Image and Vision Computing 85 (2019) 1–13

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

A survey on human detection surveillance systems for Raspberry Pi☆

Ali Farouk Khalifa*, Eman Badr, Hesham N. Elmahdy

[a] Information Technology Department, Faculty of Computers and Information, Cairo University, Giza, Egypt

ARTICLE INFO

Article history:
Received 23 December 2018
Accepted 27 February 2019
Available online 26 March 2019

Keywords:
Human detection
Machine learning
Raspberry Pi

ABSTRACT

Building reliable surveillance systems is critical for security and safety. A core component of any surveillance system is the human detection model. With the recent advances in the hardware and embedded devices, it becomes possible to make a real-time human detection system with low cost. This paper surveys different systems and techniques that have been deployed on embedded devices such as Raspberry Pi. The characteristics of datasets, feature extraction techniques, and machine learning models are covered. A unified dataset is utilized to compare different systems with respect to accuracy and performance time. New enhancements are suggested, and future research directions are highlighted.

1. Introduction

Currently, security is an important issue to ensure overall safety of individuals. Surveillance systems have their increasing importance due to the increasing demand on security and safety. Surveillance systems are utilized in monitoring confidential areas where no one should pass around. These systems are used in critical places such as banks, military buildings, and governmental organizations and they can help in crime prevention, collision/accidents avoidance for drivers on the roads. People counting is another application where the results are used in statistical measurements for marketing, and monitoring areas with high traffic. Surveillance systems combine cameras and human factor. The camera sends a live streaming while a person monitors that stream. If there is a movement, it is mainly determined by the monitor person.

Traditional surveillance systems' main goal is to monitor the area under surveillance. However, they are passive systems. Passive means that they only monitor and can't take any further action based on what happens. There have been advancements in surveillance systems. One important advancement is the automation of surveillance systems through the use of internet. As a result, human factor is no longer needed. Object detection methods are introduced to replace the human factor. This will enable the system to detect the human movements. Human detection models are methods that enable the machine to identify a human in the captured images. These models are typically stored on internet servers or IoT platforms. A typical human detection system consists of two basic parts: motion detection and human classification. In order to classify if the moving object is human or not, two phases are performed. The first phase is feature extraction where hand crafted descriptors are calculated to identify objects. The second phase is model building where a classifier is utilized to discriminate this object of interest. Using deep learning will facilitate combining these two phases as it has the ability to learn features.

Although the system automation overcomes the human interaction limitation, internet connection proposes another limitation in surveillance systems. Internet connection is used to send frames to a server and receive the results back. If the connection is lost or the server is down, the whole system is down.

In order to tackle these limitations in surveillance systems, different systems have been developed that deploy a local human detection model and hence, it doesn't depend on internet connection. Different machine learning models are utilized in such systems. The models are uploaded on a microcontroller or chip which is attached to the camera to process the coming frames. This should solve both the internet connection and human interaction challenges.

In this paper, we review different surveillance systems for human detection. The focus is on systems that has been deployed on embedded devices such as Raspberry Pi. An online open source benchmark dataset has been used to evaluate the different techniques that have been utilized. This will give more insight into the analysis of different systems along with its associated performance.

This paper is organized as follows: Section 2 introduces the general framework of any human detection model. Section 3 covers the state-of-the-art techniques and methods used in a typical human

☆ This paper has been recommended for acceptance by Sinisa Todorovic.

* Corresponding author.
E-mail address: a.farouk@fci-cu.edu.eg (A. Farouk Khalifa).

https://doi.org/10.1016/j.imavis.2019.02.010
0262-8856/

detection model. Section 4 covers different human detection applications utilized on Raspberry Pi and the common datasets used in human detection models. Section 5 illustrate the results of comparing different techniques with different settings. Finally, Section 6 covers our conclusion and future work.

2. General framework

Surveillance systems are concerned of monitoring in general, but most importantly, monitoring human. Automated surveillance systems should be able to know that there is a person in the scene that is under surveillance. As a result, automated surveillance systems depend on human detection tasks.

Human detection system consists of sequential phases. As shown in Fig. 1, these phases are motion detection to extract regions that may contain human, and human detection to make a decision on the candidate extracted regions. Human detection is under the general term object detection. Any object detection system consists of two phases: feature extraction, and classification.

The first phase of the human detection system is to detect whether there is a moving object or not. Motion detection is the process of detecting any moving object in the scene under monitoring. There are many algorithms to detect motion in frames such as background subtraction [1], frame difference [2], and optical flow [1]. All of these methods can detect moving objects, but they differ in the performance time and the provided object information.

The next step is object detection. This step is concerned with detecting and classifying the object of interest. So, when motion is detected, the frames that contain the moving object are feed into the Object detection system. Typically, we need to identify unique features in our object in order to detect it in the image, otherwise it not the object we are searching for. Object detection phase is divided into two sub-phases: feature extraction, and machine learning model/classifier.

In Feature extraction phase, finding some characteristics of the object that can describe it well is the goal. These descriptors should be informative to discriminate this object from other objects. These descriptors are, hand crafted, mathematical operations performed on the input data to find its unique properties. There are many features developed as shown in Fig. 2. They vary from general-object features to specific-object features. For instance SIFT [3] and SURF [4] are point-based local features. Haar-like features consist of a set of templates that are compared to the regions of interest. Shapelet [5] is a set of mid-level features built from low-level features such local average of oriented gradients responses at each pixel. Edgelet [6] is part-based silhouette oriented features that consists of two steps: detection of parts and combining these parts. Shape-context [7] and local binary pattern (LBP) [8] are mainly appearance/texture-based features. There also local ternary pattern (LTP) which are edge based features such as histogram of oriented gradients (HOG) [9], C-HOG [10], R-HOG [10], HOG-III [7], chain code [11], moments, wavelets [11], and Gabor filter based context [11]. Features can be combined to get the benefits of both of their advantages such as HOG-LBP [12]. All these features can be used for object detection purposes, but HOG [9], and its modified versions such as HOG-III, V-HOG, or HOG-LBP are very effective with human detection tasks [13].

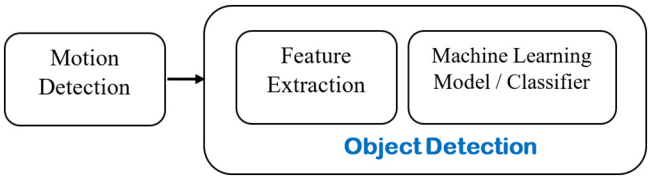

Fig. 1. General object detection system.

The extracted features are then fed into a model to classify the object. Last phase is the classification phase where a machine learning model is utilized. Machine learning model is a mathematical model used in the detection process. The model needs to learn the object that will be detected through a training phase. There are several machine learning models to use in this phase such as Bayesian classifier, AdaBoost [14], cascade classifier [13], and support vector machine (SVM) [14] with different kernels such as linear, polynomial, or radial basis function (RBF), neural networks (NN) [14], convolutional neural networks (CNN) [15], and recurrent neural networks (RNN) [15].

Neural network models with its variations (NN, CNN, and RNN) can be implemented with many layers which is called deep learning. In CNN, no features are required as it can learn the features as well. Consequently, the problem arises from the fact that features are hand crafted features and may not be suitable to represent some objects can be solved with CNN. The problem with deep learning is that it need high processing power, so typically, GPU is needed in the training phase.

3. Techniques

This section covers the different techniques used in every phase in a typical human detection system. We are focusing on techniques that has been used in surveillance systems for human detection and they are explained in details. Object detection tasks depend on two steps: object detection and object classification. The presence of object can be detected using motion detection methods. Classification decision is made on the detected object utilizing the features extracted of and the model used to classify.

3.1. Motion detection

Automated surveillance systems used to do processing on servers. This introduces networking issues that need to be considered such as security, and connection loss. With the increase in technological advancements in embedded devices such as Raspberry Pi, ODROID, and Dragon Board, it is now possible to upload machine learning systems on these devices. Embedded devices now are like computers of five years ago. So, the human detection system can be deployed on one of these devices such as Raspberry Pi. As mentioned in the previous section, motion detection is applied on Raspberry Pi. Motion detection approaches are background subtraction, frame difference, and optical flow.

3.1.1. Frame difference

Frame difference method is computed generally through calculating the difference between two consecutive frames. The general algorithm according to [2] is as follows: First frame is captured from static camera. Then a sequence of frames is captured at regular intervals. Absolute difference between consecutive frames is calculated. Difference image is stored. Difference image is then converted into gray then binary based on “threshold”. Morphological filtering “Opening” is applied to remove noise.

3.1.2. Background subtraction

Background subtraction method is based on estimating a background model that is then compared to a sequence of frames to find whether there is a motion in this sequence or not. The main step of background subtraction technique is the background modeling. It is to build a reference model that is used in recognizing moving objects. Background model is compared against video sequences. This comparison yields variations that determine the existence of motion in the scene.

Mean and median filters are the most currently used approaches in building the model. The difference between the current frame

and model is used to determine the motion. Background subtraction approach is simple, but sensitive to changes. There are mainly two approaches as described in [1]:

- Recursive algorithm
  Recursive techniques recursively update a single background model based on input frames instead of maintaining a buffer for the estimated background model. As a result, old frames will affect the current model. This method requires less storage compared to non-recursive methods, but errors in any frame will last long in the model.
  Various methods used in recursive algorithm such as approximate median, adaptive background, and mixture of Gaussians.
- Non-recursive algorithm
  A sliding window approach is used for background model estimation in non-recursive methods. A buffer stores previous L video frames. Temporal variations of each pixel in the buffer are used in the estimation process. Although non-recursive methods do not require old frames beyond L frames and they are highly adaptive, the storage may be significant when L is large.

3.1.3. Optical flow

In this method, image optical flow field is calculated, and clustering process is done based on the optical flow distribution characteristics of image. Complete object movement information can be obtained and the detection is better, however, this method requires large amount of calculations, it is very sensitive to noise, and has poor anti-noise performance, which make it not suitable for real time applications [1,16].

These methods differ in performance and shape information it can give. Optical flow is the best in giving shape information, background subtraction gives information less than optical flow, and frame difference doesn't give shape information at all. Frame difference is the fastest of them all and optical flow requires more computations. Fig. 3 compares between these methods.

Most systems uses either frame difference or background subtraction methods due to the computational time required is low to moderate. Systems in [17-20] uses frame difference approach to detect moving objects.

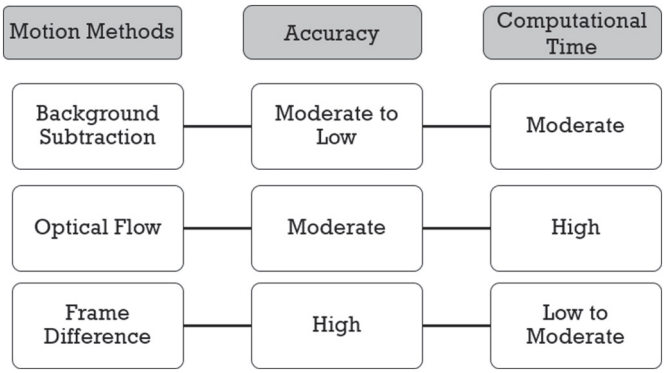

Fig. 3. Motion detection technique comparison.

3.2. Feature extraction

The next step in human detection system is feature extraction. This also can be applied on Raspberry Pi. As mentioned in the previous section, there are many features used. The most widely feature used with applications that deals with human detection is histogram of oriented gradients (HOG) that was introduced in [9]. HOG is invariant to scale, shadowing, contrast, and illumination changes. HOG is block-based features usually $128 * 64$ that computes a normalized histogram of orientations based on gradients. The general steps of HOG is to compute first order gradient at each pixel then binning the cells. Make histogram at each cell for the orientations. Normalize histogram along directions. Finally concatenate all normalized histograms to one feature vector.

Compute the gradient magnitude in horizontal and vertical directions, and orientation. Then for each window of size $128 * 64$ as shown in Fig. 4, it is divided into $16 * 16$ blocks with 50% overlapping. Each block is divided into $2 * 2$ cells of size $8 * 8$ pixels. Centralized filters used to compute horizontal and vertical gradients are as follows:

-1
0
1

-1	0	1

As can be seen in Fig. 4, each block consists of $2 * 2$ cells each of size $8 * 8$. At each cell, histogram of orientations is computed in 9

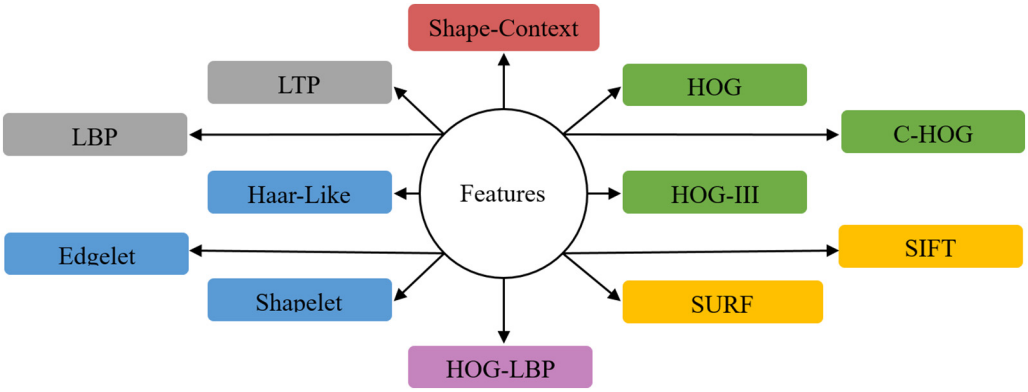

Fig. 2. Features used in object detection. Colors represent a set of related types of features.

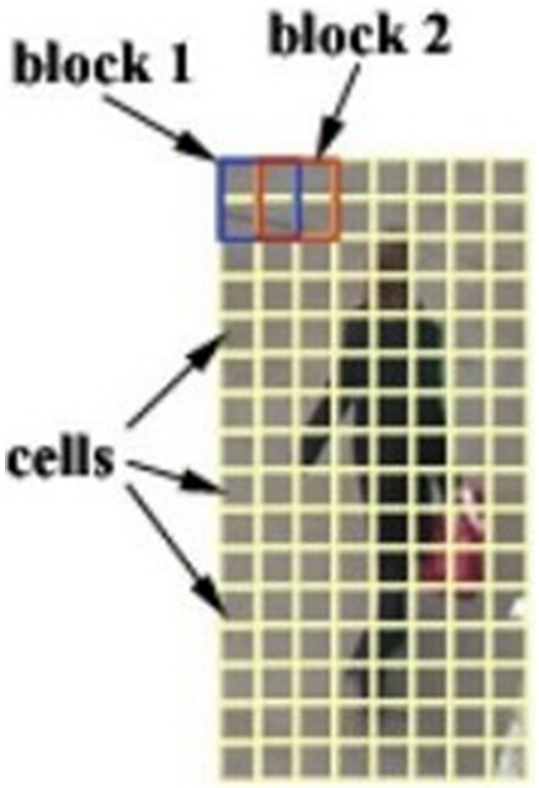

Fig. 4. HOG for window of size 128 * 64 [17].

directions. So, orientations need to be quantized to 9 directions/bins only between 0 and 180° this can be seen in Fig. 5. The value at each bin is the aggregation of magnitude values in the cells that their orientations belong to the same bin. The magnitude value of each pixel in the cell is partitioned between neighboring bins, example: If orientation of pixel is 85°, then distance to center bin 70 and bin 90 is 15 and 5 respectively. So, the ratios of the magnitude value for each bin are 5/20 = 1/4 and 15/20 = 3/4, respectively. Then the values in block are normalized. Finally, all values of all blocks are concatenated in 1D vector to form the feature vector. HOG visualization is represented in Fig. 6.

HOG is usually used with support vector machine (SVM), but it can be used with any machine learning classifier. HOG is used on Raspberry Pi in [21-24] in different applications, mainly for human detection. In [23], HOG is used along with shape similarity as the system uses thermal images and it can help finding humans through body temperature.

Local binary pattern (LBP) and local ternary pattern (LTP) features are based on a threshold. The feature is computed at every pixel. In LBP if a pixel of 8 connected neighbor pixels to the pixel of interest has a value less than the value of the pixel of interest, then a binary zero will be added to the resulting pattern. Otherwise, a binary one will be added to the resulting pattern. Fig. 7 shows an example of LBP. In LTP is the absolute difference between the values of pixel of interest and the one of the 8 connected neighbors is less than a threshold, then the pattern returned is 0. If the value of the pixel of 8 connected neighbors of the pixel of interest is greater than or equal to the value of the pixel of interest plus the threshold value, then 1 is returned. Otherwise, −1 is returned.

3.3. Machine learning model/classifier

After extracting the features, the model is introduced. Choosing the model is a vital step in any object detection system. The model learns the object features through training from dataset. In human detection systems the most used features is HOG and it is usually used along with support vector machine (SVM) [25] as they were firstly used in [9]. In [21-24] SVM is used in the detection process. In [24], SVM, Naïve Bayesian, and AdaBoost classifiers are used and SVM outperforms all of them in terms of accuracy.

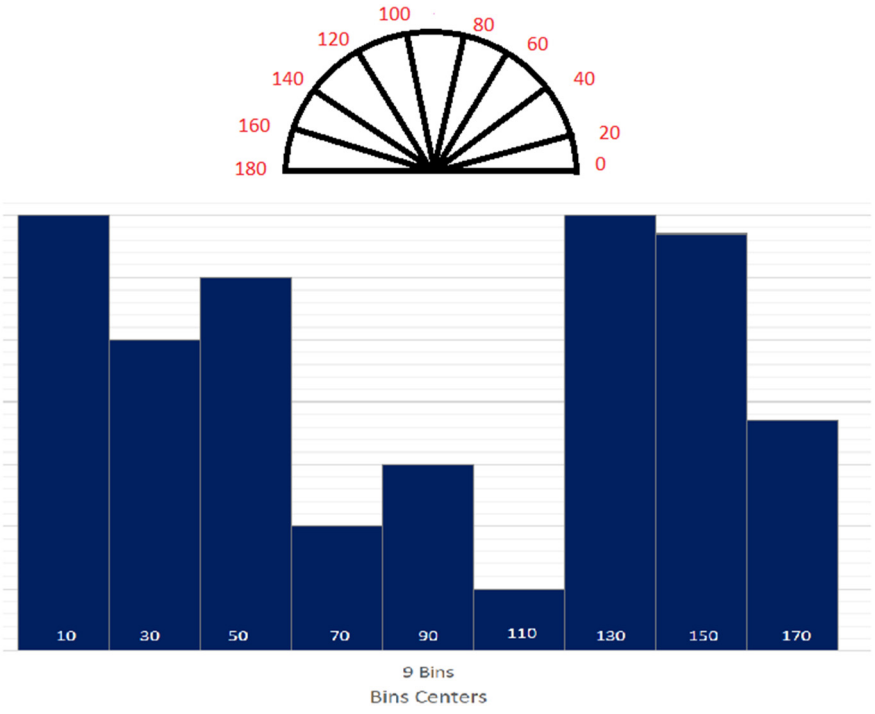

Fig. 5. Quantized orientations and bin histogram.

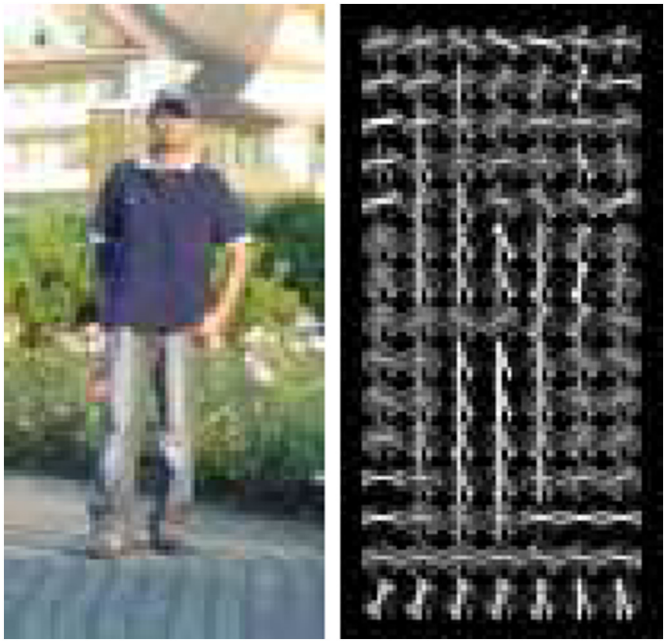

Fig. 6. HOG visualization [9].

According to [26-29], support vector machine algorithm is a binary classifier that is based on quadratic programming optimization. It is the enhancement of linear classification. SVM tries to find the optimal margins that reduces the errors, Fig. 8.

SVM advantage is that it can be used in nonlinearly separable data as SVM can transform nonlinear data to another domain where it can be linearly separable. This can be done through SVM kernel functions. There are different kernel functions such as linear, polynomial — with different degrees, and radial basis function (RBF). These kernel equations are illustrated in Eq. (1).

$$\begin{aligned} &\text{Polynomial kernel with d-degree:} \quad K(x,x^T) = (1+(x,x^T))^d, \\ &\text{Radial basis function (RBF):} \quad K(x,x^T) = \exp(-x-x^{T2}), \\ &(x,x^T)\text{: is the inner product of feature vector} \quad x. \end{aligned} \tag{1}$$

The kernel functions differ from accuracy and complexity. SVM gives good results in human detection problems. In such problems

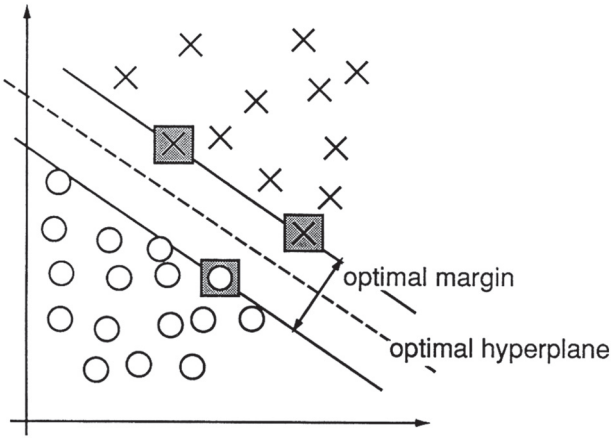

Fig. 8. SVM with margin [22].

SVM is the state of the art. Many variations and enhancement on SVM were introduced such as piece-wise linear SVM (PL-SVM) [30,10]. PL-SVM is a set of linear SVMs, each of them is supposed to discriminate human pose from other poses and non-human. Most kernel function used on embedded devices is the linear kernel as it is the simplest and gives acceptable accuracy.

4. Applications

This section covers different application and surveillance systems to detect human presence in different scenarios. These systems are utilized on embedded device, Raspberry Pi where different datasets were used in training and testing.

4.1. Datasets

There are many datasets for human detection systems such as MIT [31]. In MIT pedestrian detection dataset there are a set of 1800 color positive images and their mirror. There are 16,726 people-free images. All images are resized to 128 * 64. In images containing people, people are centered in the image. INRIA [9] contains 614 positive images and 1218 person-free images. There is also 1208 positive images and their mirrors of size 160 * 96 cropped on person. There are 288 test images of positive samples and 453 negative samples. There are 563 positive test samples of size 128 * 64. PennFudan [32],

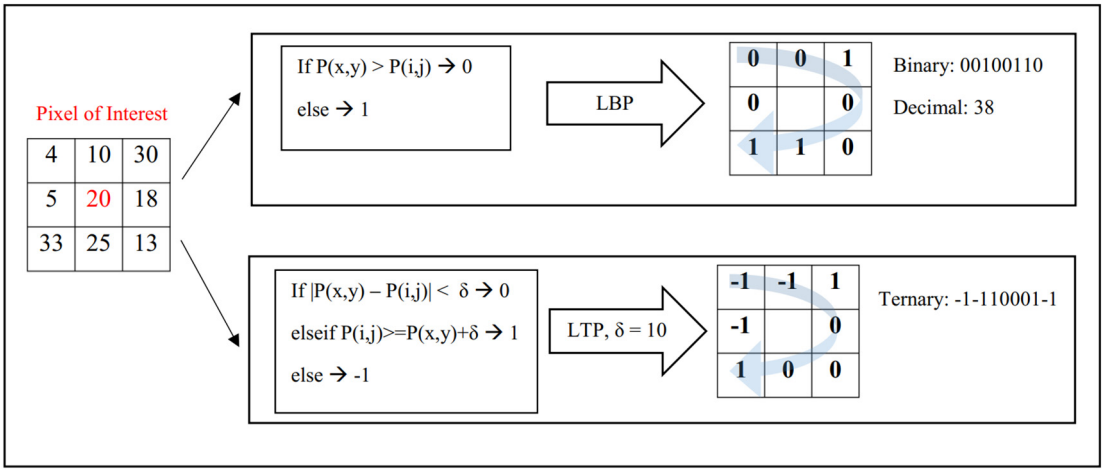

Fig. 7. LBP and LTP example.

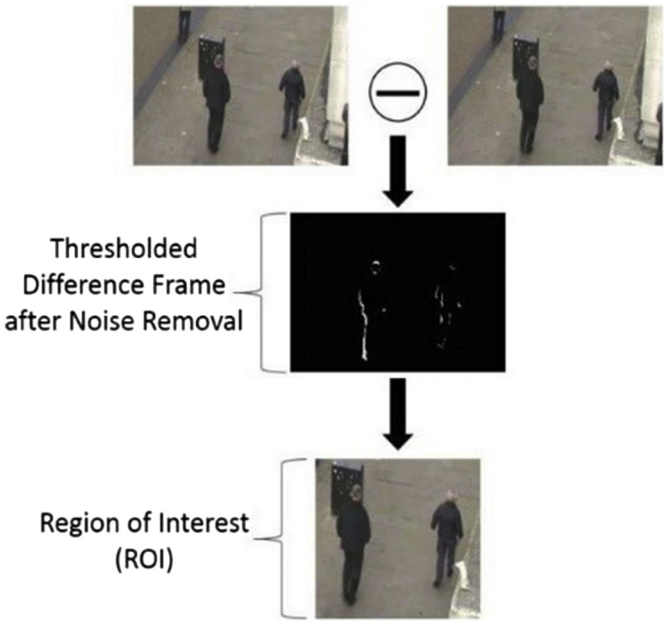

Fig. 9. Extracting region of interest using frame differencing in [17].

a pedestrian detection dataset that contains 170 images containing 345 labeled pedestrians of which 74 images are taken around Fudan University and the other 96 images are taken from Pennsylvania University. USC-A [33] contains 205 images from the internet of 313 humans without inter-human occlusion. Images contain of frontal, rear, walking, standing humans. USC-B [33], collected from CAVIAR [34] video campus of frontal, rear, walking, standing humans. It contains 54 images with 271 humans with partially inter-human occlusion. USC-C [35], collected from the internet with multi-view walking, standing humans. It contains 100 images with 232 humans without inter-human occlusion. PASCAL VOC [36], Voc challenge from 2005 to 2012. Each year challenge contains number of classes such as persons, cars, animals, and vehicles. Objects are labeled, for example: Pascal-Voc 2007 contains 9963 images of train, validation, and testing of 24,640 labeled objects. H3D [37] contains images of 1000 people with 3D skeleton and segmented regions. CAVIAR [34], number of different scenario clips of walking, meeting with others, window shopping, entering and exiting shops, fighting, passing out, leaving packages, etc. were recorded. There are two video sections. The first section is recorded with wide angle camera lens with $384 * 288$ pixels frames with 25 frames per second in INRIA labs. The second section is also recorded with wide angle lens camera with $384 * 288$ pixels with 25 frames per second. Caltech [38] consists of about 10 h of video taken in urban. The video is 30 frames per second each is of size $640 * 480$. There 250,000 frames were annotated with 2300 unique humans and 350,000 bounding boxes. Other datasets are TUD [39], CVC [40], DaimlerChrysler (DC) [41], and ETH [42]. All these datasets can be categorized to general purpose human detection, surveillance systems, and pedestrian detection [13].

4.2. Applications on Raspberry Pi

System in [21], utilizes SVM with HOG features on RPI 1 model B. INRIA dataset is used in training the SVM and CAVIAR and TOWNCENTER datasets are used in testing. HOG computations are very expensive, so, region of interest (ROI) is used to reduce the computations. region of interest (ROI) is obtained based on a pre-step of foreground estimation. Two methods of foreground estimation were experimented: frame difference and Gaussian mixture model (GMM). Frame difference method is illustrated above and it is used to detect the ROI as in Fig. 9 as follows:

The other method is Gaussian mixture model Fig. 10. Although it can detect small changes and it is robust, it requires more resources (computation and memory) than frame difference. HOG is then used to train SVM. Multiscale detection is used as people can appear in

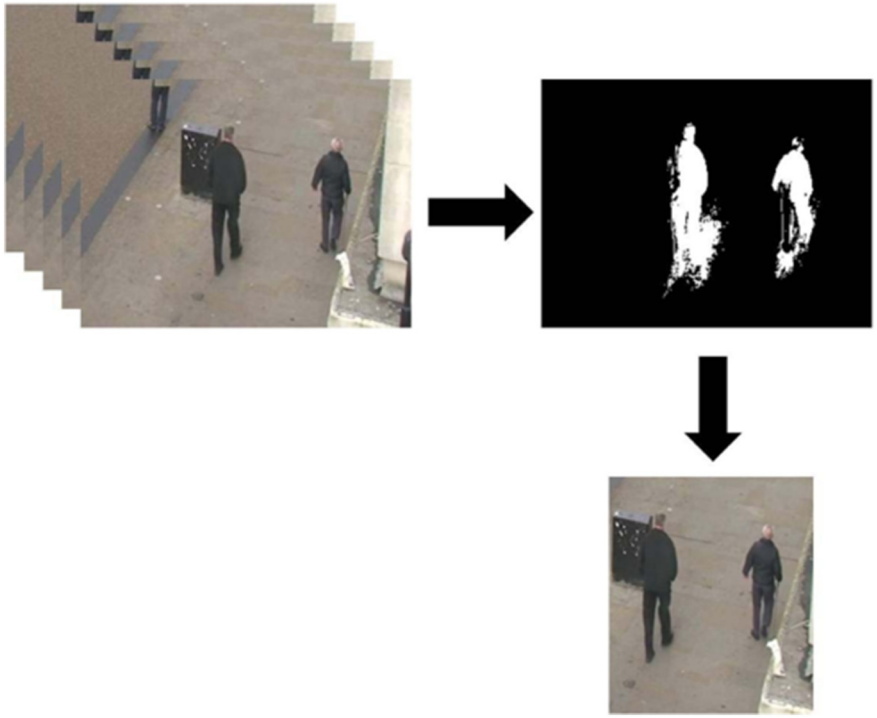

Fig. 10. Extracting region of interest using GMM in [17].

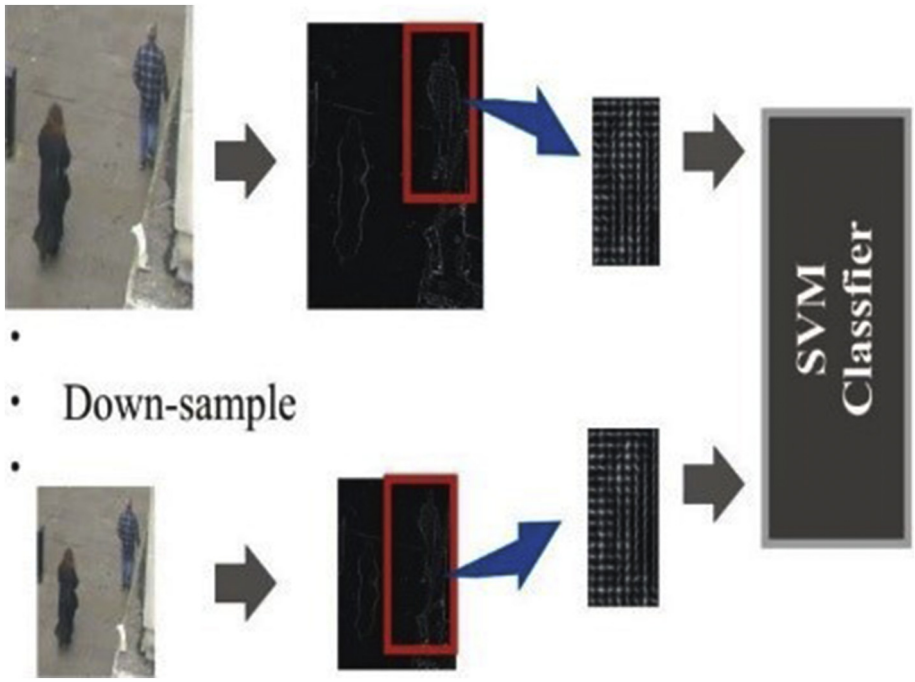

Fig. 11. Multi-scale detection in [17].

different sizes in the images. As a result, people who are near to the camera will appear in larger size than who are far away. So, to get over this problem, the image has to be down-sampled by scale factor called scale stride (SS), Fig. 11. The scale stride of 1.05 is used. The image should be down-sampled till the image size is less than the detection window. The scale in width and height is performed as in Eqs. (2)–(4) where DH and DW are the new height and width respectively, H and W are old height and old width, and SS is the stride. The authors only considered 3 scales as the computations of the scales are expensive on Raspberry Pi.

$$DH = H/SS \tag{2}$$

$$DW = W/SS \tag{3}$$

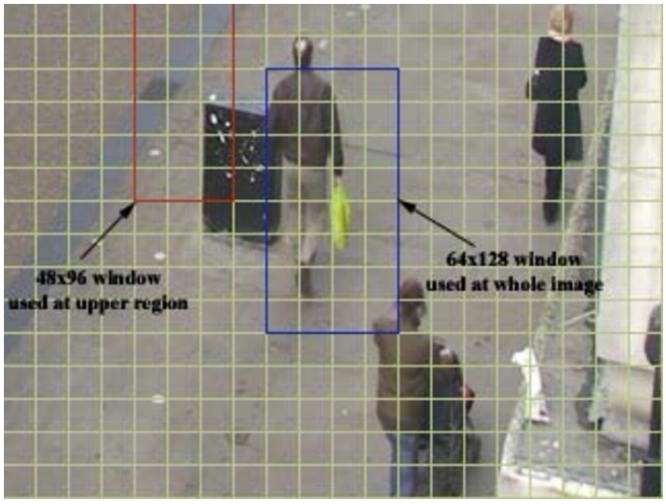

Fig. 12. Two detector windows in [17].

$$SS = SS * 1.05 \tag{4}$$

The authors used two approaches. The first one uses one window of size $64 * 128$ that traverses the whole image. The second one, as seen in Fig. 12, uses two windows: the first one is of size $64 * 128$ that traverses the whole image and the other one is of size $48 * 96$ and is used at the upper part of the image. The detection in image is the detection in all scales. So, a filtration process is done, Fig. 13.

Six testing schemes are employed. In these schemes single and double window sizes are utilized. Two or three scales are used with these different windows. All these combinations are tested with two different foreground estimation methods: frame difference and Gaussian mixture model (GMM). Example: single window with three scales (SW3S), and di window with three scales using GMM (DW3SMOG).

In Table 1, the first three approaches use frame difference foreground estimation method. The other three use the Gaussian mixture model (GMM). After foreground estimation HOG and filtration are done for human detection. Accuracy and processing time (PT), Eq. (5), are sued in the evaluation process.

$$PT = (TotalClockTicks/TickFrequency) * 1000 \tag{5}$$

From these results it can be seen that SW2S is somehow similar to SW3S with less computations. And also DW3S has many false positives FP. SW2S is better that SW3S in Precision and in processing time. Single window method outperforms Different Window method. Using frame difference method gives better Precision and less in computation time that GMM.

In [22], SVM with HOG are used on RPI 3 which is better than RPI 1 in terms of computation power and memory. A new approach is introduced to reduce the search process to find the object of interest in the image. This can be achieved through filtering the overlapping areas of bounding boxes and detect only one to be tested instead of scanning the whole image. In [21] and [22], both systems use down-sampling to cover the large objects.

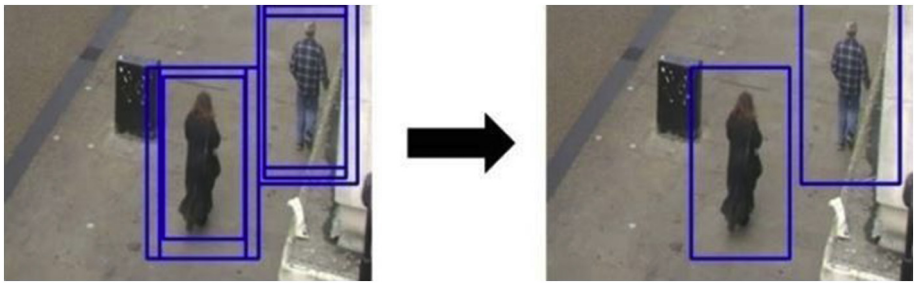

Fig. 13. Detection result filtration in [17].

Table 1
TOWNCENTER and CAVIAR dataset results.

	TOWNCENTER dataset				CAVIAR dataset			
	Processing time (ms)	Accuracy %	Recall %	Precision %	Processing time (ms)	Accuracy %	Recall %	Precision %
Original HOG	2544.4	73.23	66.65	99.75	2559.5	65.05	60.55	99.39
SW3S	463.98	65.94	58.16	98.54	435.63	61.33	61.06	92.86
SW3SMOG	675.71	61.57	53.83	95.39	776.02	55.34	53.02	94.47
SW2S	333.52	62.31	53.35	98.83	291.99	52.98	49.29	92.07
SW2SMOG	500.93	58.48	48.74	98.02	515.19	49.1	43.81	93.51
DW3S	416.58	55.07	53.02	84.05	464.44	31.7	55.96	39.79
DW3SMOG	591.51	54.47	52.94	82.19	612.47	32.72	56.14	41.27

After down-sampling steps, there would be many rectangles surrounding the human. To remove this redundancy a non-maxima suppression is used to remove the feature points that are likely to be close to each other. This step called filtering, Fig. 14.

For 2 consecutive frames, the first frame is searched to get the bounding boxes around humans. Then using the result of the first/odd image, in the second/even image the algorithm chooses a bounding box detected in the odd image with starting position xi and yi. Then iterating sliding step ($s = 0.5$) that will create a new position for the window till the person is covered. As a result, the odd image will be scanned as a whole, but in the even image only the detected bounding boxes are scanned. So, the HOG will be applied on some area(s) instead of the whole image and hence, the time required in processing is reduced.

Three videos of different scenarios that contain many people with 300 frames are used in the testing process. The first video contains three persons in non-complex scene. The second video contains many people walking around (Fig. 15). The third one contains one person leaving a shop. Table 2, shows a comparison between the original HOG and the method used by the authors. The results are a little lowered but the system is faster than the original HOG.

In [23], thermal images were utilized where objects are represented in terms of heat or body temperature. Thermal cameras can detect the infrared radiation emitted by different objects in the scene. As the thermal information are not enough for object detection, features and descriptors are extracted.

HOG is used as the features on the thermal images after extracting the ROI to reduce the processing time to detect and classify pedestrians using SVM. The system consists of three steps: Foreground estimation, HOG for human detection, and human shape similarity.

In [23], the authors collected their own thermal dataset used in training and they used BU-TIV (thermal infrared video) dataset. Training dataset consists of 3620 positive images and 6096 negative images and samples of the training dataset is depicted in Fig. 16. Two datasets of BU-TIV dataset are used: atrium-red and atrium-orange. The results on atrium-red and atrium-orange are depicted in Table 3.

In [24], human detection is used in different application. A system is built to perform image classification to detect obstacles with

Fig. 14. Filtering bounding boxes using non-maxima suppression in [18].

feedback in means of vibration to help blind people in navigation. The detection of obstacles is in middle, left, and right directions. The classification has two categories: walkable and non-walkable. The authors used HOG features with three classifiers and compared the three classifiers: Naïve Bayes, AdaBoost, and SVM, to each other. Naïve Bayes is a probability-based classifier, while AdaBoost is a weighted sum of the outputs of weak classifiers. About 15 min of video manually recorded. 3000 still frames produced (50% walkable, 50% non-walkable) manually labeled. Non-walkable images are the ones that contain dirt, rubbish, bins, water puddles, etc. Five different characteristics paths were considered. Focusing on a single kind of path could improve the accuracy of the system. System architecture is depicted in Fig. 17. System results and performance can be shown in Table 4.

In [43], the authors addressed a new challenge in human detection in aerial images. Human detection in aerial images is very complex due to different poses, scales, etc. Traditional classifiers can't deal with such problems in aerial images. As a result, deep learning is utilized through the use of CNN. The authors used a saliency map technique that tries to reduce the processing time through reducing the search space using the classifier window. Saliency map finds the object of interest quickly through the visual differences of image elements.

The used features are Haar features and local binary patterns (LBPs). Haar features are visual attributes extracted from images based on the Haar wavelets, Fig. 18. These features are performed using black and white regions rectangles. Each rectangle represents a feature with a single value. This value is obtained by subtracting the white and black regions. LBP are texture-based features. It calculates the spatial variations of pixels intensities. These features can represent fine details, resistant to illumination changes, and have low computations.

Table 2
Comparison of original HOG and proposed system in [18].

	Video number	Processing time (s)	F-score
Original HOG	1	0.27	0.70
	2	0.27	0.76
	3	0.28	0.90
Proposed method	1	0.15	0.68
	2	0.25	0.74
	3	0.14	0.92

In [43], the authors examined two classifiers: cascade classifier and CNN. Cascade classifier consists of several weak classifiers. Combining several weak classifiers produces a strong classifier that is computational-effective (Fig. 19).

The CNN used in [43] consists of 8 layers, five convolutional layers and three fully connected layers. The output layer is binary classifier. The input the CNN is RGB image of size: $227 * 227 * 3$. This is a cropped version of original image of size: $256 * 256 * 3$. Fig. 20 shows the model structure.

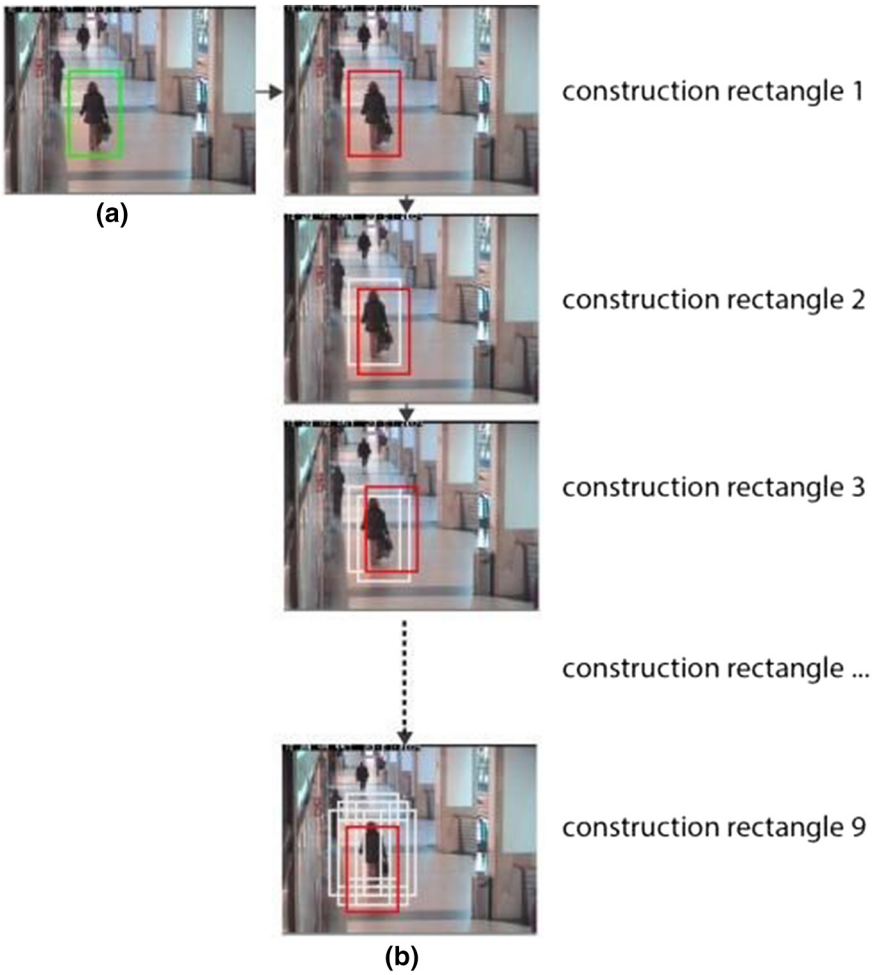

Fig. 15. (a) The first input image with detected rectangle. (b) The construction of the rectangle around bounding box in (a) in [18].

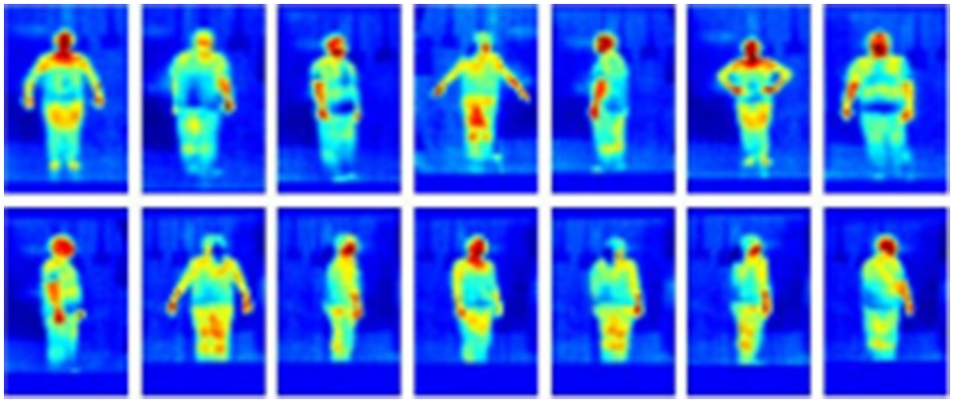

Fig. 16. Samples of positive training thermal images in [19].

Table 3
Atrium-red and atrium-orange performance and accuracy.

	Atrium-red			Atrium-orange		
	Processing time (PT)	Precision	Recall	Processing time (PT)	Precision	Recall
HOG	321.41	0.4989	0.4836	321.82	0.6489	0.3844
HOG region	81.89	**0.7522**	0.4631	81.60	**0.6804**	0.3448
HOG region + shape	94.01	0.7377	**0.5603**	91.93	0.6697	**0.4488**

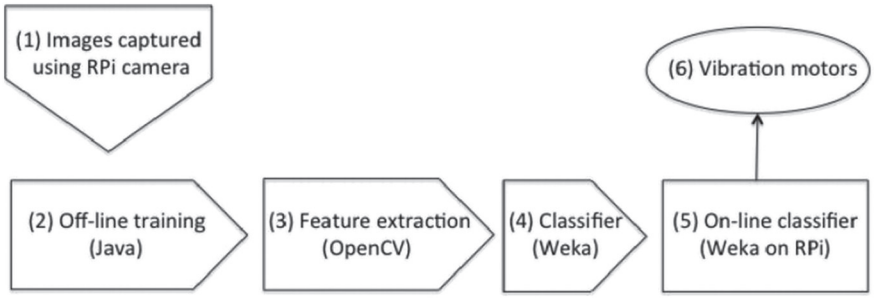

Fig. 17. System architecture in [20].

In this work two datasets has been used. One in training and the other is in testing. GMVRT-v2 [44] dataset is the one used in training phase. It consists of 3846 positive images and 13,821 negative images. Other dataset has been produced during the experiments. 15 thermal images with resolution $80 * 60$ pixels captured using low cost Long Wave Infrared thermal camera named "FLIR" Lepton. Another 15 conventional color images captured using Raspberry Pi camera "Raspicam v 1.3" with resolution of $1280 * 720$ pixels. All images are taken from the same altitude and at the same time so, objects that appear on both images, thermal and color, can be correlated.

Saliency Maps and Thermal Image Processing are the two object detection techniques used. Both of them detect the regions that have high probabilities to have the interested object. As a result, they reduce the search space in the image.

Two platforms have been used in testing Raspberry Pi 2 Model B, and mobile ground control stations (MGCS). From the results, the use of thermal images is better than saliency maps as they require smaller execution time.

Table 6 shows that cascade classifier is the best in terms of execution time. Table 5 shows that CNN outperforms other classifiers used with about 94 true positive (TP) and only 6 false negative (FN). CNN

Table 4
Naive, AdaBoost, and SVM result comparison.

	Naive Bayesian			AdaBoost			SVM		
Class	Precision	Recall	ROC area	Precision	Recall	ROC area	Precision	Recall	ROC area
Walkable	0.77	0.57	0.73	0.79	0.65	0.76	0.79	0.87	0.85
Non-walkable	0.55	0.76	0.71	0.60	0.75	0.76	0.79	0.67	0.85
Average	0.68	0.64	0.73	0.71	0.69	0.76	0.79	0.79	0.85

Fig. 18. Haar features, a) border features, b) point features, and c) line features, in [40].

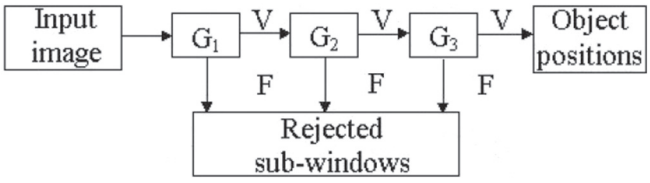

Fig. 19. Cascade classifier, V) the passed features to the next stage, F) rejected features, in [40].

true negative is still need to be enhanced as there is 22 false positive (FP). CNN is the best in terms of classification accuracy, but it requires more time. So, to provide real-time recognition along with good detection accuracy, CNN and thermal images are combined. This combination reaches 1.08 frame per second (fps).

Table 7 summarizes a comparison of the different systems illustrated in Section 4. It is clear that HOG is the prominent feature extraction techniques in the human detection task. SVM outperforms other classifiers despite the fact that only linear kernel is used.

5. Results and analysis

As indicated in Section 4, there is no unique dataset used by all systems. Consequently, there is a need to have a unified benchmark that allow the comparison between different systems. There are also different Raspberry Pi versions utilized by the systems which greatly affect the system performance. Here, we unified the dataset used in training and testing the systems in comparison. One Raspberry Pi device is used to identify its effect on the system performance. We utilized INRIA dataset as a benchmark. The images used in training are INRIA training images. The images are flipped horizontally and downscaled, by factor of 1.05, versions to enlarge the positive images and balance the dataset. The total number of positive samples is 3711 images, and the total number of negative samples is 4000 images.

For the embedded device, the utilized device is Raspberry Pi 3 model B with the following specs: 1 GB RAM, quad-core 64-bit ARM Cortex-A53 processor clocked at 1.2 GHz, Broadcom VideoCore IV of 400 MHz GPU. Raspberry Pi 3 is faster than Raspberry Pi 2 by 50%.

The systems tested are standard HOG with SVM and single window 2 scales (SW2S) method used in [21] with both HOG and SVM. These systems utilized the linear kernel of SVM.

Table 8 contains the comparison results between these systems. It also contains investigation of the effect of using other kernels in SVM on both accuracy and performance. Thus, other kernels such as polynomial and RBF are tested. Different measures were used to compare different systems such as fusion matrix, F-score, and processing time (PT).

As illustrated in Table 8, different SVM kernels are utilized with both systems. Utilizing linear SVM with SW2S achieve somehow higher results as the F-score is 0.9314 in this case while it is 0.9288 for linear SVM with standard HOG. However, the average processing time increased by a small fraction of time (about 0.06 s). As a result, utilizing linear SVM with SW2S gives better results than linear SVM with standard HOG.

When utilizing polynomial SVM kernel with both systems the F-score of the polynomial SVM with standard HOG is 0.9162 while it is 0.8385 for the polynomial SVM with SW2S. Consequently, although the (TP) detection accuracy of polynomial SVM outperforms the other three settings (linear SVM with both systems and polynomial SVM with standard HOG), it is not the better choice in such a case.

Radial basis function (RBF) SVM kernel is also utilized. RBF is used with both systems. Both of these systems are better than polynomial SVM with SW2S. This can be seen from the F-score of these systems which is 0.8531 and 0.8436 for RBF with standard HOG and RBF with SW2S respectively and it is 0.8385 for polynomial with SW2S. The F-score measures of RBF systems indicate that although the (TN) detection accuracy is reduced, the overall system is accepted. The average processing time for RBF systems is 1.7677 s and 2.1064 s for RBF with standard HOG and RBF with SW2S respectively.

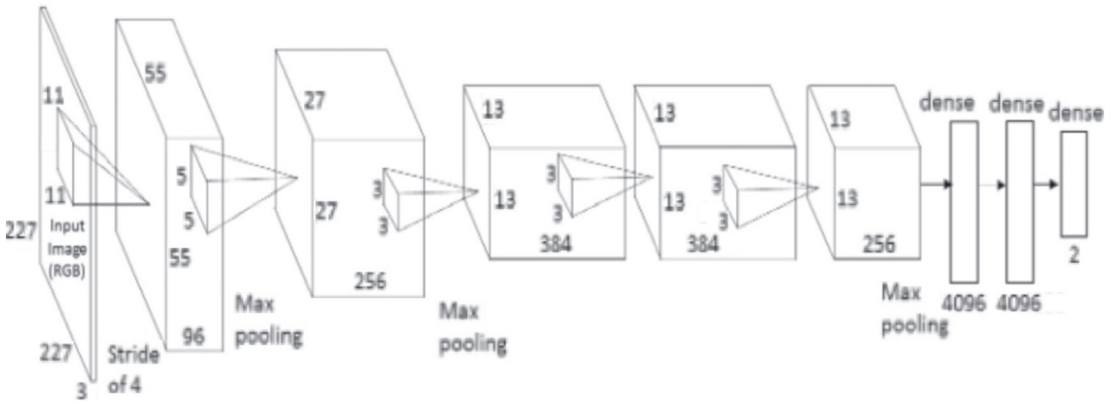

Fig. 20. The proposed CNN architecture in [40].

Table 5
Confusion matrices of Haar cascade, LBP, and CNN.

		Haar cascade		LBP		CNN	
		Labeled					
		People	Negative	People	Negative	People	Negative
Classification	People	59	31	70	48	94	22
	Negative	41	69	30	52	6	78

Table 6
Best and worst case execution time, in [40].

	Raspberry Pi 2		MGCS	
Total time	Min	Max	Min	Max
SM + CNN	5.41	13.23	0.9	2.2
T1 + CNN	3.89	11.86	0.39	0.93
SM + Harr cascade	5.22	9.37	0.34	0.61
T1 + Harr cascade	1.99	7.97	0.07	0.28
SM + LPB cascade	5.74	9.69	0.45	0.76
T1 + LBP cascade	1.91	7.08	0.07	0.26

6. Conclusion and future directions

Several types of human detection systems have been surveyed. Each system has a different environment and settings. Different features and classifiers are utilized in human detection problem. However, HOG was mostly used as well as SVM in the classification stage. Different motion detection methods are available but frame difference and background subtraction are the most frequently used. Due to Raspberry Pi limited resources, other complex methods were not utilized. Recently, embedded devices have great advances that will enable the use of complex models such as ANN or CNN. Also, other SVM kernels such as polynomial and RBF other than linear kernel can be used.

It can be noticed that the most used kernels of SVM in embedded systems is linear kernels. Linear kernel computations are not as heavy as other kernels such as polynomial or RBF kernels. The results can be enhanced if other kernels are used, however that can affect the performance.

Another factor may increase the computations when using other kernels is the length of HOG feature vector. Dimensionality reduction methods can be utilized to make HOG feature vector length shorter. As a result, the use of RBF or polynomial kernels may not require high computations as expected. Based on our analysis in Section 4, other kernels in SVM can enhance the results with small increase in processing time (PT) that may be acceptable.

Computational resources have increased in the subsequent models of RPI which open the potential to try other machine learning techniques in human detection systems such as neural networks. Current devices can run neural networks with sufficient layers.

Human detection, in general, has many challenges like various poses, occlusion, and clutter. Consequently, features should be well designed to overcome these challenges. Utilizing convolutional neural networks (CNN) can learn features that can overcome the limitations addressed. However, CNN computations are heavy and that can greatly affect the system performance. To overcome this limitation, other architectures can be utilized instead of standard CNN. Currently, there are efforts to make a lightweight version of the CNN that can be used in limited resource environment such as embedded devices, or mobiles such as MobileNets [45], SqueezeNets [46], and lightweight CNN (L-CNN) [47]. Some architecture tries to enhance the CNN to reduce the processing time required while keeping its accuracy the same. CNN is mainly based on matrices multiplications. For a single convolutional layer, the multiplication is of n-dimensional matrices. As a result, the complexity will be huge. As a result, a current important research area is to improve the convolution part performance to reduce the overall complexity, and hence, reduce the processing time required.

Conflict of interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Table 7
Summary of different covered systems in human detetection.

Name	Classifier	Features	Dataset	Raspberry Pi specs
[17]	SVM	HOG with different scales + region of interest (ROI)	INRIA in training CAVIAR & TOWNCENTER in testing	RPI 1 model B: 700 MHz single core processor, 512 MB memory and Broadcom VideoCore IV @ 250 MHz GPU. The operating system used was Raspbian.
[18]	SVM	HOG	CAVIAR	RPI 3: Quad-core processor (ARM Cortex-A53) @1200 MHz, 1 GB SDRAM, BroadCom VideoCore IV @400 MHz of GPU.
[20]	SVM naïve Bayes AdaBoost	HOG	About 15 min of video manually recorded. 3000 still frames produced (50% walkable, 50% non-walkable) manually	RPI
[19]	SVM	HOGHOG + ROIHOG + ROI + shape similarity	Manual dataset in training BU-TIV (atrium-red & atrium-orange) in testing	RPI 3 model B:1.2 GHz quad-core processors, and 1 GB memory using Raspbian Jessie with Pixel as an operating system
[40]	CNN cascade classifiers	Haar features local binary pattern (LBP)	GMVRT-v2 in training and testing	RPI 2 model B v1.1: 900 MHz quad-core ARM Cortex-A7 CPU running the "regular" Raspian operating system. Mobile ground control stations (MGCS): consists in a laptop with Intel Core i5-3210 M CPU running the Linux Ubuntu 14.04 operating system.

Table 8
Comparison of SVM linear kernel and other SVM kernels on original HOG and SVM system used in [9], [20] and system used in [17].

	TP (%)	TN (%)	FP (%)	FN (%)	PT (s)	F-score
Linear SVM + standard HOG with SVM	93.209	92.5	7.5	6.791	1.4926	0.9288
Linear SVM + SW2S [17]	94.907	91.103	8.8965	5.093	1.5458	0.9314
Polynomial SVM standard HOG with SVM	94.567	88.135	11.865	5.433	1.5085	0.9162
Polynomial SVM + SW2S [17]	97.963	64.310	35.6898	2.037	1.5718	0.8385
RBF SVM + standard HOG with SVM	95.925	71.052	28.948	4.075	1.7677	0.8531
RBF SVM + SW2S [17]	98.132	65.475	34.525	1.868	2.1064	0.8436

References

[1] H.S. Parekh, D.G. Thakore, U.K. Jaliya, A survey on object detection and tracking methods, Int. J. Innov. Res. Comput. Commun. Eng. 2 (2014) 2970–2978.
[2] N. Singla, Motion detection based on frame difference method, Int. J. Inf. Comput. Technol. 4 (2014) 1559–1565. http://www.ripublication.com/irph/ijict_spl/ijictv4n15spl_10.pdf.
[3] D.G. Lowe, Distinctive image features from scale-invariant key-points, Int. J. Comput. Vis. 60 (2004) 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94.
[4] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (SURF), Comput. Vis. Image Underst. 110 (2008) 346–359. https://doi.org/10.1016/j.cviu.2007.09.014.
[5] P. Sabzmeydani, M. Greg, Detecting pedestrians by learning shapelet features, Comput. Vis. Pattern Recognition. CVPR'07. IEEE Conf, 2007. pp. 1–8. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.205.8978&rep=rep1&type=pdf.
[6] B. Wu, R. Nevatia, Detection of multiple, partially occluded humans in a single image by Bayesian combination of edgelet part detectors, Proc. IEEE Int. Conf. Comput. Vis. 1 (2005) 90–97.
[7] Y. Jiang, J. Ma, Combination features and models for human detection, Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (07-12-June 2015) 240–248. https://doi.org/10.1109/CVPR.2015.7298620.
[8] Y. Mu, S. Yan, Y. Liu, T. Huang, B. Zhou, Discriminative local binary patterns for human detection in personal album, 26th IEEE Conf. Comput. Vis. Pattern Recognition CVPR, 2008. https://doi.org/10.1109/CVPR.2008.4587800.
[9] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, IEEE Comput. Soc. Conf. 1 (2005) (2005) 886–893.
[10] K.B. Bhangale, Human body detection in static images using HOG & piecewise linear SVM, Int. J. Innov. Res. Dev. 3 (2014) 179–184.
[11] S. Theodoridis, K. Koutroumbas, Pattern Recognition, 4th ed. ed., Elsevier. 2009. https://doi.org/10.1016/B978-1-59749-272-0.X0001-2.
[12] X. Wang, T.X. Han, S. Yan, An HOG-LBP human detector with partial occlusion handling, Comput. Vision, 2009 IEEE 12th Int. Conf., 2009. pp. 32–39. https://doi.org/10.1051/matecconf/20165602003.
[13] D.T. Nguyen, W. Li, P.O. Ogunbona, Human detection from images and videos: a survey, Pattern Recognit. 51 (2016) 148–175. https://doi.org/10.1016/j.patcog.2015.08.027.
[14] S. Shalev-Shwartz, S. Ben-David, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press. 2013. https://doi.org/10.1017/CBO9781107298019.
[15] F. Chollet, Deep Learning With Python Manning Publications Company, Manning Publications Company. 2017.
[16] M. Paul, S.M.E. Haque, S. Chakraborty, Human detection in surveillance videos and its applications - a review, EURASIP, J. Adv. Signal Process. 2013 (2013) 176. https://doi.org/10.1186/1687-6180-2013-176.
[17] P.B. Patel, V.M. Choksi, S. Jadhav, M. Potdar, Smart motion detection system using Raspberry Pi, Int. J. Appl. Inf. Syst. 10 (2016) 37–40.
[18] A.N. Ansari, M. Sedky, N. Sharma, A. Tyagi, An Internet of things approach for motion detection using Raspberry Pi, Proc. 2015 Int. Conf. Intell. Comput. Internet Things., 2015. pp. 131–134. https://doi.org/10.1109/ICAIOT.2015.7111554.
[19] P. Jansi, K. Mahesh, A review on motion detection and tracking techniques, Int. J. Mod. Trends Sci. Technol. (2017) 3.
[20] S.A. Shifani, Security system using Raspberry Pi, Third Int. Conf,. Sci. Technol. Eng. Manag. (ICONSTEM), IEEE. 2017, pp. 863–864.
[21] M. Noman, S.A. Velastin, An optimized and fast scheme for real-time human detection using Raspberry Pi, Digit. Image Comput. Tech. Appl. (DICTA), 2016 Int. Conf., 2016, IEEE. 2016, pp. 1–7.
[22] S. Saypadith, W. Ruangsang, S. Aramvith, Optimized human detection on the embedded computer vision system, Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC), 2017, 2017, IEEE. 2017, pp. 1707–1711.
[23] S. Rujikietgumjorn, N. Watcharapinchai, Real-time HOG-based pedestrian detection in thermal images for an embedded system, 2017 14th IEEE Int. Conf. Adv. Video Signal Based Surveillance, 2017, AVSS. 2017, https://doi.org/10.1109/AVSS.2017.8078561.
[24] D.K. Aljasem, M. Heeney, A.P. Gritti, F. Raimondi, On-the-fly image classification to help blind people, Proc. — 12th Int. Conf. Intell. Environ. IE 2016 (2016) 155–158. https://doi.org/10.1109/IE.2016.33.
[25] C. Saunders, M.O. Stitson, J. Weston, L. Bottou, B. Schoelkopf, A Smola, Support vector machine — reference manual, J. Mach. Learn. Res. 7 (1998) 1687–1712. http://eprints.ecs.soton.ac.uk/8959/.
[26] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (1995) 273–297. https://doi.org/10.1023/A:1022627411411.
[27] V.N. Vapnik, The Nature of Statistical Learning Theory, 2000. https://doi.org/10.1109/TNN.1997.641482.
[28] P. Melin, O. Castillo, Studies in Fuzziness and Soft Computing, 172. 2005. https://doi.org/10.1007/3-540-32367-8_3.
[29] Y. Ma, G. Guo, Support Vector Machines Applications, Springer, New York, 2014.
[30] R. Reddy, Detection of human bodies from the background in an image using piecewise linear-support vector machine, J. Innov. Electron. Commun. Eng. 2 (2014) 56–64.
[31] C. Papageorgiou, A trainable system for object detection in images and video sequences, Int. J. Comput. Vis. 38 (2000) 15–33. https://doi.org/10.1023/A:1008162616689.
[32] L. Wang, J. Shi, G. Song, I. Shen, Object detection combining recognition and segmentation, Comput. Vis. — ACCV 2007 (2007) 189–199. https://doi.org/10.1007/978-3-540-76386-4_17.
[33] B. Wu, R. Nevatia, Detection and tracking of multiple, partially occluded humans by Bayesian combination of edgelet based part detectors, Int. J. Comput. Vis. 75 (2007) 247–266. https://doi.org/10.1007/s11263-006-0027-7.
[34] Z. Lin, L.S. Davis, D. Doermann, D. Dementhon, Hierarchical part — template matching for human detection and segmentation, IEEE 11th Int. Conf. Comput. Vis. (2007) 1–8. https://doi.org/10.1109/ICME.2009.5202576.
[35] B. Wu, R. Nevatia, Cluster boosted tree classifier for multi-view, multi-pose object detection, Proc. IEEE Int. Conf. Comput. Vis. (2007) https://doi.org/10.1109/ICCV.2007.4409006.
[36] M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, A. Zisserman, The Pascal visual object classes (VOC) challenge, Int. J. Comput. Vis. 88 (2010) 303–338. https://doi.org/10.1007/s11263-009-0275-4.
[37] L. Bourdev, J. Malik, Poselets: Body part detectors trained using 3D human pose annotations, 2009 IEEE 12th Int. Conf. Comput. Vis., 2009. pp. 1365–1372. https://doi.org/10.1109/ICCV.2009.5459303.
[38] P. Dollar, C. Wojek, B. Schiele, P. Perona, Pedestrian detection: an evaluation of the state of the art, IEEE Trans. Pattern Anal. Mach. Intell. 34 (2012) 743–761. https://doi.org/10.1109/TPAMI.2011.155.
[39] C. Wojek, S. Walk, B. Schiele, Multi-cue onboard pedestrian detection, 2009 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work. CVPR Work. 2009, 2009, IEEE. 2009, pp. 794–801. https://doi.org/10.1109/CVPRW.2009.5206638.
[40] D. Gerónimo, A.M. López, A.D. Sappa, T. Graf, Survey of pedestrian detection for advanced driver assistance systems, IEEE Trans. Pattern Anal. Mach. Intell. 32 (2010) 1239–1258. https://doi.org/10.1109/TPAMI.2009.122.
[41] S. Paisitkriangkrai, C.S.C. Shen, J.Z.J. Zhang, An experimental study on pedestrian classification using local features, IEEE Int. Symp. Circuits Syst. 28 (2008) (2008) 1863–1868. https://doi.org/10.1109/ISCAS.2008.4542024.
[42] A. Ess, B. Leibe, L.V.a.n. Gool, Depth and appearance for mobile scene analysis, Proc. IEEE Int. Conf. Comput. Vis. (2007) https://doi.org/10.1109/ICCV.2007.4409092.
[43] D.C.D.e. Oliveira, M.A. Wehrmeister, Towards real-time people recognition on aerial imagery using convolutional neural networks, 2016 IEEE 19th Int. Symp. Real-Time Distrib. Comput, 2016. pp. 27–34. https://doi.org/10.1109/ISORC.2016.14.
[44] R. Blondel, P. Potelle, A. Pegard, C. Lozano, Fast and viewpoint robust human detection for SAR operations, IEEE Int. Symp. Safety, Secur. Rescue Robot (2014) 1–6. https://doi.org/10.1109/SSRR.2014.7017675.
[45] A.G. Howard, Z. Menglong, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Arxiv Prepr. Arxiv1704.04861, 1. 2017.arXiv:1704.04861.
[46] F.N. Iandola, S. Han, M.W. Moskewicz, K. Ashraf, W.J. Dally, K. Keutzer, Squeezenet: Alexnet-Level Accuracy With 50× Fewer Parameters and 0.5 MB Model Size, Arxiv Prepr. Arxiv1602.07360, 4. 2016, 1–13. https://doi.org/10.1007/978-3-319-24553-9.
[47] S.Y. Nikouei, Y. Chen, S. Song, R. Xu, B.-Y. Choi, T.R. Faughnan, Real-Time Human Detection as an Edge Service Enabled by a Lightweight CNN, Arxiv Prepr. Arxiv1805.00330, 1. 2018, 1–5. http://arxiv.org/abs/1805.00330.