Journal of Science and Technology ***49****, Number 5 (2011) 91-104*

NEW SYMMETRY PEAK PROCESSING AND GA-BASED SVM ALGORITHM FOR PEDESTRIAN DETECTION

Quoc Dinh Truong[1], Quoc Bao Truong[2]

[1]*College of Information&Communication Technology, Cantho University, Can Tho Vietnam*

[2]*School of Education, Cantho University, Cantho Vietnam*

Received September 30, 2011

ABSTRACT

In this paper, we consider the problem of pedestrian detection using a two-stage vision-based approach. The first stage is hypothesis generation (HG), in which potential pedestrian are hypothesized. We proposed a method called “colors difference” to determine the leg position of pedestrian. Then a new symmetry peaks processing is performed to define how many pedestrians are covered in one potential candidate region. The second stage is hypothesis verification (HV). In this stage, all hypotheses are verified by the combination between Decision Tree and a Modified Adaptive Genetic Algorithm to find the best features subset based on Haar-like feature and an appropriate parameters set of Support Vector Machine for classification. Our methods have been tested on different real road images and show very good performance.

Keywords. Hypothesis Generation (HG), Hypothesis Verification (HV), Different Color Method, Symmetry Peak Processing, Decision Tree (DT), Adaptive Genetic Algorithm (AGA), Support Vector Machine (SVM).

1. INTRODUCTION

Most of vision based pedestrian detection activities have been made by using the visible light vision sensors in the last years. According to the number of sensors in use, there exist two big sets of vision systems. There are the monocular vision system and stereo vision system. Many approaches on pedestrian detection have been presented with stereo vision systems because of the capability of the easy acquisition of depth information [1 - 3]. In contrast, few attempts have been made with monocular systems. In [4 - 5] they try to scan all image range with different scale of bounding box or find pedestrian which limit in ego lane or distant from moving vehicle to pedestrian is performed in some limit range based on the condition of road sense and camera calibration. This approach is quite simple but takes long time for processing and these systems not have ability to detect pedestrian in wide range distant or all possible pedestrian positions in image.

Vision-based pedestrian detection like other object detection problems can perform following two basic steps: hypothesis generation (HG) where the locations of possible pedestrians in an image are hypothesized, and hypothesis verification (HV), where tests are performed to verify the presence of pedestrians in an image [2] (Figure 1).

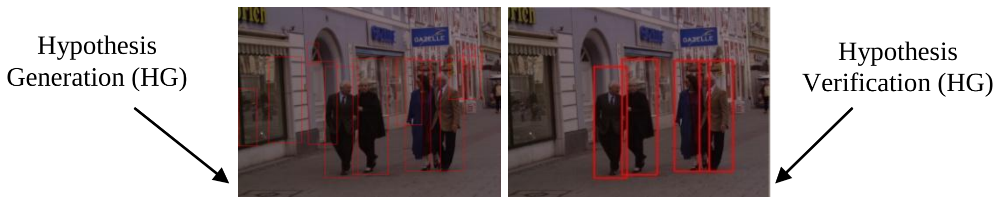

Figure 1. Illustration of two-step pedestrian detection strategy

Following the two-stage paradigm illustrated in Figure 1 above, we built a systematic approach to routinely detect *multi-view of pedestrians and we try to recognize all pedestrians that are appear in image as much as possible*. The remainder of this paper is organized as follows: Section 2 presents the hypothesis generation step with repair horizontal edges, colors difference method, and novel symmetry peaks processing in detail. Then, section 3 details the hypothesis verification with Haar-Like features, SVM classifier, decision tree, and modified adaptive GA algorithms. Section 4 describes some simulation results and some conclusions and discussions are shown in section 5.

2. HYPOTHESIS GENERATION STAGE

To hypothesize potential pedestrian locations in an image, prior knowledge about the appearance of the pedestrians may be used. By analyzing the image of pedestrian, he or she has very strong vertical edge symmetry structure and the position where determines border between the road and the leg of pedestrian shows many horizontal edges. Based on this, we propose an hypothesis generation algorithm is presented following sections.

2.1. Constructing edge maps or contour images

First, horizontal and vertical structures of interest could be identified by applying horizontal and vertical Sobel edge detectors to extract the gradient image of whole traffic scene. Then one threshold is performed to obtain the edge map or binary contour image by employing the experiment threshold. Next, we select the most promising horizontal and vertical structures by doing an analysis to get long vertical and horizontal edges from vertical and horizontal edges map (Figure 2).

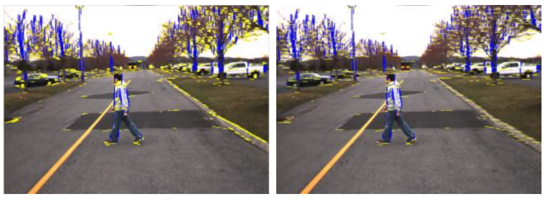

Figure 2. Long vertical and horizontal edges images (before and after repaired)

2.2. Repairing horizontal edges

In the pedestrian's foot region, there are many horizontal edges in addition and the horizontal edges are interrupted or gathered with each other. Therefore, it is hard to find the foot position of pedestrian.

To overcome the above problem and obtain bottom position of pedestrian, we follow a method that is presented in [6]**.** Here, we call *"Repairing long horizontal edges"* method. The neighborhood long horizontal edges in one region will be connected to each other into one long horizontal edge and the horizontal edges which are already used are deleted. The main idea is shown in Figure 3.

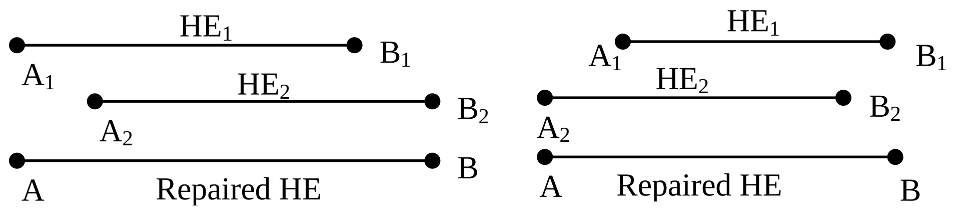

Figure 3. Principle of repairing horizontal edges

After applying the above procedure to repair the horizontal edges of the image, the number of long horizontal edges is reduced significantly (from 1791 to 491 in our tested case) and the sizes of the horizontal edges are increased (see right image in Figure 2).

2.3. Reducing the number of horizontal edges after repaired

To increase processing speed, we try to reduce useless horizontal edges as much as possible. Firstly, we remove too long horizontal edges using an experimental threshold. Then, based on the observing that there always exist vertical edges, which belong to shin of pedestrian, near the horizontal edges, we can remove the horizontal edges which do not have vertical edges near them. The number of long horizontal edges is reduced significantly (as shown in right image of Figure 4).

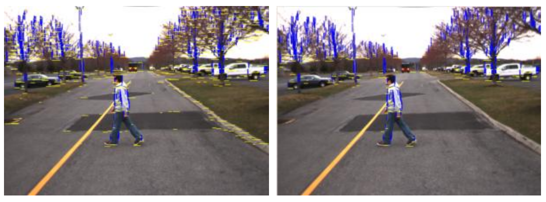

Figure 4. Result after repairing horizontal edges (left) and removing useless horizontal edges (right)

2.4. Determining pedestrian's leg position using "colors difference method"

Consider the pedestrian's images after removing useless horizontal edges. Then we can see that there are many horizontal edges (yellow edges) stay on the road or other objects (not belong to pedestrians). In addition, there are many horizontal edges belong to body of pedestrian. It is too hard to define where the leg position of pedestrian is. Observing pedestrian images and road scene, we can recognize that the color of the road and the pedestrian's clothes is usually different. It is also true with the road and other objects. Therefore, we propose a method that we call *"colors difference"* to remove useless horizontal edges one more time and indicate the leg position of pedestrian.

We calculate the average color of the road at the region whose height is from the bottom to one twelfth of image height. Considering the pedestrian images taken by a camera mounted on the vehicle, the specified region is covered by road background in almost cases. Then, we perform the following algorithm to remove horizontal edges staying in the road, other objects, and pedestrian's body. This algorithm will keep the horizontal edges that indicate the position of pedestrian's foot (Figure 5).

---

1. Construct two small regions based on remaining of repaired horizontal edges. One region is above and the other is below of considered horizontal edge.
2. Calculate average color for each region as same as in calculating average color of the road.
3. We applies an experimental threshold trial to remove these horizontal edges belong to the road, object and pedestrian body as much as possible. Try to keep the horizontal edges that indicate the foot position of pedestrian.

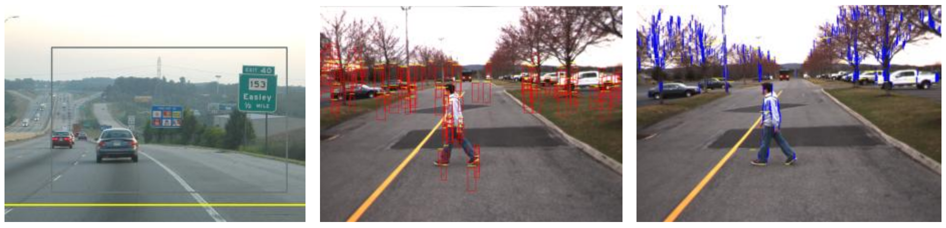

Figure 5. The region used to calculate average color of the road (left); Above and below regions used to calculate different average color (middle); Result of "colors difference method" (right)

2.5. Constructing regions of interest for performing symmetry peaks processing

After performing the steps mentioned above, if we build pedestrian candidate regions based on the remaining horizontal edges, in some cases, having regions that are covered two pedestrians (see Figure 6). Therefore, we have to perform peak processing of vertical symmetry histogram to construct real pedestrian candidate regions.

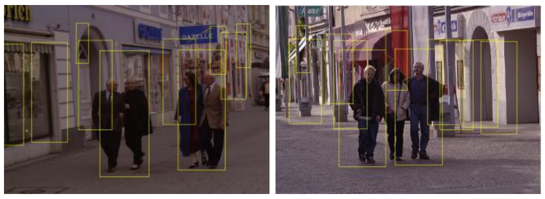

Figure 6. Two pedestrians are covered by one candidate regions.

The image of the pedestrian also satisfies a certain constraints in size and width-to-height ratio. The average width-to-height ratio is about 0.33 [7]. We use this ratio combination with the bottom position of remaining horizontal edges from previous step to construct one small ROI that cover the lower leg part of pedestrian and perform symmetry peak processing. This is based on the observation that the background of lower leg parts is, in general, homogeneously distributed ground plane. Contrarily, the background of upper body parts can be very complicated may lead wrong result in the symmetry search (left image of Figure 7).

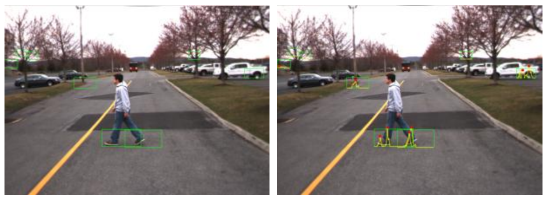
Figure 7. The ROI used for symmetry searching (left) and result of finding peaks (right)

2.6 Building vertical symmetry histogram of ROI and finding peaks of histogram

We applied following formula to calculate symmetry value for each column *k* at position *x* in ROI area is constructed above of vertical edge image:

$$If \begin{cases} |x - x_1| = |x - x_2| \\ y_1 = y_2 \\ I(x_1, y_1) = 255 \\ I(x_2, y_2) = 255 \\ d(I(x_1, y_1), I(x_2, y_2)) \le 0.2w \\ 0 \le y_1,\ y_2 \le h \end{cases} \quad then \quad S_k = S_k + 1 \tag{1}$$

where: S_k is vertical symmetry value of column k^{th}; w, h: width and height of ROI, respectively; $d(I(x_1, y_1), I(x_2, y_2))$: distance between two pixels $I(x_1, y_1), I(x_2, y_2)$.

We apply this process for all columns in ROI. As a result, we have an accumulator array of S_k, and the vertical symmetry histogram can be built using values of S_k. Then, we find peaks of vertical symmetry histogram and one illustrative result for our approach is illustrated in right image of Figure 7.

2.7 Determining the number of pedestrians covered by one ROI

After built vertical symmetry histogram of ROI, we apply following condition rules to determine how many pedestrians are covered by one ROI:

---

1. Consider the number of peaks of vertical symmetry peak processing:
   1.1. If one peak is obtained: the region is covered by one pedestrian.
   1.2. If three peaks are obtained: the region is covered by two pedestrians.

1.3. If two peaks are obtained: the region is covered by one or two pedestrians and go to step 2.

2. Consider the distance between two peaks, d :

   If $d \le \frac{1}{3} w$, then the region is covered by one pedestrian. (w is width of ROI); Otherwise, go to step 3.

3. Do the following steps if the ROI region is cover one or two pedestrians:

   3.1. Build an additional ROI at an upper position of current ROI. The new ROI is exactly same size with the current ROI.

   3.2. Find peaks of vertical symmetry of new ROI.

   3.3. If there is one peak is found, then the region is covered by one person. Otherwise, two persons are covered by that ROI.

Some illustrative result images of above algorithm are shown in Figure 8.

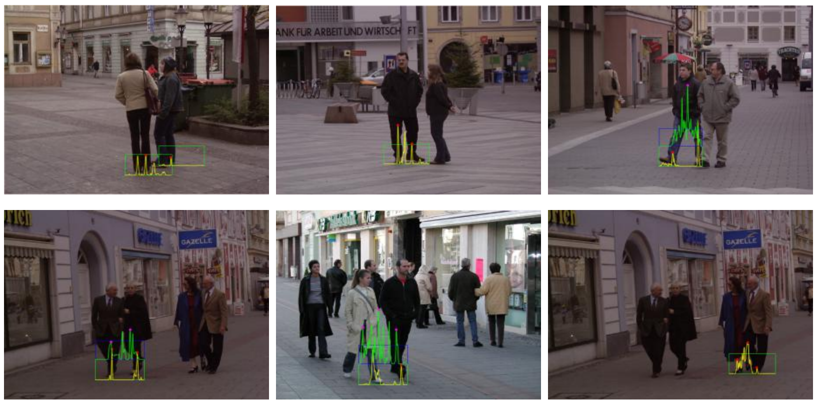

Figure 8. Illustration of many different cases of symmetry peak processing

2.8. Building pedestrian candidate regions

Based on the result of vertical symmetry peak processing we define the number of pedestrians that are covered by one ROI in section 2.7. Now, we will build candidate regions for pedestrian hypothesis generation purpose.

----

1. Determine left and right border of pedestrian to find candidate regions:

   1.1. If one peak is obtained, then the region is covered by one pedestrian. We will find general symmetry axis for that pedestrian; utilizing the region for building symmetry histogram to find the maximum peak in symmetry histogram (Pos), without considering the distance between 2 vertical edge pixels. Then left and right border of pedestrian region is calculated using equation (2):

$$LeftBorder = Pos - 0.35 * w; \quad RightBorder = Pos + 0.35 * w$$
where, w is width of ROI (2)

1.2. If two peaks are obtained and the distance between two peaks $d \le \frac{1}{3} w$, then the region is covered by one pedestrian. We examine:

1.2.1. If the magnitude of two peaks is very different we will use equation (2) to determine left and right border of candidate region with the value of parameter Pos here is the position of bigger peak.

1.2.2. Else, if the magnitude of two peaks is nearly same and the distance between two peaks is less than 20% of ROI size, w, then two peaks are stayed at left (P_{Left}) and right (P_{Right}) leg of pedestrian. We will use equation (3) to calculate value of Pos:

$$Pos = \frac{P_{Left} + P_{Right}}{2} \tag{3}$$

Next, the left and right borders of the candidate region are determined using equation (2).

1.3. If two peaks obtained and the distance between two peaks satisfies $d > \frac{1}{3} w$, then we consider:

1.3.1. If one pedestrian is covered by the region, P_{Left} and P_{Right} are the position of left and right peak, respectively. We test:

1.3.1.1. If the distance between P_{Left} and P_{Right} is less than $0.5w$, then the left and right border of the candidate region is:

$$LeftBorder = P_{Left} - 0.25 * w; \quad RightBorder = P_{Right} + 0.25 * w \tag{4}$$

1.3.1.2. Otherwise,

$$LeftBorder = P_{Left} - 0.1 * w; \quad RightBorder = P_{Right} + 0.1 * w \tag{5}$$

1.3.2. If two pedestrians is covered by the region:

1.3.2.1. For left region,

$$LeftBorder = Pos - 0.4 * w; \quad RightBorder = Pos \tag{6}$$

1.3.2.2. For right region,

$$LeftBorder = Pos; \quad RightBorder = Pos + 0.4 * w \tag{7}$$

1.4. If three peaks are obtained by vertical symmetry peak processing, then the region is covered by two pedestrians. We set the positions of three peaks as p_1, p_2 and p_3, respectively. We calculate left and right border of two regions for two pedestrians as follows:

1.4.1. For left region,

$$LeftBorder = P_1 - 0.4 * w; \quad RightBorder = P_2 + 0.1 * w \tag{8}$$

1.4.2. For right region,

$$LeftBorder = P_2 - 0.1 * w; \quad RightBorder = P_3 + 0.4 * w \tag{9}$$

2. For right region, the top border of pedestrian is computed based on the bottom position and width of pedestrian candidate region. Where:

$$W = RightBorder - LeftBorder \tag{10}$$

We call the bottom border of pedestrian $BottomBorder$ that is the bottom position of ROI that is constructed in section 2.7. The top border, $TopBorder$, can be calculated by:

$$TopBorder = BottomBorder - (W * Ratio) \qquad (11)$$

where $Ratio$ is the parameter that is determined based on width-to-height ratio of pedestrian in the image (see section 2.5).

Some illustrative images for the above algorithm are shown in Figure 9.

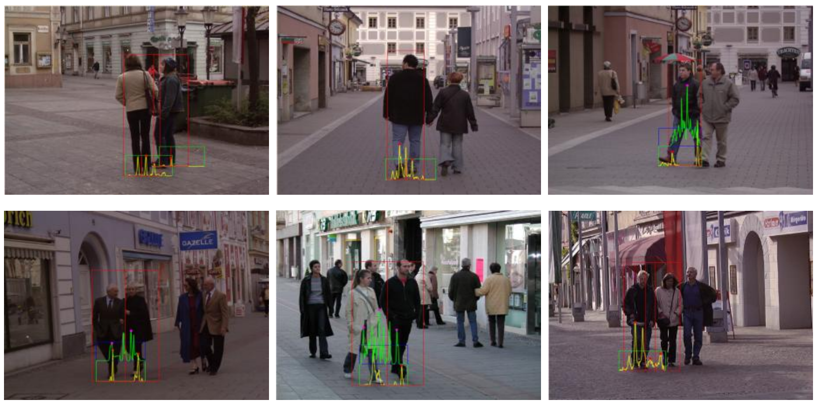

Figure 9. Illustrate many different cases of building pedestrian candidate regions

2.9. Removing these useless pedestrian candidate regions

Finally, after building pedestrian candidate regions using the method presented above, there are some regions overlapped too much to each other (see first row of Figure 10). To solve this problem, we search the area where two regions are overlapped to each other and check whether the following condition is satisfied:

$$\frac{The\ area\ of\ small\ candidate\ region}{The\ area\ of\ big\ candidate\ region} \geq 0.8 \qquad (12)$$

Then the small region will be removed. Results are illustrated in second row of Figure 10.

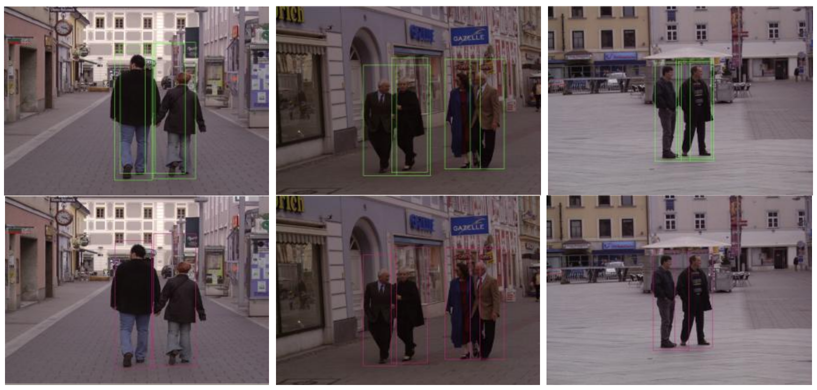

Figure 10. Results after removing useless regions.
First row: Overlapping candidate regions; Second row: Real candidate regions

3. HYPOTHESIS VERIFICATION STAGE

Feature selection is an important data pre-processing technique for training a SVM classifier. When the size of feature set becomes large, the detection speed decreases; when too few features are included in the set, however, the detection accuracy decreases. In our classification system, the size of original extracted Haar-like feature set is too large to be directly used for pedestrian detection. As a result, the Decision Tree algorithm can be used as simple measure to evaluate the classification ability of each feature, and a coarse feature set is obtained. Then we apply an evolutionary method which is presented in [8] to simultaneously optimize the feature set and the parameters of the SVM classifier.

3.1 Haar – Like Features and Support Vector machine (SVM)

As described in [9-10], each feature is represented by a template (shape of the feature) and its coordinate relative to the search window origin and the size of the feature (its scale). A subset of the features prototypes used is shown in Figure 11. Each feature is composed of two or three “black” and “white” rectangles joined together – these rectangles can be up-right or rotated by 45 degrees [10].

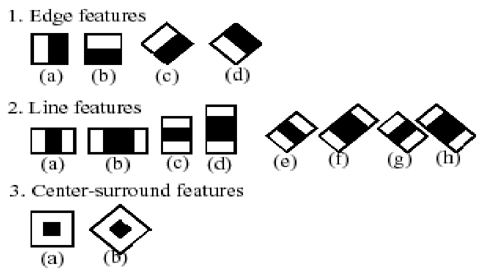

Figure 11. Subject of Haar-Like features prototypes used in our pedestrian detection

SVM is one of strongly classifier learning machines but its performance depend on the kernel parameter $K(x_i, x_j)$ is used [11]. In the literature, quite a few kernel functions have been investigated for various real-world applications. In most cases, radial basis function (RBF) kernel shows better performance than others kernel. Therefore, in our system we use Gaussian radial basis kernel to train classifier system.

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right), \gamma > 0 \tag{13}$$

In addition, the optimal feature set and parameters of SVM classifier are highly related to each other however, most existing classification system proposed in literature, the parameters are usually tuned in an experimental fashion and set separately, which cannot always guarantee an optimal model. The optimal values of the parameters may vary with different feature sets. Therefore, it is more desirable to develop them automatically and simultaneously.

3.2. Evolutionary Method

In our study, we build decision tree which has only one node to evaluate classification ability of one feature. The process is looped until all features are evaluated. Next, we pick out top N features that have highest classification ability. In our case, we choose **N = 298** top of features after decision tree training as the coarse feature set. The coarse feature set result will be input for a modified adaptive GA learning process to obtain the final detected result output.

3.2.1. Coding and encoding individual

In our GA scheme, each individual is designed to encode both the feature subset and the parameters in one individual. Feature subset is a vector of Haar-like features use to represent an image and the parameters are used to represent SVM training model.

$$\underbrace{0\,1\,0\,0\,\ldots\ldots\,1}_{P_1}\;\underbrace{1\,0\,1\,1\,0\,0\,1\,1\,0}_{P_2}\;\underbrace{0\,0\,1\,1\,1\,0\,1\,0\,1\,1\,0\,1}_{P_3}$$

Figure 12. Desmonstrate an individual using binary code scheme

As presented in Figure 12, an individual has three parts. For the left most P_1 has 298 bits use to represent Haar-like feature subset selection. Bit “1” means that the feature in corresponding position has been selected. The middle part P_2 has 9 bits and last part 12 bits of P_3 to represent value of γ and value of C for the Radial Basis Function (RBF) kernel of SVM training model, respectively. When C and γ are decoded, equation $\gamma = 0.01 * P_2$ and $C = 0.01 * P_3$ are used, where P_2, P_3 are two integers corresponding to two binary codes.

3.2.2. GA operators

In the simple GA (SGA), the cross-over probability P_C and mutation probability P_m, which are the key parameters affecting the performance of the GA should be determined through repeated experiments and it is difficult to find a suitable value for different problems. To solve this problem, an adaptive genetic algorithm (AGA) was proposed based on [12]. In the AGA, P_C and P_m can adjust themselves automatically as shown in the following equations:

$$p_c = \begin{cases} p_{C1} - k_1 \dfrac{f_{\max} - f'}{f_{\max} - f_{avg}}, & f' \geq f_{avg} \\ p_{C1}, & otherwise \end{cases} \tag{14}$$

$$p_m = \begin{cases} p_{m1} - k_2 \dfrac{f_{\max} - f}{f_{\max} - f_{avg}}, & f \geq f_{avg} \\ p_{m1}, & otherwise \end{cases} \tag{15}$$

where $f_{\max}$ is the maximum fitness value; f_{avg} is the average fitness value of the population f' is the larger fitness value of the two individuals which will undergo the cross-over operation; f is the fitness value of individual which will undergo the mutation operation; $0 < k_1, k_2 \leq 1$; P_{C1} is the default cross-over probability, and P_{m1} is the default mutation probability.

After the adjustments in equation (14) and equation (15), the cross-over probability and mutation probability of the individual with a fitness that equals the maximal fitness are separate, and the problems of poor astringency and prematurity can be overcome.

3.2.3. Fitness evaluation

We use the fitness function that is proposed in [8]. Three kind of estimates for classification performance can be obtained: error rate $Err(f(x))$ is defined as a probability for a sample not to be classified correctly; recall rate $\mathrm{Re}c(f(x))$ is defined as a probability for a sample with label $f(x)=1$ to be classified correctly; precision rate $\mathrm{Pr}ec(f(x))$ is defined as a probability for a sample with $f(x)=1$ to be indeed classified correctly, where $f(x)$ is the decision function of SVM training classifier. We define the evaluation function of SVM classifier as follows:

$$G(f(x)) = (1-\lambda)*(1-Err(f(x))) + \lambda*\mathrm{Re}c(f(x)); \qquad \lambda = 0.3 \tag{16}$$

The parameter $\lambda = [0.1, 0.5]$ is the trade-off between $Err(f(x))$ and $\mathrm{Re}c(f(x))$.

4. SIMULATION RESULTS

4.1. Dataset

Our pedestrian training database contains 6000 pedestrian images and 6000 non-pedestrian 16*40 pixels images. Some of these images are collected from the public downloadable NICTA pedestrian dataset [13], while the rest were taken by ourselves.

4.2. Performance test of classifier systems

We compare the performance of classifier system using evolutionary method with two approaches: (1) traditional GA and (2) Modified adaptive GA. The comparison of the performances is shown in Table 1:

Table 1. Comparison the performance of SVM classifier

Method	Number of features	Error rate	Recall rate	Precision rate
Traditional GA	153	0.02933	0.975902	0.965167
Modified AGA	145	0.008167	0.998648	0.98500

The comparison result shows that the modified GA can achieve a higher detection rate and lower false positive rate using only **145 features.** Modified AGA approach can benefit the final classifier from the two major advantages: the detection ability of the classifier is enhanced due to the omission of some noisy features; and the classification speed is accelerated, which is benefited from the reduction of feature size.

4.3. Simulation of pedestrian detection

The complete results were analyzed and simulated using a Visual C++ program in combination with an Open CV computer-vision tool and LIBSVM version 2.88 [14].To prove the strong performance of our method, several real road examples ranging from simple to complex were used for testing.

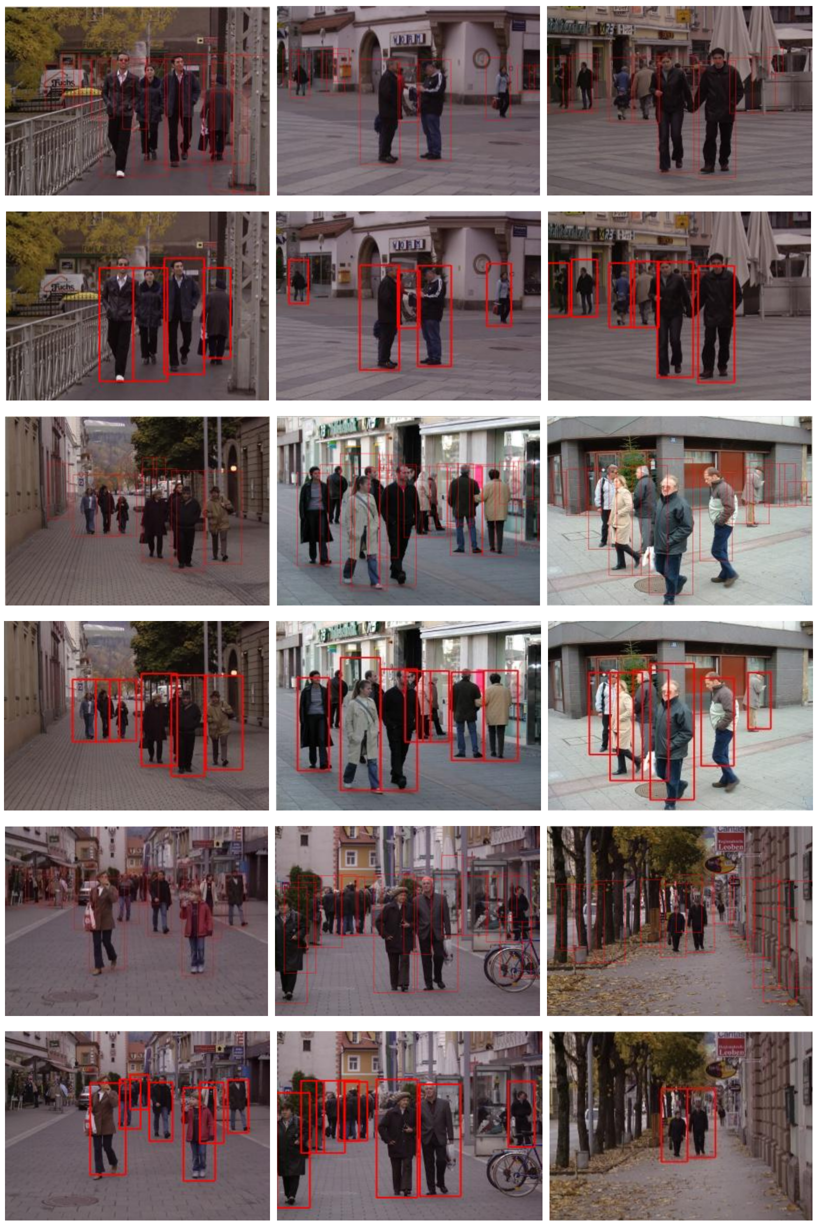

Figure 13. Result of pedestrian detection system test - HG stage (odd rows of images) and HV stage (even rows of images)

5. CONCLUSION

The strong contribution of our paper is that we introduce together new algorithms, presentation and insights which are quite generic about how to determine the position of objects such as vehicles or pedestrians in image using *"repairing horizontal edges"*, *"colorsdifference method"* and*"vertical symmetry peak processing"*. The vertical symmetry peak processing algorithm based on the edge contour images is robust and less sensitive with image's noise. The proposed approach based on some heuristics. However, we try to test them in many different real pedestrian images and it is actually robust inreducing sensitivity to noise and environmental conditions in the hypothesis generation step. This approach do not need to know information about the distance from the ego vehicle to pedestrians and camera calibration, and can be applied to generate candidate regions for different views of pedestrian.

Our pedestrian detection approach is quite different with many other approachesthat are proposed in the literature. Almost all systems that are presented in the literature just detect pedestrians crossing the street in the ego lane (the lane which is the ego vehicle is moving) or a few pedestrians in the short limit range in front of ego vehicle. It means that they just detect only two or three preceding pedestrians. So that the detection time is nearly short. On the other hand, in our case, we decide to detect all of pedestrians appearing on the image (at any lanes, any distance from ego vehicle). Therefore the detection time may take much time (especially the crowded scene with a lot of pedestrians moving on the road) when compared with other detection systems. To evaluate the average time of every frame in our method, we try to limit the range to detect pedestrians in image and just detect maximum 3 or 4 near pedestrians. Finally, we obtain the average time about **94 ms per frame**. The system is tested with standard PC machine (CPU Intel (R) Core (TM)2 Duo, 1.33GHz).

Acknowledgement. This publication was made possible through support provided by the IRD-DSF.

REFERENCES

1. Bertozzi M., Binelli E., Broggi A., Rose M. D. - Stereo Vision-based approaches for Pedestrian Detection, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, pp. 16-22.
2. Han F., Shan Y., Cekander R., Sawhney H. S., Kumar R. - A Two-stage approach to people and vehicle detection with HOG-Base SVM, In: The 2006 Performance Metrics for Intelligent Systems Workshop, 2006, pp.133-140.
3. Fernández D., Parra I., Sotelo M. A., Revenga P. A. - Bounding Box Accuracy in Pedestrian Detection for Intelligent Transportation Systems, In: IEEE Industrial Electronics IECON 2006, 2006, pp 3486-3491.
4. Xin L., Bin D., Hagen H. - Vision-based Real-time pedestrian detection for Autonomus Vehicle, In: IEEE Vehicular Electronics and Safety, 2007, pp. 1-5.
5. Yu L., Yao W., Liu H., Liu F. - A monocular Vision Based Pedestrian Detection System for Intelligent Vehicles, In: IEEE Intelligent Vehicles Symposium 2008, Eindhoven, the Netherlands, 2008, pp. 524-529.
6. Truong Q. B., Lee B. R. - Vehicle Detection Algorithm Using Hypothesis Generation and Verification, Huang D. S., Jo K. H., Lee H. H., Kang H. J., Bevilacqua V. (eEds.) LNCS, Vol .5754/2009, Springer-Verlag Berlin Heidelberg, 2009, pp. 534-543.
7. Lie G., Rong-ben W., Li-sheng J., Lin-hui L., Lu Y. - Algorithm Study for Pedestrian

Detection Based on Monocular Vision, In: IEEE Vehicular Electronics and Safety, 2006, pp. 83--87.

8. Cao X. B., Xu Y. W., Chen D., Qiao H. - Associated Evolution of Support Vector Machine-Based Classifier for Pedestrian Detection, J. Inf. Sci. **179** (2009) 1070-1077.
9. Viola P., Jones M. J. - Rapid Object Detection Using a Boosted Cascade of Simple Features, In: IEEEInternational Conference on Computer Vision and ParttemRecognition **1** (2001) I-511- I-518.
10. Lienhart R., Maydt J. - An extended set of Haar-like Features for Rapid Object Detection, In: IEEE International Conference on In Image Processing **1** (2002) I-900-I-903.
11. Gunn S. R. - Support vector machine for classification and regression.Technical report of Faculty of Engineering, Science and Mathematics School of Electronics and Computer Science, University of Southampton, USA, 1998.
12. Srinivas M., Patnaik L. M. - Adaptive probabilities of cross-over and mutation in Genetic Algorithm, IEEE Transactions OnSystems, Man and Cybernetics **24** (4) (1994) 656-667.
13. NICTA dataset, http://www.nicta.com.au/research/projects/AutoMap/computer_vision_datasets.
14. Hsu C. W., Chang C. C., Lin C. J. - LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Corresponding author:

Quoc Dinh Truong,

College of Information&Communication Technology,

Cantho University, Cantho Vietnam.

Email: *tqdinh@cit.ctu.edu.vn*