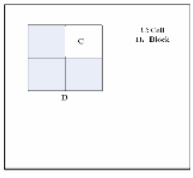

Fig. 2 The template region division

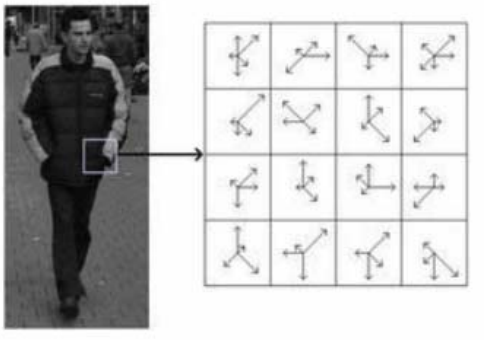

Fig. 3 The orientation and magnitude of the dominant pixels

C. Binary Representation for Gradient Orientation Difference

TABLE I
BINARY PRESENTATION OF ONE PIXEL

(-10° ,10°)	[- 15° ,- 10°)] [10° ,15°)	[15° ,40°)	[40° ,65°)	[65° ,90°]	(-40° ,-15°]	(-65° ,40°]	(-90° ,-65°]	no dominant pixels in cell
00000000_b	00001000b	00000100b	00000010b	00000001b	00010000b	00100000b	01000000b	10000000b

For efficient template matching, using the idea of literature [7], a binary representation method is developed here to use every single bit of a byte, i.e. an 8-bit integer, to represent the bins of the dominant gradient orientation differences. As shown in Table I, if the pixel has obvious gradient in current cell and the gradient orientation angel difference between it and the most dominant pixel in the previous cell is smaller than 10 degree, then the gradient orientation angel difference for the current dominant is represented by 00000000b. A difference between 10 and 15 degrees is represented by 00001000b. If the current pixel's gradient orientation angle is larger than the gradient orientation angel of the most dominant pixel in the previous region by 15 to 40 degrees, then the difference is represented by 00000100b. Similarly, a difference of 40 to 65 degrees is represented by 00000010b; a difference of 65 to 90 degree is represented by 00000001b. When the gradient orientation angle of the current pixel is smaller than gradient orientation angel of the most dominant pixel in the previous region by 15 to 40 degrees, the difference is represented by 00010000b. In a similar fashion, a difference of 40 to 65 degrees is represented by 00100000b, and a difference of 65 to 90 degrees is represented by 01000000b. Then, after calculating every dominant pixel's gradient orientation difference, the feature of a cell F_{CS} is defined as:

$$F_{CS} = F_{P1} \oplus F_{P2} \oplus F_{P3} \cdots \oplus F_{Pn} \quad (6)$$

where F_{pm} denotes the m_{th} dominant pixel, n denotes the total number of the dominant pixels, and $\oplus$ denotes bitwise OR operation. However, if there are no dominant pixels in the cell, then represent F_{CS} with the 8-bit integer 10000000b.

At last, after calculating every cell's gradient orientation difference feature in a block, a sequence of bytes that can represent a block's gradient orientation difference feature F_{BS} is obtained.

D. Achieving Rotation Invariance

It is noted that the block gradient orientation difference feature F_{BS} is still not rotationally invariant. If the image is rotated, the position of the cell in the block will be changed. The cell's feature which is first calculated will be changed. To remove the effect of rotation, after calculating the feature of every block, we define:

$$F_{BSR} = \min\{ROR(F_{BS}, i) \mid i = 0,1,..., P-1\} \quad (7)$$

where P is the number of cells in one block, and ROR() performs a byte right shift on the P-bytes number F_{BS} i times.

Then a list of F_{BSR} to represent the shape feature of the template is constructed and referred to F_{TS} .

E. Add LBP Feature

The method in Literature [6] performs not very well with respect to textured object, especially when the templates do not exhibit strong gradients. To solve this problem, the LBP feature is involved in our approach. Fig. 4 shows how to add LBP feature to the template feature.

LBP feature is a kind of gray scale and rotation invariant texture operator, also it is robust to slight varying illumination. Our algorithm will adopt the $LBP_{8,1}$ feature which also uses one byte to represent LBP feature. So the proposed approach combines these two kinds of features to represent the template feature. It only takes 2 bytes to represent the feature of one cell of the block.

At last, after calculating every cell's texture feature in a block, a sequence of bytes that can represent a block's texture feature F_{BT} is obtained.

However, F_{BT} is still not rotation invariant, too. This is because if the image is rotated, the position of the cell whose feature is first calculated will be changed. To remove the effect of rotation, after calculating the feature of every block, we define:

$$F_{BTR} = \min\{ROR(F_{BT}, i) \mid i = 0,1,..., P-1\} \quad (8)$$

where P is the number of cells in one block, and ROR() performs a byte right shift on the P-bytes number F_{BT} i times.

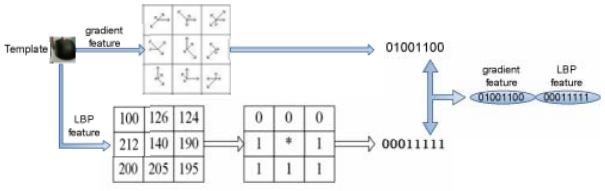

Fig. 4 The combine of gradient feature and LBP feature

Then, after calculating every block's texture feature in the template, a list of F_{BTR} to represent the texture feature of the template is constructed and referred to F_{TT}.

F. Multiresolution Division

To reduce the effect of the varying size of the object due to the camera's adjustment or object's small motions, the template image is divided into blocks in a multi-resolution way. That means the block size is different due to the different division and the cell size is the same.

G. Image pyramids and rotation translation

To make our measure tolerant to small deformations, the image pyramids and rotation translation will be executed before calculating the template feature. It will cost additional computation time but it will improve the detection rate.

III. Template Match

A. Template feature extraction of the input video

Our template matching process is based on every block. The feature of every region is calculated with same size of template image except for the image pyramids and rotation translation. Besides, only the gradient orientation angel differences between the strongest gradients in current cell and the front cell need to be calculated. The reason is that it can make it run fast and be robust to small deformation.

B. The prinple of template match

The next step is to measure the similarity between the feature of every region in the input image and the template image. We can do that with a bitwise AND operation and a

bit-count operation. In fact, it is computed with bitwise operations on the binary representations of the template feature F_T and input image feature F_I. The similarity measure is just a count that is simply the number of bits where both F_T and F_I are equal to 1. The similarity measure can be obtained efficiently by a bit-count on an 8-bit integer:

$$e = bitcount\ (F_T \otimes F_I) \tag{9}$$

where ⊗ denotes bitwise AND, e is the similarity. Currently the parallel bit-count method from [8] is fast. So we use the bit-count method in [8]. As to threshold, a threshold τ is used for determining matches. If $e \geq \tau$, the region in the input image matches the template. That is, the place in the input image is the matched place for the template. If $e \leq \tau$ then it means that the region is not the matched place. Fig. 5 illustrates the matching score of the proposed method.

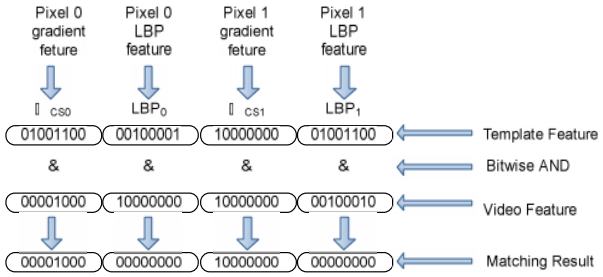

Fig. 5 Matching score bitwise operation

IV. Onlie Learing

Currently, the dominant approach to object recognition is to use statistical learning to build a classifier offline, and then to use it at run-time for the recognition [9]. Hence, the current dominant methods like HOG[1] require a significant training time to build a classifier offline and is not suitable for all scenarios, for example, for a online system. The time consuming training stage will result in losing efficiency while updating the classifier.

An approach based on real-time template recognition that does not require a time consuming training stag is proposed in [9]. It is trivial and virtually instantaneous to learn new incoming objects by simply adding new templates to the database while simultaneously maintaining reliable real-time recognition. In this point, our approach is the same as [9].

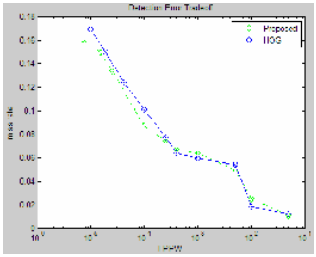

Fig. 6 Detection Error Tradeoff

V. Expriment Results

A. Detection Rate

Fig. 7 Experiment result for human match

As illustrated in Fig. 7, the grey rectangle denotes the detection region, and the white region denotes the edge of the template to be detected. The result shows that the person is recognized successfully.

As illustrated in Fig. 6, compared to HOG, our detection rate gets 91.3% at the level of 10^{-4} FPPW, while HOG gets 89.9%, and means our approach outperforms HOG in a low level of FPPW. In high level of FPPW, our approach keeps the almost same performance as HOG.

As illustrated in Fig. 8, the results show that our approach method has a good detection rate both in textured object and texture-less object with a low FFPW. Fig. 9 shows the original frame of the matching result.

B. Speed

The experiment was done on a computer with Intel Centrino Processor Core2Duo 2GHz and 2GB DDR3 SDRAM. The resolution of the video sequences is 640 × 480. The experiments show that it takes an average of 0.9 second to

extract features from the template image. Compared to HOG, the training time of our approach is at least 130 times faster than that of HOG, using 2000 templates. This is because our approach can skip most unimportant pixel locations that do not have obvious gradients. Additionally, the binary representation of the template feature can be computed with bitwise operations, which make our extraction of template features requires much less computation.

Fig. 8 Other experiment results

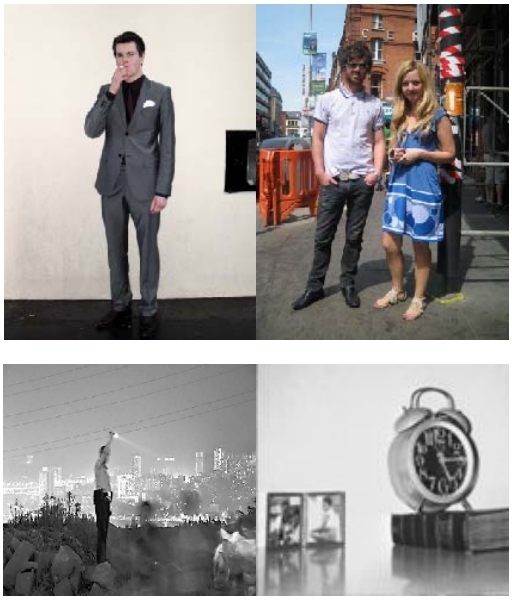

Fig. 9 The original frame of the matching result

C. Video Sequence Test

Fig. 10 shows the matching result under the condition that the target's size changes since she walked straightly at first then turn right.

Fig. 10 Experiment result for the condition of size changing

D. Rotation Invariance and Robust Performance

As showed in Fig. 11 under the condition that rotation and deformation has happened to the recognized target, and even the gray scale is changed. The target is still matched successfully.

Fig. 11 Test of the rotation invariance and robust performance

E. Online Learning

To test the online learning of the proposed method, we change the recognized target from mouse to a picture on the cup as showed in Fig. 12. It successfully locates the position of the picture on the cup quickly.

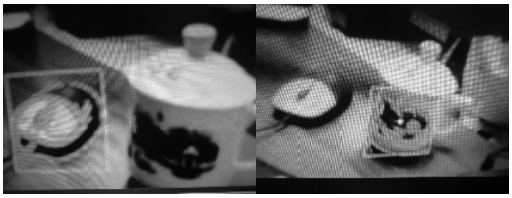

Fig. 12 Online learning test

F. Proportion of gradient feature to LBP feaute

The proportion of gradient feature to LBP feature is an important parameter to the result of our approach. Its value is not critical by theory. Our experiments show that when the proportion of gradient feature is about 80%, the result is the best and the detection rate for our test dataset gets 91.3% at the level of 10^{-4} FPPW as showed in Fig. 13.

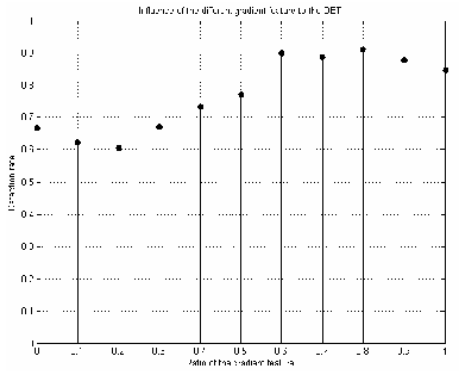

Fig. 13 Detection rate for different ratio of gradient feature

VI. Conclusion

Our proposed method based on template match and uses both gradient and LBP feature as the template feature is gray-scale invariant and robust to rotation and small deformation due to its simplification of process, transformation. The experimental results have demonstrated that the proposed scheme has the features of lower computation complexity, higher detection rate while maintaining approximately the same performance compared to the HOG-based method, and can be more suitable for run time applications.

References

[1] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In CVPR, 2005.

[2] M.Grabner, H.Grabner, and H.Bischof. Semi-supervised on-line boosting for robust tracking. In ECCV, 2008.

[3] Timo Ojala, Matti and Topi. Multiresolution Gray-Scale and Rotation Invariiant Texture Classification with Local Binary Patterns. IEEE Trans on Pattern Analysis And Machine Intelligence, 2002.

[4] Qing Jun Wang and Ru Bo Zhang. LPP-HOG: A New Local Image Descriptor for Fast Human Detection. In KAM, 2008.

[5] Hui-Xing Jia, Yu-Jin Zhang. Fast Human Detection by Boosting Histograms of Oriented Gradients. In ICIG, 2007.

[6] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Pascal Fua and Nassir Navab. Dominant Orientation Templates for Real-Time Detection of Texture-Less Objects. In CVPR, 2009.

[7] Simon Taylor and Tom Drummond. Multiple Target Localisation at over 100 FPS. In BMVC, 2009.

[8] M.Ozuysal, P.Fua, and V.Lepetit. Fast key point recognition in ten lines of code. In CVPR, 2007.

[9] P. Viola and M. Jones. Robust real-time object detection. IJCV, 2001.