Noman, M., Yousaf, M.H., Velastin, S. A. (2016). An Optimized and Fast Scheme for Real-Time Human Detection Using Raspberry Pi. *In: 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*.

DOI: https://doi.org/10.1109/DICTA.2016.7797008

An Optimized and Fast Scheme for Real-time Human Detection using Raspberry Pi

Mubashir Noman
Department of Computer Engg.
University of Engg. & Tech.
Taxila, Pakistan
mubashir.noman87@gmail.com

Muhammad Haroon Yousaf
Department of Computer Engg.
University of Engg. & Tech.
Taxila, Pakistan
haroon.yousaf@uettaxila.edu.pk

Sergio A. Velastin
Department of Computer Science University Carlos III de Madrid Spain,
sergio.velastin@ieee.org

Abstract — Real-time human detection is a challenging task due to appearance variance, occlusion and rapidly changing content; therefore it requires efficient hardware and optimized software. This paper presents a real-time human detection scheme on a Raspberry Pi. An efficient algorithm for human detection is proposed by processing regions of interest (ROI) based upon foreground estimation. Different number of scales have been considered for computing Histogram of Oriented Gradients (HOG) features for the selected ROI. Support vector machine (SVM) is employed for classification of HOG feature vectors into detected and non-detected human regions. Detected human regions are further filtered by analyzing the area of overlapping regions. Considering the limited capabilities of Raspberry Pi, the proposed scheme is evaluated using six different testing schemes on Town Centre and CAVIAR datasets. Out of these six testing schemes, Single Window with two Scales (SW2S) processes 3 frames per second with acceptable less accuracy than the original HOG. The proposed algorithm is about 8 times faster than the original multi-scale HOG and recommended to be used for real-time human detection on a Raspberry Pi.

Keywords — Histogram of oriented gradients (HOG); Raspberry Pi; Town Centre dataset; CAVIAR dataset; Support Vector Machine; Human Detection

I. INTRODUCTION

Human detection plays a key role not only for surveillance and monitoring systems but it is also a fundamental step in pedestrian detection and people counting systems. Computer vision based human detection in surveillance systems provides assistance to government and private organizations for monitoring purposes and prevention of crimes. Similarly, automated pedestrian detection can help drivers to avoid collisions/accidents on the roads. Moreover, people counters would be useful in shopping malls to measure marketing statistics and to monitor high traffic areas. The above stated scenarios clearly indicate that human detection systems may assist us in everyday life in a variety of application scenarios.

Human detection is a challenging task due to a wide variety of poses and appearances. It becomes more difficult to detect humans in cluttered background. Also frequent occlusion between pedestrians or people makes it very challenging to detect them. These issues should be tackled properly otherwise they could result in a major damage in case of false results. Therefore, a robust visual feature selection is necessary that could discriminate humans cleanly from other objects in a dynamic background.

Human detection has been explored by a number of researchers in the literature. One of the prime scheme for multi-scale human detection was proposed by Dalal et al. [1] using Histogram of Oriented Gradients (HOG). To address the real-time need of human detection, Cao et al. [2] presented a real-time pedestrian detection system using HOG and FPGA. They used five angular bins for HOG and according to them, unevenly spaced five angular bins give low false positive rate. Chavan et al. [3] used HOG with some modification for real-time pedestrian detection using the TI C674x DSP embedded platform. A real-time pedestrian detection system based on edge factor and HOG was presented by Xu et al. [4]. Schwartz et al. [5] presented Partial Least Square (PLS) analysis based approach for human detection. They extracted features using original HOG, co-occurrence matrix and color frequency and then used PLS to reduce the dimensionality of the feature vector. Besides HOG some other pedestrian detection approaches include: Local Binary Patterns (LBP) based pedestrian detection [6], patterns of motion and appearance filters based pedestrian detection [7], background subtraction for extracting moving region and then using bounding ellipse/aspect ratio for human classification [8] have already been introduced. Similarly, part-based human detection approaches were also been explored and discussed in [9-11].

For real-time detection, the timely processing of the scene under consideration is of prime importance. Delegation of processing of visual data to central computing servers also affects the efficient use of human detection. To achieve this, many dedicated hardware i.e. Digital Signal Processors (DSPs), Field Programmable Gate Arrays (FPGAs) along with optimized software may be used to meet the real-time computational challenges. Software optimization is essential because poor algorithms could lead to system failure. That is more critical if the system uses a low processing hardware. Processing of significant information from visual data at the rate of 25-30 frames per second demands robust, fast and optimized human detection algorithm. Keeping in view existing research work, there is a need of an efficient

algorithm for human detection to process the visual data in real-time.

Deployment of smart solutions (DSP boards, Smart phones, Raspberry Pi etc.) in real world environment has also captured the attention of researchers and tech industry. Development of real-time human detection algorithm on Raspberry Pi is a challenging task because of its limited processing power. It has simple architecture, easy to understand Linux kernel based operating system thus a preferred choice for cost-effective solution. It has already been used for blob detection based people counting. This research work presents a robust and fast human detection algorithm for that device. As HOG features based human detection makes it more challenging because of large computations requirement, so research work is focused on an optimized and fast human detection algorithm considering foreground/background estimation as a prerequisite to different HOG based scenarios to avoid processing of un-interesting regions.

The rest of the paper is organized as follows. In the next section, the detailed overview of the proposed technique is presented, and section III explains the experimentation results of the proposed system on selected datasets. Finally, conclusions are summarized in section IV.

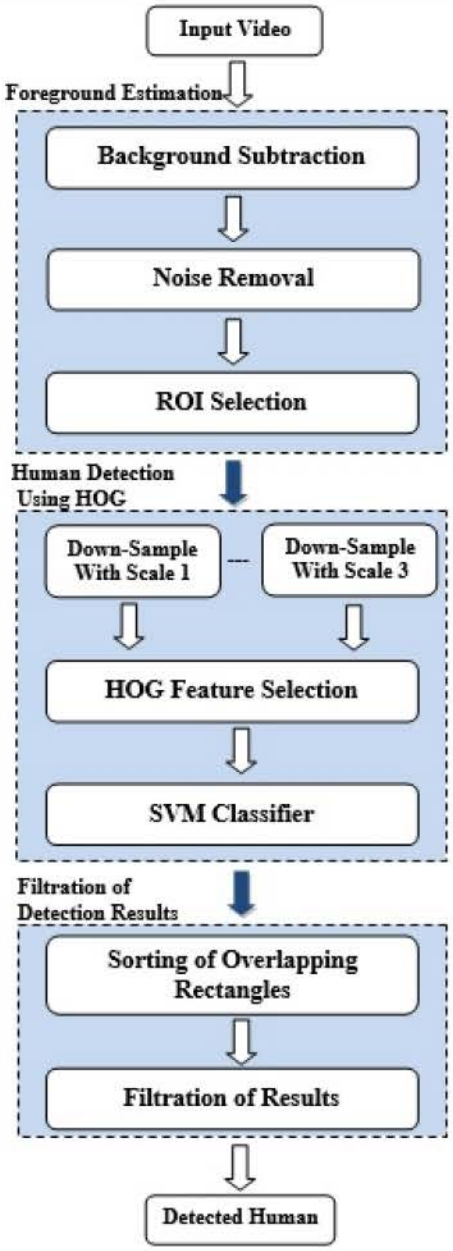
II. PROPOSED METHODOLOGY

Figure 1 shows the block diagram of the proposed system. Keeping in view the limitations of the Raspberry Pi, this is divided into three phases: *Foreground Estimation. Human Detection using HOG and Filtering of Detection Results*. Each phase is explained in the subsequent sub-sections.

A. *Foreground Estimation*

To process visual data of frame size of 640x480 @ 30fps for human detection on Raspberry Pi would become too expensive computationally. Therefore, before proposing any scheme for human detection, we have proposed a pre-requisite step of foreground estimation to extract the region of interest i.e. the areas that have probability of having some change that me be due to humans or other objects in the scene. After conversion of frames into gray scale, two background subtraction techniques have been proposed and evaluated to estimate potential foreground region i.e. frame difference technique and Gaussian mixture model (GMM).

The frame difference technique uses two consecutive frames from the video sequence and computes their difference. Then the absolute value of the difference image is processed for noise removal to reduce the effect of illumination changes. For this purpose, a global thresholding scheme is carried out. An appropriate threshold is required to get the best results. So, after experimentation it is determined that the intensity value 25 – 35 gives acceptable results. To get the region of interest, the coordinates of the foreground regions are found and resultant region is selected as region of interest to apply further procedure for human detection. An example of foreground estimation using frame difference technique is shown in Fig. 2.



Fig, 1. Block Diagram

The other foreground estimation technique i.e. Gaussian mixture model [12], is more robust and it can detect small changes easily but it requires more computations and more memory than frame differencing technique. In GMM background is approximated using following equation

$$p\left(x^{(t)}\middle|X_T, BG\right) \sim \sum_{m=1}^{B} \pi_m N(x; \mu_m, \sigma_m^2 I) \qquad (1)$$

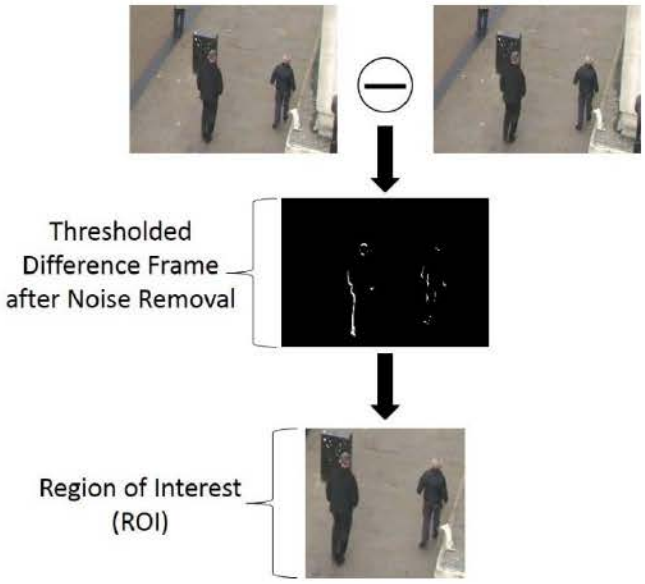


Fig. 2. Foreground Estimation/ ROI Selection using Frame Differencing

where X_T is training dataset, x is sample data, π_m is mth mixing weight, μ_m and σ_m^2 are the mean and variance of mth Gaussian component respectively. After background subtraction using GMM, noise removal is done as done for the frame difference technique and a foreground region is cropped using coordinates of foreground pixels. The sample result of extraction of region of interest using GMM is shown in the Figure 3.

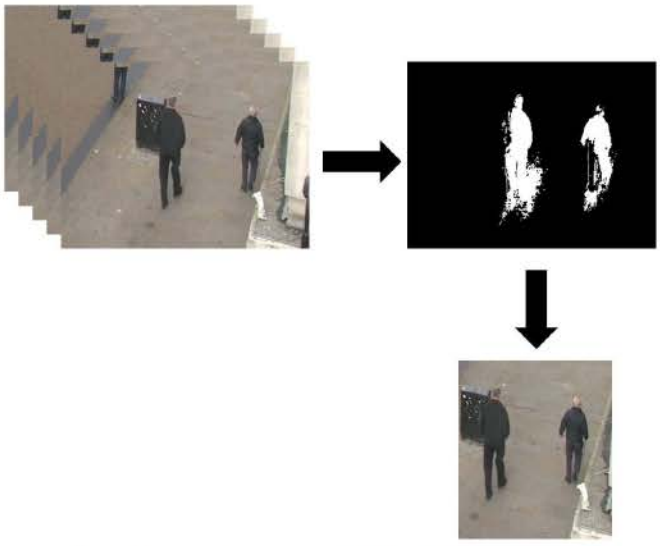


Fig. 3. Foreground Estimation / ROI Selection using GMM

B. *Human Detection using HOG*

Extraction of region of interest will be helpful in reducing the computational cost of the human detection algorithm. After extracting region of interest, we have employed HOG features on the ROI for multi-scale human detection. First step in HOG feature extraction is gradient computation using 1-D centered filter [-1 0 1] in both x and y direction. According to Dalal et. al [1], a centered 1-D filter has best performance. Magnitude and phase of the gradients are computed using

$$S = \sqrt{S_x + S_y} \text{ and } \theta = \arctan(S_y / S_x) \quad (2)$$

Orientation binning is done using nine orientation bins evenly spaced over 0º – 180º. The magnitude of each orientation is used as a weighted vote for that orientation. After orientation binning, blocks histogram are computed and normalized using L2-norm i.e. $\sqrt{||v||_2^2 + \epsilon^2}$. As mentioned in [1], block normalization is necessary because it reduces local illumination variances and increases performance of detector. HOG feature extraction is carried out using following parameters i.e. 8x8 cell size, 16x16 block size, unsigned evenly spaced 9 bins, rectangular HOG with 50% block overlap (Figure 4).

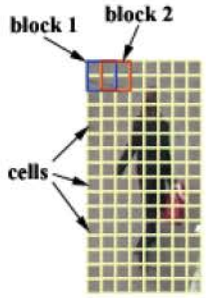


Fig. 4. Rectangular HOG using 50% overlap

After feature extraction, the next step is the classification of the computed features. For this purpose a linear support vector machine (SVM) is used. Computed descriptors are fed into a linear SVM (C=0.01) which is trained with using INRIA person detection dataset. Linear SVM separates the two classes by finding the optimal hyper-plane between them. The output of this part is whether the detector window contains or does not contain person.

To detect people in the whole image, the detector window is needed to be traversed over the whole image. Persons who are near to the camera will be greater in size as compared to the people who are far away. It might be the case that near person's size is greater than the detector window size. To detect persons who are bigger than the window size, the image is needed to be down sampled by a certain factor. This factor is known as scale stride (SS). If detection is done on an image with the original resolution or at a certain resolution and image is not resized to low resolution for further detection then it known as single scale detection. On the other hand, if an image is further down-sampled for detecting people then it is known as multi-scale detection. Dalal [1] uses multi-scale detection with scale stride of 1.05. Using this value for SS provides better detection performance. In multi-scale detection, first, the whole process i.e. feature extraction and

classification, as described above is done on images with original size WxH (width=W and height=H). Then the width and height of the down-sampled image are calculated as DW=W/SS and DH=H/SS (down-sampled width=DW and down-sampled height=DH) and image is resized to DWxDH. Figure 5 shows the detailed picture of multi-scale detection process. The scale stride is updated as SSx1.05. Then detection is performed on the resized image. Again the image is down-sampled and scale stride is updated. This process is continued until further down-sampling results in image size less than the window size. The detection results for each scale are then combined. For Raspberry Pi, using multi-scale detection with scale stride of 1.05 requires large computation time which is not desirable. So, a maximum of three scales are used and this will be a trade-off between detection accuracy and computation time. The optimum values of the three scales are chosen after different experiments.

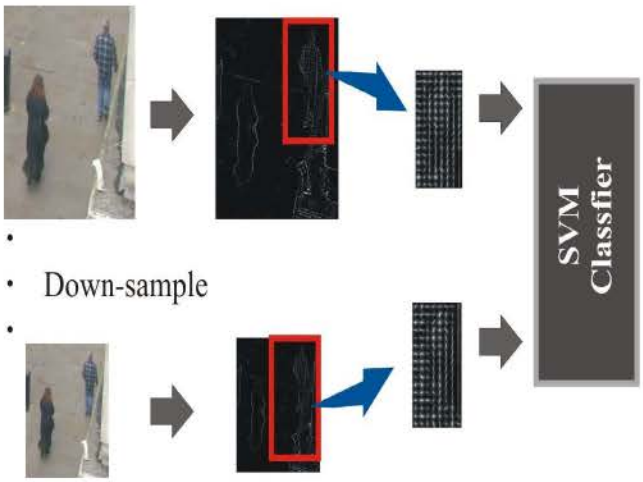


Fig.5. Multi-scale Detection

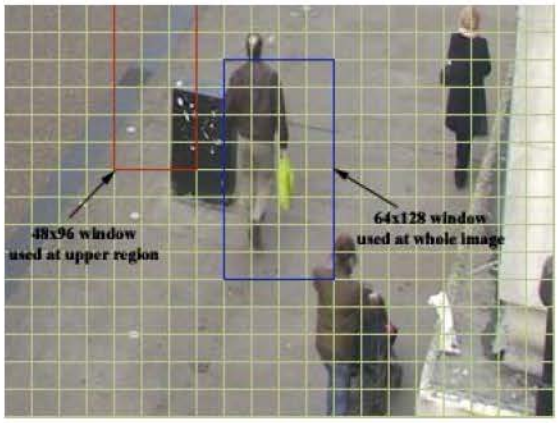


Fig.6. Human Detection using Two Detector Windows

Two schemes are used for detecting people using HOG. In the first scheme, only one detector window, i.e. 64x128, is used which is traversed on the whole image to find people. In the second approach, two detection windows are used: 64x128 window is used for the whole image and Daimler detector window of size 48x96 pixels is used at the upper part of the image. A smaller window is used in the upper portion of image because far persons appear in the upper region and we can detect those persons using smaller window size. Detector window with size 64x128 pixels is used at two scale levels and one scale is used with 48x96 detection window.

C. *Filtration of Detection Results*

The final step is to filter the detection results. In multi-scale detection a human might be detected more than one times at either one scale level or multiple scale levels. So, the detection results should be filtered so that there is only one rectangle over a person. Keeping in view the processing power of Raspberry Pi and real-time processing, we used a simple approach for filtration of overlapped detections. First, the area of the two bounding boxes is calculated. Then, the area of the intersected region is computed. If the intersected area of bounding boxes is greater than a threshold then one of them is deleted as follows.

$$(A_1 \cap A_2) > T, (A_2 \cap A_3) > T, \ \ldots . \ (A_{N-1} \cap A_N) > T \quad (3)$$

where A_1, A_2 … A_N are the areas of N overlapped rectangles and T is equal to 60 %. The value of T is determined after analysis of unfiltered results. The overlapping of rectangles is calculated for single and two persons, and it is concluded that overlapping of rectangles for single person is greater than 60 %.This process is done for all the overlapped rectangles. Final results of the detection are shown in Figure 7.

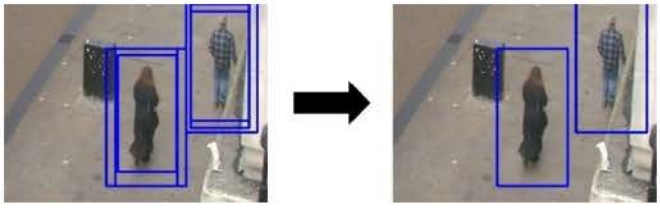


Fig.7.Filtration of Detection Results

III. Experimental Results

A Raspberry Pi 1 model B was used. It includes 700 MHz single core processor, 512 MB memory and Broadcom VideoCore IV @ 250 MHz GPU. The operating system used was Raspbian. All the experiments were done using OpenCV library version 2.4.10. For object detection HOG with default parameters was used i.e. 64x128 detection window, linear SVM, block size of 16x16 pixels, cell size of 8x8 pixels, 8x8 block stride and 9 orientation bins. The people detector used was trained on the INRIA dataset and uses linear SVM for classification. The following sub-sections will discuss the testing datasets, testing schemes and results.

A. *Testing Datasets*

Two datasets are used for testing of our proposed algorithm. First dataset we used is the Town Centre dataset that is publicly available at [15]. The dataset contains a video recorded at the busy town center street. It has frame rate of 25

frames per second and contains 7500 frames. The size of the video frame is 1920x1080 pixels. For testing on frame size of 320x240, all the frames of video dataset are needed to be resized. Resizing the video frame to 320x240 results in very small person size which cannot be detected either using 64x128 window size or even in most cases with 48x96 window size. So, the video frames are resized to 640x480 pixels. As the upper portion of resized frames still has small sized people, each resized frame is divided into four sub-frames of size 320x240. From these sub-frames, the sub-frames that contain small persons are discarded and the remaining ones are used for testing purpose.

The other dataset is CAVIAR dataset [16] that contains video clips recorded from different scenarios which include people walking, meeting with each other, entering and leaving shops, fighting and passing out, etc. It contains two video sets: first set is filmed for CAVIAR project at the entrance of INRIA labs and second set includes clips from shopping center in Portugal. The second set contains clips recorded at two different view i.e. corridor and front view. Each clip is recorded at 25 frames per second and has a frame size of 384x288 pixels. In our experiments, corridor view of second set is used and front view is discarded because front view video clips are recorded from far camera. Similarly, set one is recorded from top view and therefore it is also not used.

B. *Testing Schemes*

The following six testing schemes are employed for the evaluation of proposed algorithm on Raspberry Pi:

i. *Single Window with Three Scales (SW3S)*: SW3S uses a single detector window of size 64x128 for HOG features computation and three scales for multi-scale human detection.

ii. *Single Window with Two Scales (SW2S)*: SW2S uses two scales for multi-scale detection. The difference between first and second scheme is number of scales used for human detection.

iii. *Di Window with Three Scales (DW3S)*: DW3S uses a different strategy from the first two. It uses two detector windows (as in Figure 6): 64x128 detector window and 48x96 detector window. The small window (48x96) is used at the upper portion of the frame to detect far persons who are small in size whereas 64x128 window is used at whole frame to detect large persons as well as persons which cannot be detected by small window.

In the first three schemes, frame difference technique is used for foreground estimation followed by human detection using HOG and finally filtration of detection results. The remaining three schemes use Gaussian mixture model for background subtraction while rest of the procedure is the same.

iv. Single Window with Three Scales using GMM (SW3SMOG)

v. Single Window with Two Scales using GMM (SW2SMOG)

vi. Di Window with Three Scales using GMM (DW3SMOG)

The results of the original HOG, SW3S, SW2S and DW3S are shown in Fig. 8.

C. *Evaluation Methodology and Results*

To evaluate the performance of testing schemes for real-time implementation, two parameters are used: Accuracy and Processing Time. Processing time (PT) is how much time is required to process a single frame and is computed using

$$\mathrm{PT} = \frac{\text{Total Clock Ticks}}{\text{Tick Frequency}} \times 1000 \qquad (4)$$

Accuracy and processing time of the original HOG and six proposed testing schemes are shown in table 1 and table 2 for town centre and CAVIAR datasets respectively. Moreover, Precision and Recall rate are also computed for each scheme and are shown in table 1 and table 2 respectively.

First, the original HOG with multi-scale detection is tested on town center video dataset. After experimentation, it is found that the average processing time for a single frame is about 2544.4ms. This shows that 23.6 frames can be processed in one minute and processing 7500 frames requires 318 minutes.

TABLE I. RESULTS FOR TOWN CENTER DATASET

	Processing Time (ms)	Accuracy (%)	Recall (%)	Precision (%)
Original HOG	2544.4	73.23	66.65	99.75
SW3S	463.98	65.94	58.16	98.54
SW3SMOG	675.71	61.57	53.83	95.39
SW2S	333.52	62.31	53.35	98.83
SW2SMOG	500.93	58.48	48.74	98.02
DW3S	416.58	55.07	53.02	84.05
DW3SMOG	591.51	54.47	52.94	82.19

TABLE II. RESULTS FOR CAVIAR DATASET

	Processing Time (ms)	Accuracy (%)	Recall (%)	Precision (%)
Original HOG	2559.5	65.05	60.55	99.39
SW3S	435.63	61.33	61.06	92.86
SW3SMOG	776.02	55.34	53.02	94.47
SW2S	291.99	52.98	49.29	92.07
SW2SMOG	515.19	49.1	43.81	93.51
DW3S	464.44	31.7	55.96	39.79
DW3SMOG	612.47	32.72	56.14	41.27

As depicted from results in the two tables, SW2S gives the minimum average processing time i.e. 333.52ms and 291.99ms for Town Center and CAVIAR datasets respectively. The achieved accuracy of SW2S is 62.31% and 52.98% respectively. In case of SW3S the accuracy is about 66% and average processing time for single frame is 464 ms approx. for Town Center dataset. DW3S gives significantly less accuracy as compared to SW3S while processing time for this approach lies between the processing times of the SW3S and SW2S. The results of techniques that use GMM for motion detection show that processing time is increased considerably while accuracy is moderately decreased as compared to SW3S. This increase in computation time of GMM based techniques is due to complexity of GMM.

Detection Accuracy for the CAVIAR dataset is low compared to Town Center dataset because CAVIAR dataset contain small persons in the upper region of the frames. Moreover, false positive rate for CAVIAR dataset is high due to reflections and shadows. False positive rate of algorithms which use single window is very low as compared to the algorithms in which two windows are used. In the case of the Town Centre dataset, SW2S and SW3S have false alarm rates of about 2.5% and 3.5% respectively while DW3S has false positive rate of 37% approx.

IV. CONCLUSION

In this paper, different schemes are proposed and implemented on Raspberry Pi for human detection and their performance is compared with each other including original multi-scale HOG algorithm. It is evident that ROI and scale selection reduces the processing time of the single frame. Proposed testing scheme (SW2S) is pretty much optimized in terms of processing time and is about 7.5 times faster than the original multi-scale HOG algorithm. SW2S gives an average frame rate of about 2.9 fps. On the other hand, accuracy of the SW2S is less than the original HOG but still within acceptable limits. The time performance of proposed schemes on raspberry pi is much better than the original HOG and it can be used for the real-time people detection in variety of environments. Although, accuracy of proposed system is slightly low but it can be used in outdoor applications like pedestrian detector, and surveillance system. The Raspberry Pi based solution has advantages over other smart solutions depending on the problem. Given a limited number of fps on nominal resolution, such low-cost independent and portable solution can be employed. However, for visual data at higher fps and resolution, Raspberry Pi might not be a good choice.

As later version of raspberry pi has quad core processor than operate on 1.2 GHz, so processing time could be further reduced. Moreover, the Graphics Processing Unit of raspberry pi might be accessible in future, so it will be possible to improve the performance in terms of both accuracy and processing time.

REFERENCES

[1] Navneet Dalal, Bill Triggs, "Histogram of Oriented Gradients for Human Detection", in IEEE CVPR, 2005, pages 886 – 893

[2] Tam Phuong Cao, Guang Deng, David Mulligan, "Implementation of Real-Time Pedestrian Detection on FPGA", in IEEE IVCNZ, 2008, pages 1-6,

[3] Akshay Chavan, Senthil Kumar Yogamani, "Real-Time DSP Implementation of Pedestrian Detection Algorithm using HOG Features", in IEEE ITST, 2012, pages 352-355, doi:10.1109/ITST.2012.6425196

[4] Guoqing Xu, Xiaocui Wu, Zhengbin Wu, Li Liu, "Real-Time Pedestrian Detection based on Edge Factor and Histogram of Oriented Gradients", in IEEE ICIA, June 2011, pages 384-389, doi:10.1109/ICINFA.2011.5949022

[5] W. R. Schwartz, A. Kembhavi, D. Harwood, L. S. Davis, "Human Detection Using Partial Least Squares Analysis", in International Conference on Computer Vision (ICCV'2009), Kyoto, Japan, 2009

[6] Ahmed Boudissa, Joo Kooi Tan, Hyoungseop Kim, Seiji Ishikawa, "A Simple Pedestrian Detection using LBP based Patterns of Oriented Edges", in IEEE ICIP, 2012, pages 469-472, doi:10.1109/ICIP.2012.6466898

[7] Paul Viola, Michael J. Jones, Daniel Snow, "Detecting Pedestrians Using Patterns of Motion and Appearance", in IEEE ICCV, 2003, pages 734 – 741, doi:10.1109/ICCV.2003.1238422

[8] G. L. Foresti, C. Micheloni, C. Piciarelli, "Detecting Moving People in Video Streams", Pattern Recognition Letters, 2005, 26(14):2232-2243, doi:10.1016/j.patrec.2005.03.031

[9] Mykhaylo Andriluka, Stefan Roth, Bernt Schiele, "People Tracking by Detection and People Detection by Tracking", in IEEE CVPR, 2008, pages 1-8,

[10] Mykhaylo Andriluka, Stefan Roth, Bernt Schiele, "Pictorial Structures Revisited: People Detection and Articulated Pose Estimation", in IEEE CVPR, 2009, pages 1014-1021, doi:10.1109/CVPR.2009.5206754

[11] Antonio Prioletti, Andreas Møgelmose, Paolo Grisleri, Mohan Manubhai Trivedi, Alberto Broggi, Thomas B. Moeslund, "Part-Based Pedestrian Detection and Feature-Based Tracking for Driver Assistance: Real-Time, Robust Algorithms, and Evaluation", in IEEE Transactions on Intelligent Transportation Systems, 2013, pages 1346-1359, vol-14, no. 3,

[12] Zoran Zivkovic, "Improved Adaptive Gaussian Mixture Model for Background Subtraction", in Proc. ICPR, Pattern Recognition, 2004, vol. 2, pages 23-26

[13] Juan Li, Chunfu Shao, Wangtu Xu, Chunjiao Dong, "Real-Time Pedestrian Detection based on Improved Gaussian Mixture Model", in IEEE ICMTMA, Sixth International Conference on Measuring Technology and Mechatronics Automation, 2009, pp. 269-272

[14] OpenCV Documentation,"Object Detection" available online http://docs.opencv.org/2.4/modules/gpu/doc/object_detection.html

[15] Active Vision Laboratory, Department of Engineering Science, University of Oxford, "Stable Multi-Target Tracking in Real-Time Surveillance Video", available online at http://www.robots.ox.ac.uk/~lav/Papers/benfold_reid_cvpr2011/benfold_reid_cvpr2011.html

[16] University of Edinburg, "CAVIAR Test Case Scenarios", available online at http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/

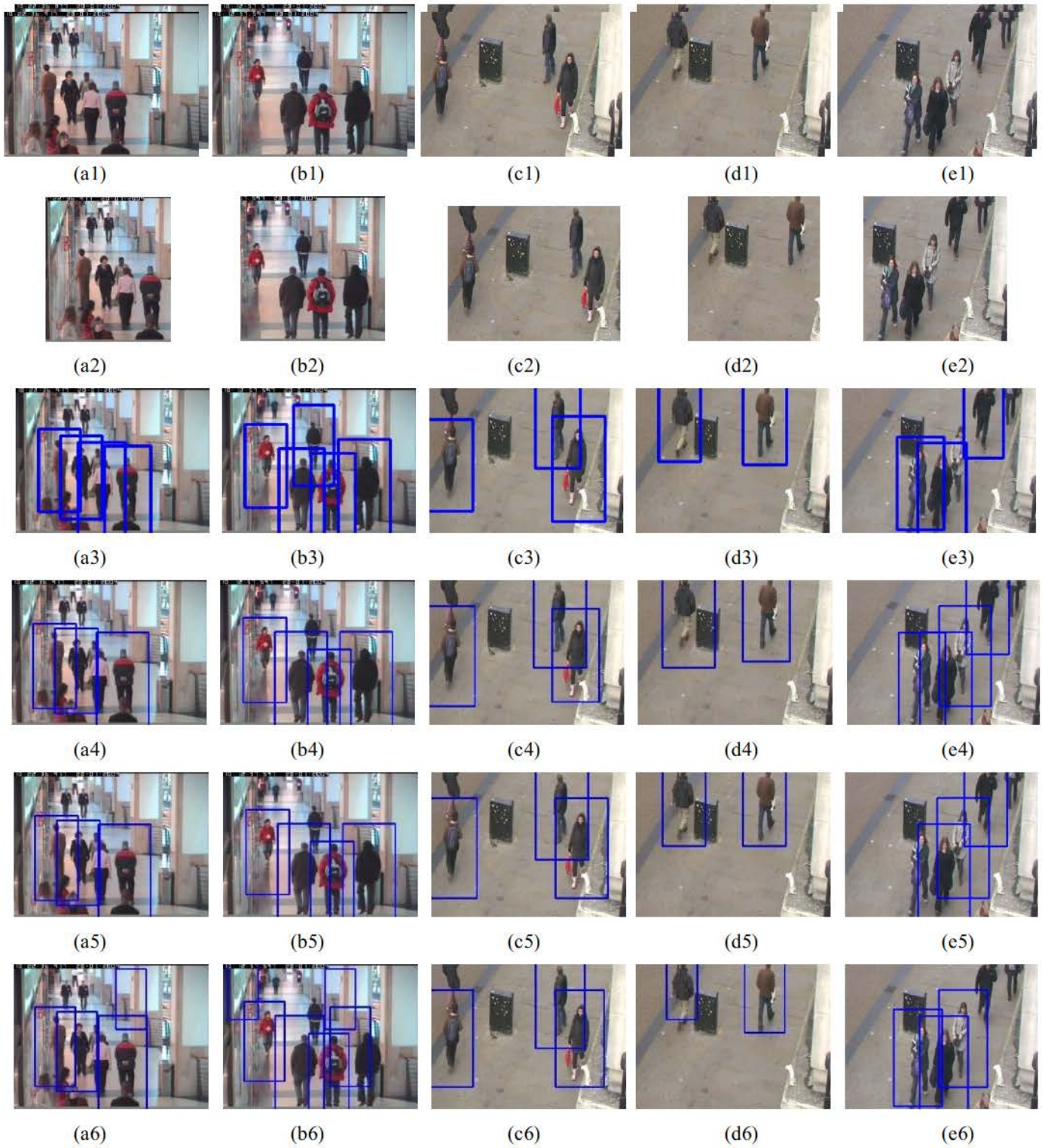


Fig. 8. Video Images (a1-e1), ROI (a2-e2), Results of original HOG (a3-d3), SW3S (a4-d4), SW2S (a5-d5), DW3S (a6-d6)