REALTIME PEDESTRIAN DETECTION SYSTEM USING COMBINATIONS OF MULTIPLE FEATURES AND OBJECT TRACKING METHOD

[1] *Zhe-yuan Gao(高哲遠)*, [2] *Tsu-Rong Chiang(江祖榮)*, [3] *Chi-Fu Lin(林奇賦)*, [4] *Yun-che Tsai(蔡昀哲)*
[5] *Chiou-Shann Fuh (傅楸善)*

Dept. of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan
1E-mail: gaozheyuan13@gmail.com
[2]E-mail: cutesciuridae@gmail.com
[3]E-mail: daky1983@gmail.com
[4]E-mail: jpm9ie8c@gmail.com
[5]E-mail: fuh@csie.ntu.edu.tw

ABSTRACT

This paper proposed a pedestrian detection framework using combinations of features (including Histogram of Oriented Gradient, Local Binary Pattern, and Color Self-Similarity Features), and we used the Lucas-Kanade method for tracking the pedestrian which we have detected before. To increase the detecting accuracy, we also derived an equation which exhibits the relationship between the pedestrian height and feet position in an image. To accelerate our program, we apply a strategy which could infer the features that we have acquired. Finally, we build a realtime pedestrian detection framework based on the methods introduced above.

Keywords *Realtime Pedestrian Detection, Geometric Constraints, Acceleration techniques.*

1. INTRODUCTION

Pedestrian Detection is a popular topic among most of the computer vision problems, because of many vehicle manufacturers putting the security of their driving systems at the first place. Avoiding collision with pedestrians becomes the real impetus of researching into this topic.

Digital Camera and Computer Vision Laboratory, Department of Computer Science and Information Engineering, National Taiwan University conducts the mission of applying the computer vision technology to the real-world application, and also carries the mission of improving the computer vision industry in Taiwan. We conduct the annual project of Ministry of Science and Technology, Executive Yuan by choosing the topic of pedestrian detection. Many companies in Taiwan manufacture driving recorders such as Syntek Semiconductor (太欣半導體), Ambarella (安霸), Service & Quality (倚強), and Novatek (聯詠). The research into the pedestrian detection could help us build a cooperative relationship with those companies and help driving recorders in Taiwan improve their functions and qualities. Meanwhile, we aim to increase the market share in the field of driving recorders.

2.FEATURE EXTRACTION

2.1. Histogram of Oriented Gradients

N. Dalal and B. Triggs [2] proposed a dense and overlapping description of image regions called HOG (Histogram of Oriented Gradients) features to classify the pedestrian in the image. It outperforms some previous works [17][19][20] using Haar-like wavelets. Classical HOG descriptors are R-HOG (Rectangular HOG) descriptors, but it also has some modified versions such as C-HOG (Circular HOG), Bar HOG and Center-Surrounding HOG.

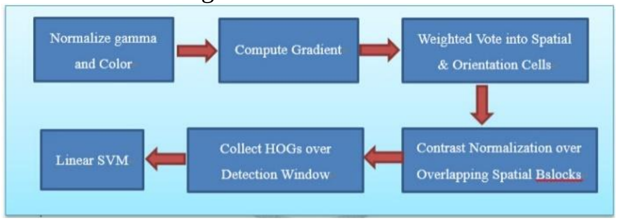

Normalize gamma and Color
Compute Gradient
Weighted Vote into Spatial & Orientation Cells
Contrast Normalization over Overlapping Spatial Blocks
Collect HOGs over Detection Window
Linear SVM

Fig. 1: An example of Figure.

HOG features are extracted from the gray image, to avoid some extreme conditions caused by color or illumination bias (such as strong or red light conditions). We first apply an optional process called Gamma/Color Normalization to make the image look more neutral. Gamma normalization includes square root and log compression, which takes the square root and log to the pixel values. Experiments show that square root compression improves performance by 1% at 10^{-4} FPPW (False Positive Per Window), however log compression reduces performance by 2% at 10^{-4} FPPW [2]. This step is optional, and we suggest not taking this step, because the improvement is negligible.

Before extracting the HOG features we have to calculate the gradient images. We used the 1-D centered operator which uses [-1,0,1] for calculating the x-gradient image and $[-1,0,1]^T$ for y-gradient image, and next step is to summarize of local patch of image, every pixel contributes a weighted vote to the local spatial regions which centers on it. These spatial regions are called cells. Orientation bins are evenly spaced from 0°

to 180° (unsigned gradient) or 360° (signed gradient). Experiments show that 9 bins with directions from 0° to 180° achieve the best performance [1].

A strategy can be applied in binning. Suppose θ_1 and θ_2 are two angle values; $h(\theta)$ is the bin which centers at angle θ; a new angle value θ' is between the two values $(\theta_1 < \theta' < \theta_2)$; and b is the angle width of a bin. Then we can distribute the magnitude value w to the two bins $h(\theta_1)$ and $h(\theta_2)$ as follows:

$$h(\theta_1) \leftarrow h(\theta_1) + w\left(1 - \frac{\theta - \theta_1}{b}\right)$$

$$h(\theta_2) \leftarrow h(\theta_2) + w\left(\frac{\theta - \theta_1}{b}\right)$$

Since the strengths of the gradients usually have a wide range because of the different illumination conditions and contrasts, we have to apply a method to reduce the effect. The normalization step [1] groups cells into larger spatial blocks, and blocks are usually overlapped so that cells can contribute to more than one block. Typical methods of normalization steps are:

Table 1. Different normalization forms

Normalization Method	Equation
L2- norm	$v \leftarrow v/\sqrt{\lVert v\rVert_2^2 + \varepsilon^2}$
L1-norm	$v \leftarrow v/(\lVert v\rVert + \varepsilon)$
L1-sqrt	$v \leftarrow \sqrt{v/(\lVert v\rVert + \varepsilon)}$

2.2. Local Binary Features

Local binary pattern is a kind of feature vector which is first proposed by T. Ojala et al.[18], which is used primarily for the classification of textures. Later in 2009, X. Wang [29] applied the Local Binary Pattern with HOG to improve the detecting rate of pedestrians. The process of extracting the LBP features is similar to the CENTRIST features except for computing the gradient image first.

LBP features are computed by computing the relative relationship of a pixel with its neighbors; usually LBP compares the eight pixels which are directly adjacent to the pixel to be computed. However, there are also some patterns computed by comparing with the pixels with farther distances.

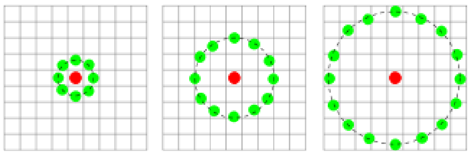


Fig. 2 Different patterns for computing the Local Binary Pattern values [18].

In our program, we choose the first pattern in Fig. 2 for feature extraction. We compare the gray value of every pixel with its eight neighbor pixels, when the gray value is larger than the one of its neighbor values, it is valued 1 at the corresponding bit, and otherwise it is valued 0.

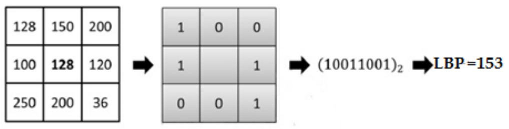


Fig. 3 Extracting the LBP features by computing the relative relationship with its neighboring pixels.

After computing the LBP values of every pixel, we can extract the LBP features by forming different sizes of blocks, to be consistent with the computation of HOG features. We define the block size to be 16×16 pixels and make a summarization of the number of specific LBP values in the image block ($2^8 = 256$ bins).

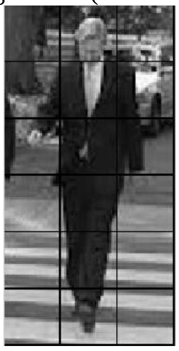


Fig. 4 Feature blocks on a pedestrian image.

Block normalization is also required in computing the Local Binary Pattern features. To accelerate our processing of block normalization, we use different strategies with the block normalization in HOG features, by dividing the number of features in a block.

$$v \leftarrow v/sum(v)$$

2.3. Color Self-Similarity(CSS) Features

Low-level features are also important to the improvement of detection rate [6][23][31]. In our program, we apply Color Self-Similarity (CSS) which uses color information to improve our detection rate.

Color itself exhibits some structures because colors of some regions are locally similar. For instance, the pedestrian is line symmetric with its axis at the center of the pedestrian, also, the color of the pedestrian's hair, skin, clothes, and trousers also exhibit some local structures. To encode this color information as features, we choose Hue-Saturation-Value (HSV) color space of image to compute local color histograms over 8x8 pixel block, and compute the histogram intersection to encode the color difference information.

Color histogram is computed by discretizing the image colors and the number of pixels belonging to every bin is counted. The histograms are invariant to translation and rotation about the viewing axis and slowly change in scale and occlusion. Of all the color spaces which are extracted as features, HSV (Hue Saturation Value) outperforms most of other color spaces [28]. HSV is a color space which transforms RGB color space into cylindrical coordinate system.

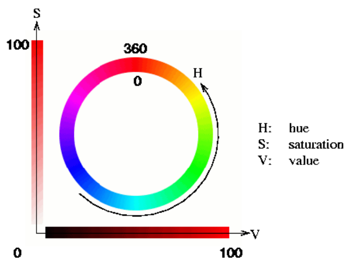


Fig. 5 HSV color space

Hue (0-360): The basic property, which represents the color, such as red, yellow, blue, and so on.

Saturation (0-100%): The purity of color, the higher the value means the purer and more saturated of the color, the lower value represents the color will become grayer.

Value (0-100%) (Lightness): the higher the value represents brighter of the color.

$$h = \begin{cases} 0^\circ & if\ max = min \\ 60^\circ \times \frac{g-b}{max-min} + 0^\circ & if\ max = r \text{ and } g \geq b \\ 60^\circ \times \frac{g-b}{max-min} + 360^\circ & if\ max = r \text{ and } g < b \\ 60^\circ \times \frac{b-r}{max-min} + 120^\circ & if\ max = g \\ 60^\circ \times \frac{r-g}{max-min} + 240^\circ & if\ max = b \end{cases}$$

$$s = \begin{cases} 0 & if\ max = 0 \\ (max - min)/\ max & otherwise \end{cases}$$

$$\mathrm{v} = max$$

We choose to compute the histogram intersection value as the measurement of color similarities. We divide the image into non-overlapping cells, each cell comprises 8x8 pixel regions, and we first transform the representation of color spaces from RGB to HSV. The range of a specific bin is achieved by evenly dividing a color space into 3 sub-spaces, and all together there are 3x3x3 bins. We calculate the bin that each pixel in the cell belongs to and add 1 to that bin. Finally we will achieve a histogram containing 27 bins. The value of i^{th} bin of the histogram of cell K is defined as K_i, then the histogram intersection value $H(I, M)$ of cell I and M is computed by:

$$H(I, M) = \frac{1}{2\sum_{i=1}^{27} I_i} \sum_{i=1}^{27} |I_i - M_i|$$

Suppose an image of 96×48 pixels, they have 12× 6=72 cells, by computing the histogram intersection value of every two cells, we can get 72×71/2=2556 features.

2.4. Integral Image Feature Computation

The three types of features introduced require the summarization of features in a specific region. For detection step, if we make summarization of features every time, it would be a very time-consuming step. To quickly get the features in a specific region, the concept of integral image should be applied in our method. The concept of integral image is first proposed by Franklin C. Crow and introduced into the pedestrian detection framework by P. Viola and M. Jones [26]. Using integral images, the statistics of features can be easily computed by just a few calculations.

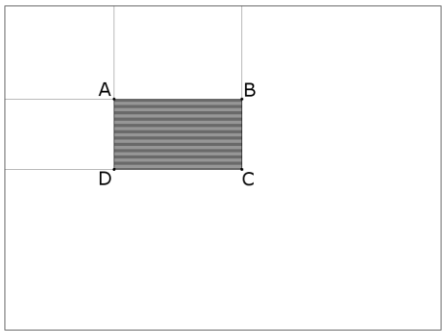


Fig. 6 Sample of integral image.

$\boldsymbol{h} =$

Take Fig. 6 as a sample of integral image. For a single point (A,B,C,D), the values on the point is the summarization of the rectangle area which is to the left and up. If we intend to get the features in rectangle ABCD, we can simply calculate it by:

$$F(A, B, C, D) = F(A) + F(D) - F(B) - F(C)$$

where *F(X)* is the summarization of features of the rectangle region which is to the left and up of point X, that is, also the values of point X on integral image.

3. OBJECT TRACKING

Object tracking is an important step to save detection time, since the positions of the same object in two consecutive images are relatively close to each other, it is not necessary to compute the features again in the next frame for the detecting step, instead, we just need to track the object in the block and move the detecting block as the target is moving. In our program, we use the Lucas-Kanade [16] method tracking for tracking the object.

Fig.7 Sample images from human motion database [2].

3.1 Introduction

Usually the method of computing the movement of object is to compare the sum of absolute difference of pixels between two different blocks. When a detecting window $d_i(r, w, l)$ (where r is the center position; w is the width; and l is the height of the window captures a pedestrian in the i^{th} frame) is found, we can search the nearby positions in the $(i + 1)^{\text{th}}$ frame, comparing the sum absolute value of pixel-wise difference and finding the minimum value as our target.

$$\vec{r} = min_{\vec{r}}\ abs[sum(d_i(r, w, l) - d_{i+1}(r + \vec{r}, w, l)]$$

However, the exhaustive search for $\vec{r}$ requires much computational time, we can introduce a fast optical flow computation method called Lucas-Kanade method [16].

3.2 Lucas-Kanade Method

Lucas-Kanade method assumes that the displacements of two images are small and constant within a neighborhood of point p. Thus, the local image flow vector (v_x, v_y) must follow:

$$\begin{cases} I_x(q_1)v_x + I_y(q_1)v_y = -I_t(q_1) \\ I_x(q_2)v_x + I_y(q_2)v_y = -I_t(q_2) \\ \vdots \\ I_x(q_n)v_x + I_y(q_n)v_y = -I_t(q_n) \end{cases}$$

where q_1-q_n are the pixels inside the window; $I_x(q_i)$, $I_y(q_i)$, and $I_t(q_i)$ are the partial derivative values at pixel position q_i of image I with the direction of x, y and time t. We can get:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i)I_y(q_i) \\ \sum_i I_y(q_i)I_x(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i)I_t(q_i) \\ -\sum_i I_y(q_i)I_t(q_i) \end{bmatrix}$$

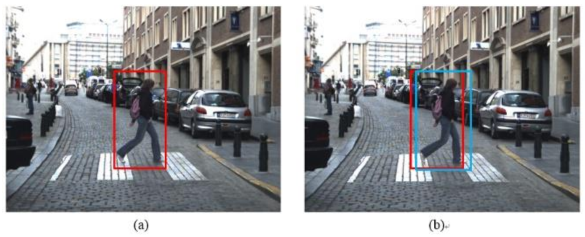


Fig.8 Sample of tracking a pedestrian in two consecutive images, red rectangle in (a) represents a found pedestrian and in (b) is the same position with (a), and the blue rectangle is the new rectangle to be computed by Lucas-Kanade optical flow method.

In our experiment, computing the optical flow of two pedestrian requires ≤ 0.001 seconds, which proves Lucas-Kanade method is very efficient for optical flow computation.

4. POST PROCESSING PROCEDURES

4.1. Geometric Constraints

It is a reasonable approach to use some basic geometric constraints to rule out the detected bounding boxes which are apparently not appropriate in the image. Usually the video recorder is placed in front of cars and taking videos while the car is running. We can use some basic assumptions to infer relations among several parameters.

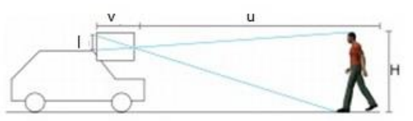


Fig. 9 Scene geometry.

We use the pinhole camera model to simplify our discussion, using the definition of similar triangles we can easily tell that the relationship between the four parameters: H (pedestrian height), u (object distance), v (image distance), and l (image height):

$$\frac{\boldsymbol{H}}{\boldsymbol{u}} = \frac{\boldsymbol{l}}{\boldsymbol{v}}$$

However, the cameras are usually placed not parallel to the ground, but with a specific angle θ_s towards the ground.

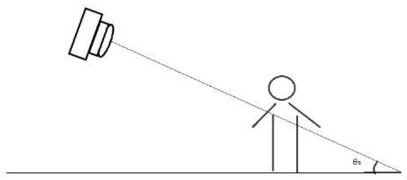


Fig. 10 Camera placement in real situation which has an angle towards the ground.

To simplify the model, we can change the observation angle by assuming that the camera is horizontal and the ground has an angle θ_s, which simplifies our analysis.

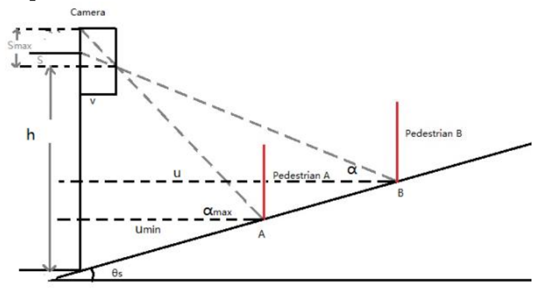


Fig. 11 Specific figure for analyzing the spatial relationships of different variables.

$$\tan\alpha_{max} = \frac{h - \tan\theta_s u_{min}}{u_{min}} = \frac{s_{max}}{v}$$

$$\tan\alpha = \frac{h - \tan\theta_s u}{u} = \frac{s}{v}$$

$$\frac{s_{max}}{s} = \frac{\frac{h}{u_{min}} - \tan\theta_s}{\frac{h}{u} - \tan\theta_s}$$

We can replace the variable of object distance u with the equation above:

$$\frac{s_{max}}{s} = \frac{\frac{h}{Hv} * l_{max} - \tan\theta_s}{\frac{h}{Hv} * l - \tan\theta_s}$$

Since the variables s_{max}, h, H, v, l_{max}, $\tan\theta_s$ are constant, we can rewrite the above equation as:

$$s = s_{max} * \frac{\frac{h}{Hv} * l - \tan\theta_s}{\frac{h}{Hv} l_{max} - \tan\theta_s}$$

Set $C = \frac{s_{max} * \frac{h}{Hv}}{\frac{h}{Hv} l_{max} - \tan\theta_s}$, $D = -\frac{s_{max}\tan\theta_s}{\frac{h}{Hv} l_{max} - \tan\theta_s}$

$$s = Cl + D$$

From the result we can see that *s* (the length from the pedestrian's feet to the center of the image) is linearly related to *l* (the height of the pedestrian on the image).

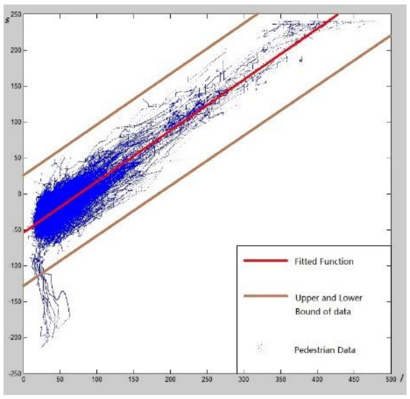


Fig. 12 Blue dots represent the positions of *(s, l)* read from annotation of dataset-USA of Caltech pedestrian data [4], and red line represents the fitted function.

After we sampled all the pedestrian data, we can use a linear function to fit the data and get $s = 0.709 * l - 54.17$. We can set two bounds (upper and lower) to include majority (>95%) of pedestrian data of $(s + 75) = 0.709 * l - 54.17$ and $(s - 75) = 0.709 * l - 54.17$. According to the two functions, we can set the detection region according to different pedestrian heights.

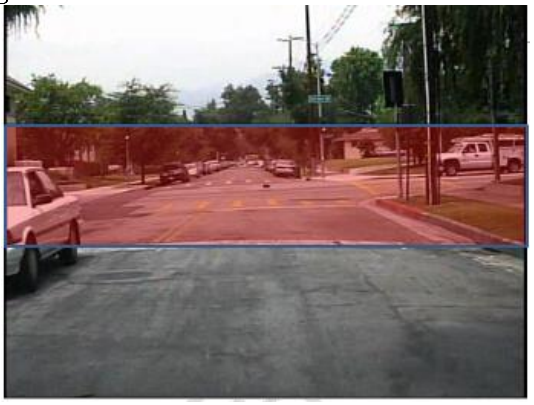


Fig. 13 Search region for pedestrian height of 50 pixels.

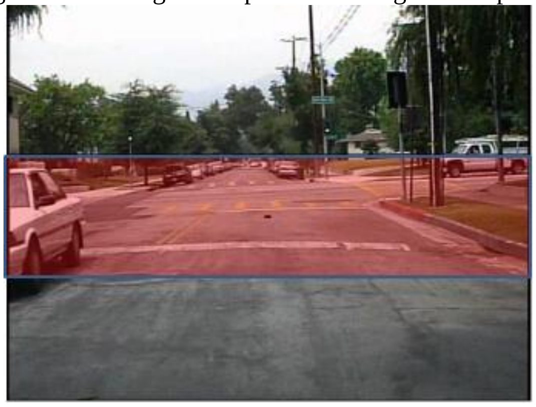


Fig. 14 Search region for pedestrian height of 100 pixels.

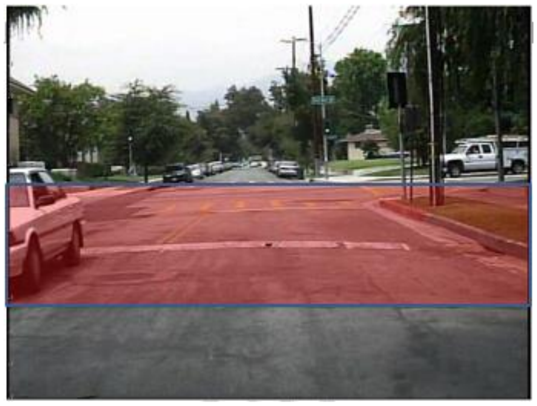


Fig. 15 Search region for pedestrian height of 150 pixels.

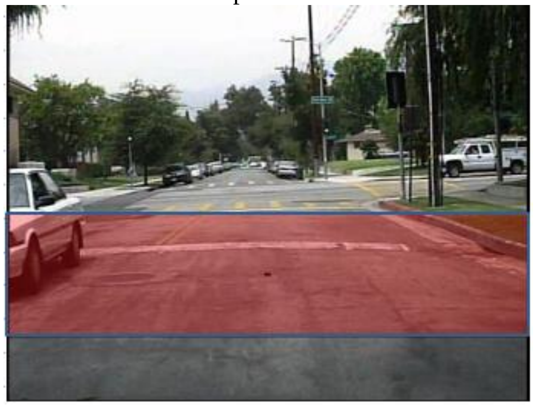


Fig. 16 Search region for pedestrian height of 200 pixels.

4.2 Non-Maximal Suppression (NMS)

Detection window in scanning the images will output some blocks in similar sizes at very close places. Non-maximal suppression solves the problem by first checking whether the overlap of the two blocks exceeds a threshold NEAR_RATIO (by default =0.8). If some blocks are judged as the same group, it will output the block with the highest score.

4.3 Repetition around Pedestrians

Experimental results show that usually there are many detection windows of similar sizes and locations around human figures. Moreover, a false positive object is usually surrounded by very few detection windows. It serves as a clue to eliminate some false detected windows. We define the minimum number of repetition detection windows as three. If a target is surrounded by fewer than three windows, then it is not considered as a pedestrian.

5. ACCELERATION TECHNIQUES

5.1 Introduction

The speed of detecting the pedestrian is critical due to the limited capability of computation of embedded system of driving recorders. Improvement of detecting accuracy is usually accompanied by the increase of detecting time. The time for state-of-art detectors of processing a frame on 640*480 pixel video requires 2-30 seconds [7]. We adopt several speed improvement techniques to accelerate the detecting speed.

5.2 Open Multi-Processing (OpenMP) Paralleliztion

Open Multi-Processing (OpenMP) [30] is an API which supports easy implementation of multiprocessing of multi-platform shared memory programming in C, C++, and Fortran. Supporting processor architectures and operating systems include Solaris, AIX, HP-UX, GNU/Linux, MacOS X, and Windows platforms. It consists of compiler directives, library routines and environment variables which influence the behavior of run-time programs.

An important prerequisite for parallelizing a program is that different tasks can be conducted by different threads, which means that the accomplishment of a task does not rely on the running of other tasks. Parallelizing a segmentation of loop program simply needs to add the support of OpenMP and add a notation before for loop.

Before parallelization:
```
for ( int i=0; i < N; i++)
{
        //tasks to be executed
}
```
After parallelization:
```
#pragma omp parallel for
for ( int i=0; i < N; i++)
{
        //tasks to be executed
}
```

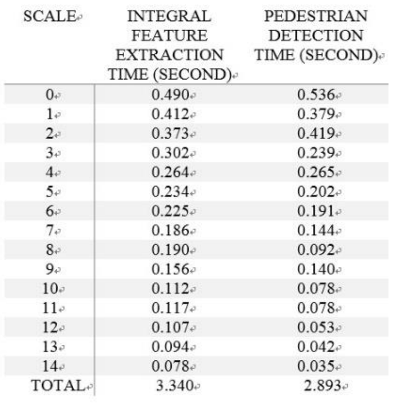
In our program, the task which is able to be parallelized is the extraction and detection process of image blocks from whole pictures. Since the tasks of detecting small spatial regions are independent of each other, we consider parallelizing this process to accelerate our program. The amount of acceleration depends on the core number of the multicore computers e.g. twice as fast on Asus Desktop computer with dual core.

5.3 Inferring Features under Different Image Scales

The high cost of the pedestrian detection is caused by the calculation of integral images and detecting pedestrians of scanning every spatial region in images of different scales. Table.2 shows the amount of time consumed by calculation of HOG integral feature and detecting every spatial region, and the width and height of the image in a layer of pyramid is 10/11 times smaller than the image in the next layer of pyramid.

Table. 2 The time to process the images of different scales.

Image pyramids:



SCALE	INTEGRAL FEATURE EXTRACTION TIME (SECOND)	PEDESTRIAN DETECTION TIME (SECOND)
0	0.490	0.536
1	0.412	0.379
2	0.373	0.419
3	0.302	0.239
4	0.264	0.265
5	0.234	0.202
6	0.225	0.191
7	0.186	0.144
8	0.190	0.092
9	0.156	0.140
10	0.112	0.078
11	0.117	0.078
12	0.107	0.053
13	0.094	0.042
14	0.078	0.035
TOTAL	3.340	2.893

In my pedestrian detecting program, a feature extraction strategy without resizing images is introduced for acceleration [5] :

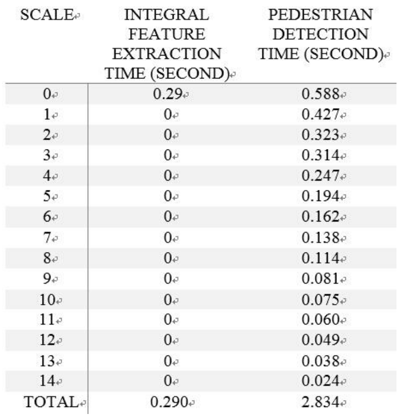
Suppose $I(x,y)$ denote an m*n discrete signal and Ω is shift-invariant function which takes $I(i,j)$ and creates a new channel image $C = \Omega(I)$, $f(I) = \sum_{ij} C(i,j)$ is the global sum of a channel which is called channel energy. Many features can be applied for this formula including gradient histograms, linear filters, color statistics and so on. Finally we write $f(I,s)$ to represent the channel energy computed over I after down-sampling the image by a factor of 2^s. We can get a conclusion introduced by P. Dollar[5]:

$$r(I,s) = \frac{f(I,s)}{f(I,0)} = 0.89 * exp(-1.099s)$$

From the above equation, we do not need to compute integral features every time. Instead, we can quickly infer the features at any scale by just computing the integral feature for one time.

Table. 3 The time to process the images of different scales after applying feature inferring steps.

Image pyramids:



SCALE	INTEGRAL FEATURE EXTRACTION TIME (SECOND)	PEDESTRIAN DETECTION TIME (SECOND)
0	0.29	0.588
1	0	0.427
2	0	0.323
3	0	0.314
4	0	0.247
5	0	0.194
6	0	0.162
7	0	0.138
8	0	0.114
9	0	0.081
10	0	0.075
11	0	0.060
12	0	0.049
13	0	0.038
14	0	0.024
TOTAL	0.290	2.834

Comparing the program with the feature inferring acceleration technique, we can achieve acceleration by 50%.

6. OUR METHOD AND PERFORMANCE EVALUATION

6.1. Introduction to Our Method

Previous chapters describe the features to be extracted for pedestrian detection, the object tracking method, post processing, and acceleration technique. To build a real-time platform for pedestrian detection, we set our pedestrian detection program to process 640×480 pixels using the most efficient settings (Maximize Speed(/O2)) + OpenMP+ Feature inferring techniques. The time for detecting pedestrians requires about 1.0 second; time for tracking objects requires 0.001 seconds. Since computing Local Binary Pattern features is not time efficient, we use Histogram of Oriented Gradient features as the primary classifier for finding pedestrians. When a spatial region is judged as containing pedestrian, it will extract Local Binary Pattern and Color histogram of Self-Similarity of low-level features for further classification. For real-time videos, program should process frames faster than 24 frames a second. We set classification step to detect pedestrians every 30 frames, the intermediate frames are computed by object tracking step.

Post-processing includes the three steps mentioned at Chapter 4. Geometric constraints help us eliminate the blocks which are unlikely to appear in certain regions on an image. Non-maximal suppression and repetition around pedestrians are two steps which help us delete the multiple rectangles around pedestrians.

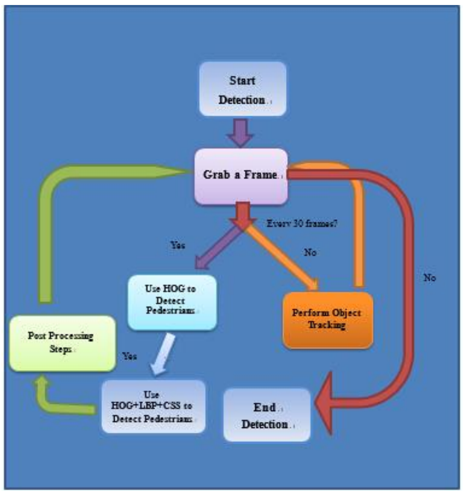


Fig. 17 Flowchart of our method.

6.2 Performance Evaluation

The performance of our program compares the result of using HOG and HOG+LBP+CSS features with the detecting result using CENTRIST features, we choose the test sets(set06-set10) of Caltech pedestrian dataset as the target dataset for evaluation, the final result are given below:

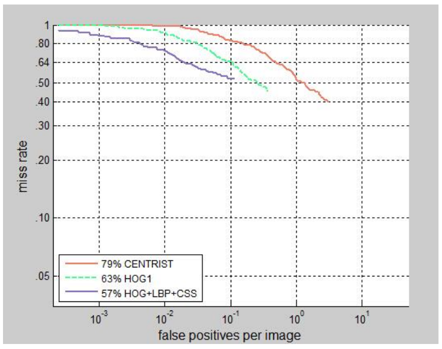


Fig. 18 Comparison of different methods

6.3 Result of Our Method

6.3.1 Samples of Correctly Classified Images

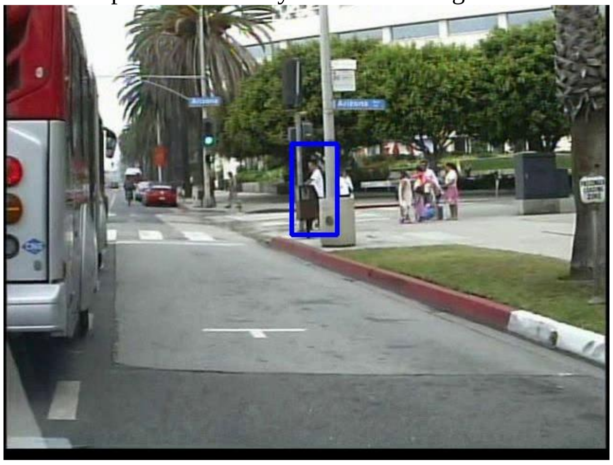


Fig. 19 Sample 1 of correctly classified pedestrians. (Other pedestrians are smaller than our minimum detection size: 96X48 pixels for image size of 480X640 pixels.)

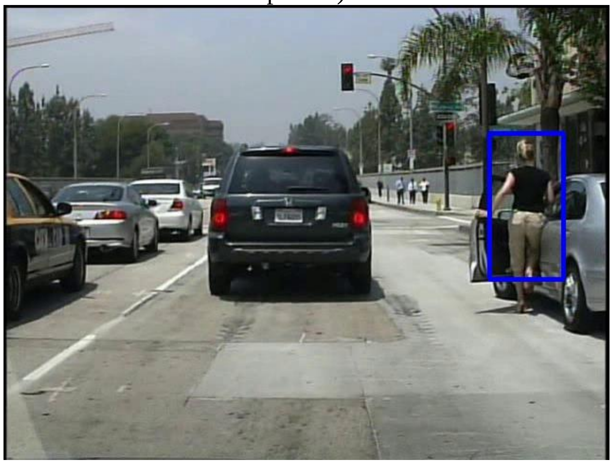


Fig. 20 Sample 2 of correctly classified pedestrians.

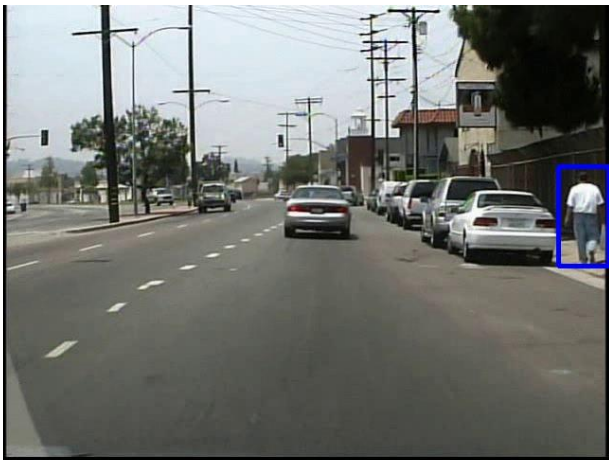


Fig. 21 Sample 3 of correctly classified pedestrians.

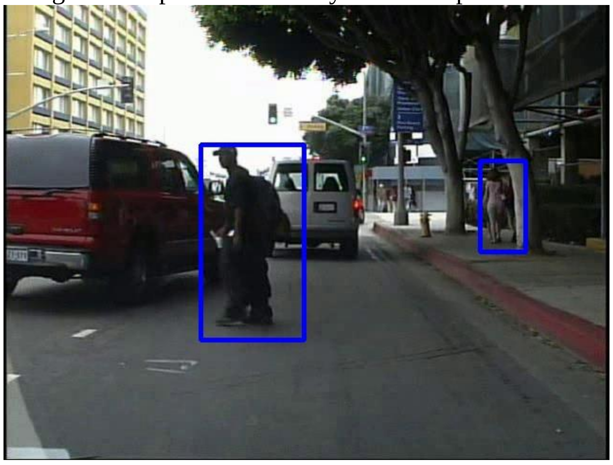


Fig. 22 Sample 4 of correctly classified pedestrians.

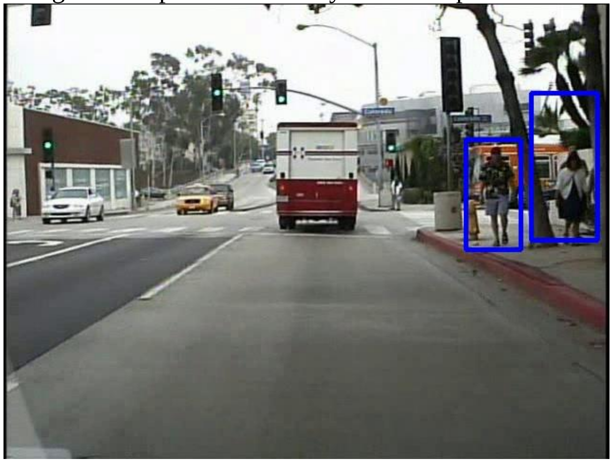


Fig. 23 Sample 5 of correctly classified pedestrians.

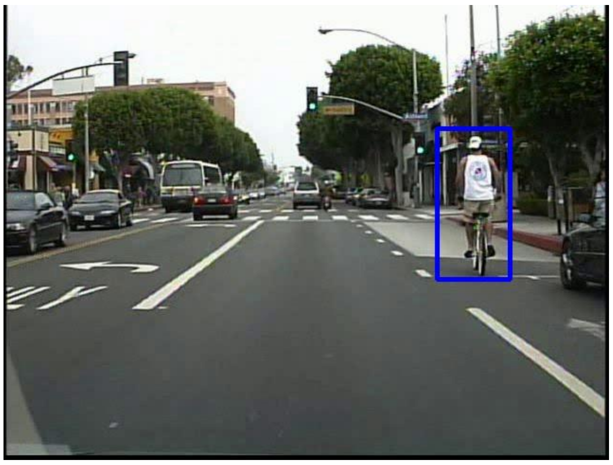


Fig. 24 Sample 6 of correctly classified pedestrians even on bicycle

6.3.2 Disturbance of Being Hidden by Other Objects

When we are tracking pedestrians, it is easy to encounter the problem that the pedestrians will be hidden by other objects (such as cars), then the detecting blocks will be distorted from their correct positions.

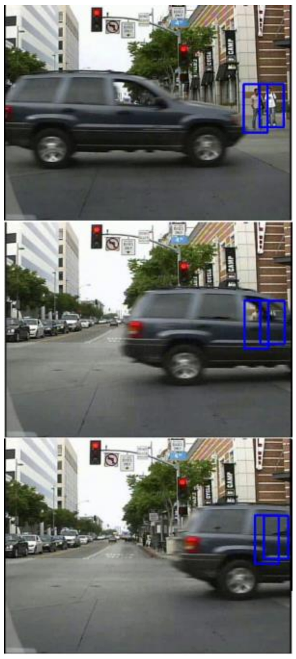


Fig. 25 Sample 1 of object occlusion and thus disturbance.

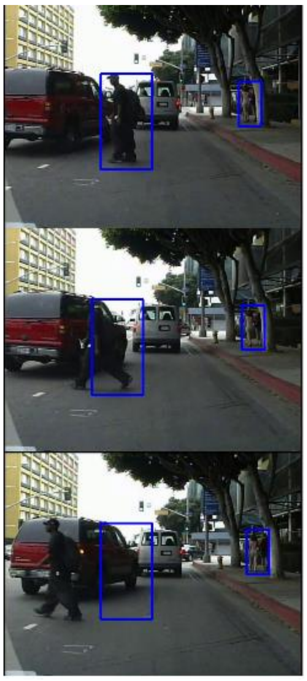


Fig. 26 Sample of wrong tracking caused by similar pedestrian and background object.

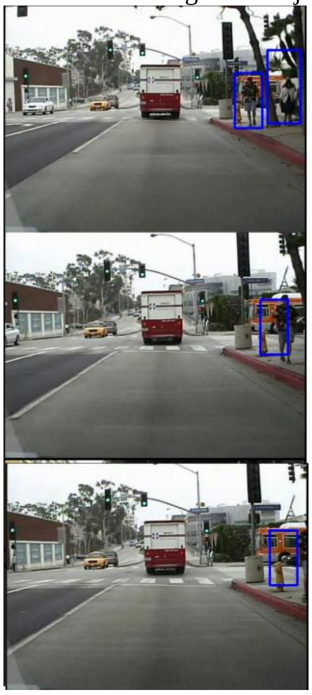


Fig. 27 Sample 3 of wrong tracking caused by approaching pedestrian and much larger block size.

6.3.3 Man-shape false positives

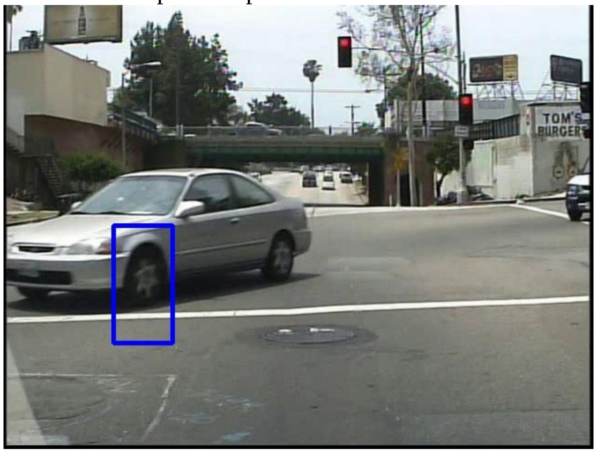


Fig. 28 False positive sample 1.

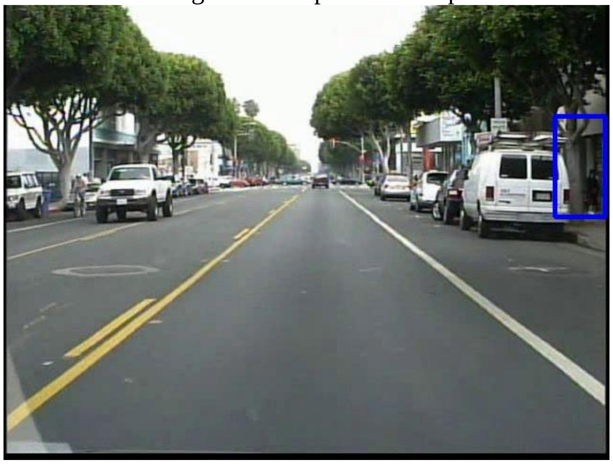


Fig. 29 False positive sample 2 due to tree similarity to pedestrian leg.

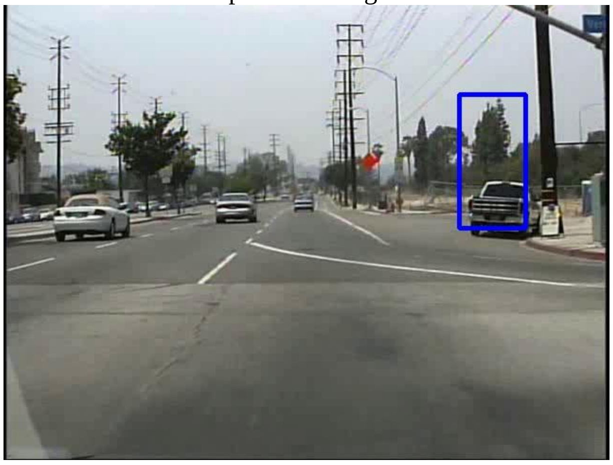


Fig. 30 False positive sample 3 (combined tree and truck similar to a pedestrian).

In our programs, there are also many false positives. Most of them are similar to pedestrians so that they are extracted by our program, such as tires, trees, and electric pole (similar to pedestrian leg).

ACKNOWLEDGEMENT

This research was supported by the Ministry of Science and Technology of Taiwan, R.O.C., under Grants NSC 101-2221-E-002 -194 and NSC 102-2221-

E-002 -177, and by Winstar Technology, Test Research, Qisda, Lumens Digital Optics, PolarLink, and Lite-on.

REFERENCES

[1] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, CA, Vol. 1, pp. 886-893, 2005.

[2] N. Dalal, B. Triggs, and C. Schmid, "Human Detection Using Oriented Histograms of Flow and Appearance," *Proceedings of European Conference on Computer Vision*, Berlin, Germany, Vol. 2, pp. 428-441, 2006.

[3] Denso, "Denso Global," http://www.globaldenso.com/en/, 2014.

[4] P. Dollar, "Caltech Pedestrian Detection Benchmark," http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/, 2014.

[5] P. Dollar, S. Belongie, and P. Perona, "The Fastest Pedestrian Detector in the West," *British Machine Vision Conference*, Ceredigion, UK pp.1-10, 2010.

[6] P. Dollar, Z. Tu, P. Perona, and S. Belongie, "Integral Channel Features," *Proceedings of British Machine Vision Conference*, London, UK, pp. 91.1-11, 2009.

[7] P. Dollar, C. Wojek, B. Schiele and P. Perona, "Pedestrian Detection: An Evaluation of the State of the Art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 43, No. 4, pp. 743-761, 2012.

[8] M. Enzweiler and D. M. Gavrila, "Monocular Pedestrian Detection: Survey and Experiments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 31, No. 12, pp. 2179 -2195, 2009.

[9] A. Ess, B. Leibe, and L. Van Gool, "Depth and Appearance for Mobile Scene Analysis," *Proceedings of IEEE Intl. Conference on Computer Vision*, Rio de Janeiro, Brazil, pp. 1-8, 2007.

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, Vol. 88, No. 2, pp. 303-338, 2010.

[11] G. D. Finlayson, B. Schiele, and J. L. Crowley, "Comprehensive Colour Image Normalization," *Proceedings of European Conference on Computer Vision*, Freiburg, Germany, pp. 475-490, 1998.

[12] B. Galvin, B. McCane, K. Novins, D. Mason, and S. Mills. "Recovering Motion Fields: An Evaluation of Eight Optical Flow Algorithms," *Proceedings of the British Machine Vision Conference*, Southampton, England, pp. 195-204, 1998.

[13] K. P. Horn and G. Schunck, "Determining Optical Flow," *Artificial Intelligence*, Vol. 17, pp.185-293, 1981.

[14] Y. Hsiao, "Pedestrian Detection Based on CENTRIST Descriptor and Stochastic Process and Implementation," Master Thesis, Department of Computer Science and Information Engineering, National Taiwan University, 2013.

[15] E. Y. Lam "Combining Gray World and Retinex Theory for Automatic White Balance in Digital Photography," *Proceedings of the International Symposium on Consumer Electronics*, Macau, pp. 134-139, 2005.

[16] B. D. Lucas and T. Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision," *Proceedings of International Joint Conference on Artificial Intelligence*, Vancouver, Canada, pp. 674-679, 1981.

[17] A. Mohan, C. Papageorgiou, and T. Poggio. "Example-Based Object Detection in Images by Components," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, No. 4, pp. 349-361, 2001.

[18] T. Ojala, M. Pietikäinen, and D. Harwood, "Performance Evaluation of Texture Measures with Classification Based on Kullback Discrimination of Distributions," *Proceedings of IAPR International Conference on Pattern Recognition*, Jerusalem, Israel, Vol. 1, pp. 582 -585. 1994.

[19] C. P. Papageorgiou, M. Oren, and T. Poggio. "A General Framework for Object Detection," *Proceedings of International Conference on Computer Vision*, Bombay, India, pp. 555-562, 1998.

[20] C. P. Papageorgiou and T. Poggio, "A Trainable System for Object Detection," *Internatioal Journal of Computer Vision*, Vol. 38, No. 1, pp. 15-33, 2000.

[21] K. C. Peng, "Pedestrian Detection and Range Estimation Based on CENTRIST Descriptor and Implementation," Master Thesis, Department of Computer Science and Information Engineering, National Taiwan University, 2011.

[22] M. Proesmans, L. Van Gool, E. Pauwels, and A. Oosterlinck. "Determination of Optical Flow and Its Discontinuities Using Non-Linear Diffusion," *Proceedings of European Conference on Computer Vision*, Stockholm, Sweden, Vol. 2, pp. 295-304, 1994.

[23] W. Schwartz, A. Kembhavi, D. Harwood, and L. Davis, "Human Detection Using Partial Least Squares Analysis," *Proceedings of International Conference on Computer Vision*, Kyoto, Japan, pp. 24-31, 2009.

[24] M. J. Swain and D. H. Ballard, "Color Indexing," *International Journal of Computer Vision*, Vol. 7, pp. 11-32, 1991.

[25] P. E. Trahanias and A. N. Venetsanopoulos, "Color Image Enhancement through 3-D Histogram Equalization," *Proceedings of IAPR International Conference on Image, Speech, and Signal Analysis*, The Hague, Netherland, Vol. 3, pp. 545-548, 1992.

[26] P. Viola and M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Kauai, Hawaii, Vol. 1, pp. 511-518, 2001.

[27] P. Viola, M. Jones, and D. Snow. "Detecting Pedestrians Using Patterns of Motion and Appearance," *Proceedings of the International Conference on Computer Vision*, Nice, France, Vol. 1, pp. 734-741, 2003.

[28] S. Walk, N. Majer, K. Schindler, and B. Schiele, "New Features and Insights for Pedestrian Detection," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, pp. 1030 -1037, 2010.

[29] X. Wang, T. X. Han, and S. Yan, "An HOG-LBP Human Detector with Partial Occlusion Handling," *Proceedings of the International Conference on Computer Vision*, Kyoto, Japan, pp. 32-39, 2009.

[30] Wikipedia, "OpenMP," http://en.wikipedia.org/wiki/OpenMP, 2014.

[31] C. Wojek, S. Walk, and B. Schiele, "Multi-Cue Onboard Pedestrian Detection," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Miami, Florida, pp. 794 -801, 2009.

[32] J. Wu, C. Geyer, and J. M. Rehg, "Real-time Human Detection Using Contour Cues," *Proceedings of IEEE International Conference on Robotics and Automation*, Shanghai, China, pp. 860-867, 2011.