Embedded Video Processing on Raspberry Pi

Gabor Arva
Brno University of Technology
Department of Radio Electronics
Technicka 12, Brno, Czech Republic
Email: xarvag00@stud.feec.vutbr.cz

Tomas Fryza
Brno University of Technology
Department of Radio Electronics
Technicka 12, Brno, Czech Republic
Email: tomas.fryza@vutbr.cz

***Abstract*—The paper presents a study of existing methods for motion and face detection algorithms and their application to the on-board miniature Raspberry Pi computer. The algorithms realized by OpenCV functions were modified to optimize their operation on the mentioned platform, which could be used as an embedded surveillance system. The paper also mentions the training of a custom classifier for hand detection, what could be further used as a basis for detecting hand gestures.**

***Keywords*—OpenCV, Raspberry Pi, motion detection, cascade classifiers, video surveillance.**

I. Introduction

The paper deals with the design of an embedded surveillance system realized on a Raspberry Pi 3 B minicomputer. The work's main part focuses on the study of selected functions used by modern surveillance systems, like motion detection methods and issues [1] and algorithms used for detecting human faces [2]. After selecting the proper methods, they are developed in the programming language C/C++ in way to exploit the computational power of the embedded minicomputer.

An external web-camera captures static scenes which are used as input data for the image processing algorithms. These algorithms analyze the images in real time, yielding information about the moving objects and saving the video sequence if a motion has occurred. To automatize these tasks, basic Computer Vision approaches [3] are modified and applied to the real-time camera feed. The functions are provided by OpenCV (Open Source Computer Vision), what is an open source library containing over 500 optimized algorithms for image and video analysis and manipulation. It has C++, C, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS [4]. The functions are completely designed with the help of OpenCV libraries and are optimized to operate effectively on the Raspberry Pi platform.

The rest of the paper is organized as follows. Section II shortly describes the used motion detecting process. Section III presents an overview of the main object detecting methods and the concept how to detect any object in a video sequence. Section IV briefly presents the one-board miniature computer Raspberry Pi and section V represents the achieved experimental results. Section VI concludes the paper and proposes the possible future work.

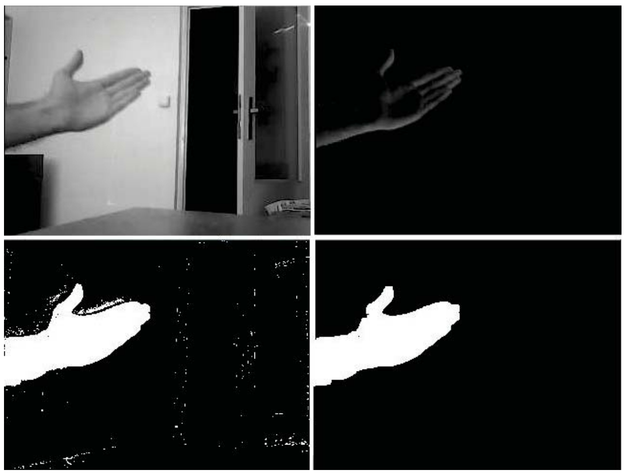

Fig. 1. Motion detecting process

II. Motion Detection

The easiest way to determine a motion in a video sequence is to compare any two consecutive frames in it. This method is called motion detection based on frame difference, and is further described here [5]. In the case of a constant background, the content of the frames are the same, except in the region of moving objects. The processed images are taken from the camera feed. After conversion to gray-scale format, a function calculating the absolute difference returns an output image highlighting the actual motion. A basic segmentation method separates this region and corresponding functions remove the pixel noise.

The blobs appearing in the processed image (as shown in Figure 1) are representing the position of the occurred motion. Further a rectangle is computed around each blob thereby they can be extracted and serve as regions of interest for the following functions.

III. Object Detection

Motion detecting methods provide information about actual events in the video sequence, but they return no information about the features of the moving object itself. The aim of the object detecting algorithm is to recognize the predetermined database of objects in the video sequences. In the case of a surveillance systems, these objects could be the faces of individuals appearing on the screen or license plates of the passing cars. In fact, the detected object can be almost anything and is only limited by the user's needs.

A. *Haar Cascades*

The first known algorithm for object detection included a Cascade of simple classifiers. To detect the specific object in an image, the first step is to learn it's features. However, working only with the intensity values of RGB pictures made the feature calculation computationally expensive. As a result in 2001, Viola and Jones [6] developed a framework using the Haar-like features.

The Haar features could be obtained by monitoring adjacent rectangular areas in a detection window, however, even a small window returns an exhaustive amount of rectangle features. The few of these, which can be combined to form an effective classifier are selected by a learning algorithm implemented to the project.

The chosen classifiers are gathered into cascades and are applied to the detection window starting with the simplest ones and a positive result from these triggers the next more complex classifiers. With this step, the algorithm achieves increased detection performance decreasing the required computation time. Further information and test results are available in the literature [6].

B. *Local Binary Patterns*

The LBP (Local Binary Patterns)[7] have allowed a more modern approach to the field of object detection. Compared to the Haar cascade based algorithm, it has improved detection speed and a more faster learning phase of the classifier, while the main idea remains the same.

The algorithm categorizes the pixels of an image by applying threshold function to them in a 3×3 sub-window with center value. This step is applied to each pixel of an image and the results are saved as 8 bit binary strings or decimal values containing the LBP features. The approach of the LBP operator is shown on Figure 2.

The method's scantiness manifests in the features captured in the small 3×3 area. These sub-windows cannot capture larger scale structures which may contain the dominant features of the detected object. To overcome this obstacle, a new representation was proposed by a group of Chinese engineers called Multi-scale Block Local Binary Patters (MS-LBP)[7]. The basis remains the same and the histogram of these labels are then further used as a texture descriptor. In each sub-region an average sum of image intensity is computed and another threshold function is applied to them by the center block, resulting in a MS-LBP process. The output images are influenced by the scale of chosen blocks: images filtered with small scale values represents more details and micro patterns, while big scale values reduce pixel noise and focuses more on the dominant features of the object.

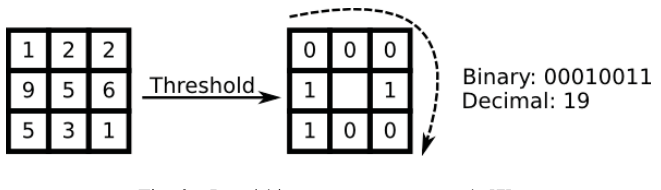

Fig. 2. Local binary patterns approach [7].

C. *Cascade of Classifiers Training*

The OpenCV library offers predefined cascade classifiers for face, eyes or mouth detection. However, if one requires a working classifier to detect other objects, the classifier needs to be trained individually. Fortunately the OpenCV library includes the tools and functions to generate a custom cascade classifier based on Haar or LBP features.

The training time of a Haar cascade classifier can take up to weeks, while an LBP classifier for the same object can be trained in a few hours. Due to this reason an LBP cascade classifier will be trained for the detection of hands.

The training process itself requires a set of positive sample images including the detected object, and a set of negative sample images containing possibly everything except it. According to researches [8], the amount of positive and negative samples may vary around a few thousand, however to acquire such a large amount of pictures about and without the desired object is hard to accomplish.

The negative samples, also called background images, can be taken from random pictures not including the detected object. Obtaining the sufficient amount of them is a time-consuming process. One way is to gather them manually or one can extract frames from video files saving them as individual images. In this paper, the amount of negative images is 800.

For this custom made classifier 100 positive sample images are used including 5 people's hands. A few sample of a the whole database can be seen in Figure 3. A specific OpenCV function is utilized to significantly increase this number [9], generating a large amount of samples from the existing images by applying transformations and distortions to them.

IV. Raspberry Pi

Raspberry Pi is a miniature one-board computer developed by the Raspberry Pi Foundation [10]. The latest model is

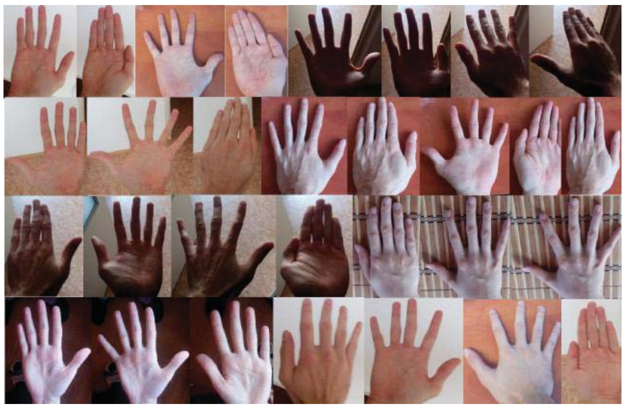

Fig. 3. Set of positive images for classifier training.

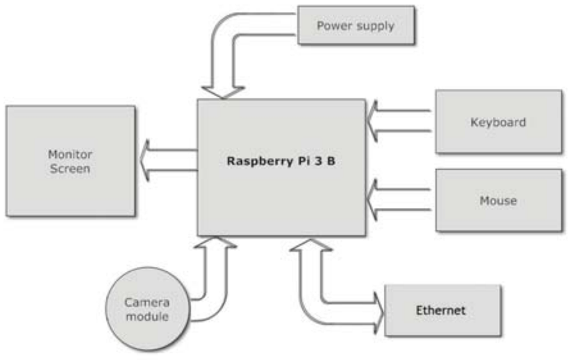

Fig. 4. Block scheme of hardware.

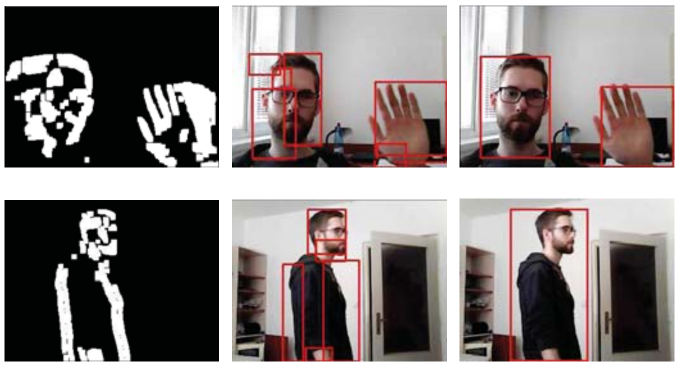

Fig. 5. The effect of rectangle merge

the third generation of Raspberry Pi and it is available since February 2016. As a central control unit for the proposed embedded surveillance system, the Raspberry Pi 3 B and the Raspbian Jessie Lite operating system was selected. The one-board computer is equipped by the 1.2 GHz 64-bit quad-core ARMv8 processor, 1 GB RAM, which are making this minicomputer an appropriate choice for the processing unit of this embedded surveillance system. The mentioned device together with a web-camera are all the hardware required for this project. The block scheme is shown in Figure 4.

V. Experimental Results

A. Comparison of Cascade Classifiers

The OpenCV library includes both Haar-feature and LBP based ready to work cascade classifiers, inter alia, for the detection of frontal faces. The main idea for the detection process itself comes from a tutorial available on the OpenCV website [11]. However, using different cascade classifiers and modifying the size of the input image significantly reduces the computational time required for the detection. Although the Raspberry Pi 3 B has remarkable computational power, it can not be compared to modern computers, what the algorithm had been originally designed for. Considering this, the paper also contains a comparison between the available classifiers, where the observed parameter is the computational time required for the detection process (see Table I). ROI stands for Region of Interest and it's function is discussed hereinafter.

Based on the measured results, the object detection algorithm using LBP cascade classifier is 3 times faster than the Haar features based one.

TABLE I
Computational Time of Used Cascades

Resolution	640×480			Based on ROI		
Comp. Time	min	max	avg	min	max	avg
	[milliseconds]					
Haar classifier	91	140	115.5	1.5	33	17.3
LBP clasifier	20	48	34	0.3	9	4.7

B. Selecting Region of Interests

Another potential to decrease the detection time of the observed algorithm is to reduce the size of the scanned area, limiting it's attention to regions of interest based on ongoing motions. The downside of this method is that the system reacts only to moving objects, making this solution effective in terms of computational time, however, it may not be suitable for every situation.

For example, in the case when a person approaches towards the camera and stops before it, the system can detect the face until the person is moving, but not after the motion ceases, missing the potential to capture the closest images. To avoid the described situation, the position of the detected face is saved in every loop and the region is further scanned until the face is present. This makes possible to track a face on the screen despite it is moving or not.

The motion detecting algorithm's main role is to determine the position of moving objects on the screen, nevertheless it can not specify these objects as individuals. Because the thresholded image of a moving object can consist of more blobs as shown in Figure 5, the algorithm will separate them to more parts. This effect is significant on close objects, since the camera can perceive more details about them, thereby complex objects with stronger contours cause the threshold function to return more blobs.

As it was noted in Section II, a rectangle is computed around each blob and their coordinates are then compared with each other. If the area of two rectangles overlap they are merged together, creating a new one with the correct top-left and bottom-right points. The process iterates while rectangles are present with interfering areas. The result is shown in Figure 5, where the final rectangles serve as regions of interest for the object detection function.

C. Effectiveness of Face detection

Considering the results from the previous measurements, the LBP classifier was selected and further observed. The images shown in Figure 6 are demonstrating the function of the detecting process. A given face is tilted to several position while the result from the classifier is observed. The same

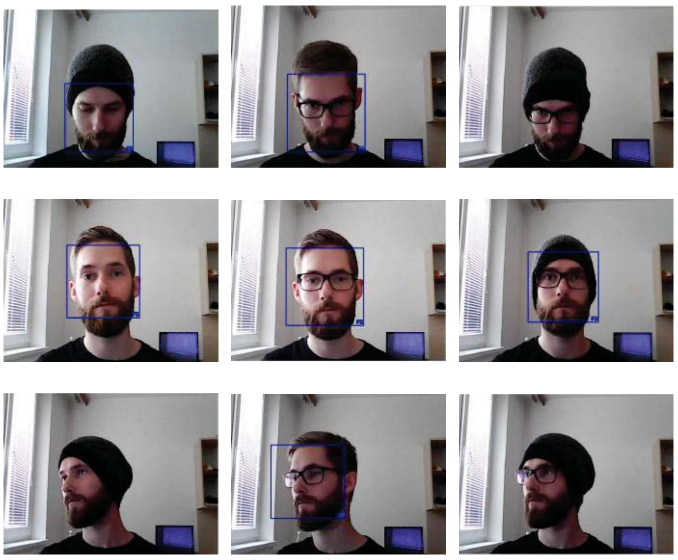
Fig. 6. Results of LBP face detection on face with beard, cap and glasses

procedure is repeated with adding new components to the same face, making it more complex and difficult to detect. These modifications are including facial hair, glasses, a cap or all of them.

Summing up the experiment, when a person is wearing glasses the results show that the detection returns a negative value if the angle between the face and the camera is larger than 45^o. If one is wearing both glasses and a cap, the face is detected only if the person looks to the camera.

D. *Effectiveness of Hand detection*

This section presents the results of the custom made hand classifier. As it was mentioned in Section III C, the amount of positive samples were artificially increased from 100 to 2000, however, classifiers trained by these samples are never as effective as those trained by real samples [8]. The results can be seen in Figure 7 and show the same: if the contours between the hand and the background are strong, they are found with high accuracy, but if the background is complex and the hand is blurred in it, the detected object is not noticed. The classifier could be upgraded by increasing the number of real samples.

VI. Conclusion

In the paper, the basic issues and solutions of embedded video processing for computer vision were presented. The Raspberry Pi computer and several functions from the OpenCV project was used. The ROI selecting and rectangle merge functions significantly increased the overall frame per second ratio of the system. The achieved modifications of the basic video processing algorithms made possible the operation of a surveillance system on the on-board minicomputer. The analysis and testing of frontal face detection algorithms was completed and in addition, a custom LBP classifier for hand and palm detection was trained. The amount of positive

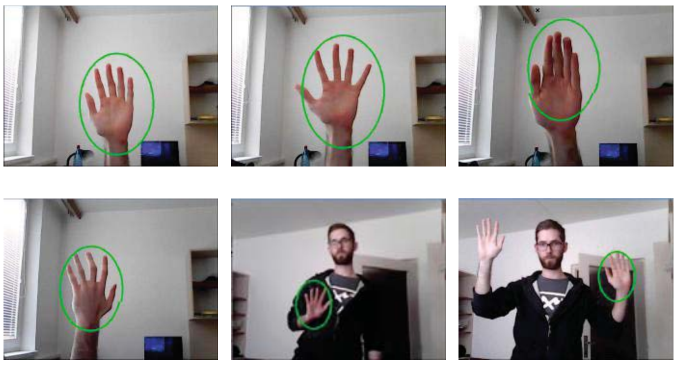
Fig. 7. Results of LBP hand detection

images used for the training process was not sufficient, but the classifier could be upgraded by adding more samples.

Acknowledgment

Research described in this paper was financed by Czech Ministry of Education in frame of National Sustainability Program under grant LO1401, under grant LD15034, and Brno University of Technology Internal Grant Agency under project no. FEKT-S-17-4426. For research, infrastructure of the SIX Centre was used.

References

[1] J. Rahman, Md. Motion Detection for Video Surveillance.. [Online]. Available: http://www.diva-portal.org/smash/get/diva2:518464/FULLTEXT01.pdf

[2] B. C. Lovell, S. Chen, T. Shan. Real-time Face Detection and Classification for ICCTV.. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.331.6676rank=1

[3] R. Szeliski. Computer Vision: Algorithms and Applications. [Online]. Available: http://szeliski.org/Book/

[4] OpenCV. [Online]. Available: http://opencv.org/

[5] N. Singala, "Motion Detection Based on Frame Difference Method," in *International Journal of Information and Computation Technology. 2014,* vol. 4, 2014, pp. 1559-1565.

[6] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–511–I–518 vol.1.

[7] Z. L. L. Z. S. Liao, X. Zhu and S. Li. Learning Multi-scale Block Local Binary Patters for Face Recognition. Center for Biometrics and Security Research & National Laboratory of Pattern Recognition. [Online]. Available: http://www.cbsr.ia.ac.cn/users/lzhang/papers/ICB07/ICB07_Liao.pdf

[8] S. Natoshi Tutorial: OpenCV haartraining. [Online]. Available: http://note.sonots.com/SciSoftware/haartraining.html

[9] OpenCV. Cascade Classifier Training. [Online]. Available: http://docs.opencv.org/trunk/dc/d88/tutorial_traincascade.html

[10] Raspberry Pi. Raspberry Pi Foundation. [Online]. Available: https://www.raspberrypi.org/

[11] Cascade Classifier. [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/tutorials.html