SENSITIVITY OF A VIDEO SURVEILLANCE SYSTEM BASED ON MOTION DETECTION

Alessandro Massaro, Valeria Vitti, Giuseppe Maurantonio, Angelo Galiano

Dyrecta Lab, IT research Laboratory, via Vescovo Simplicio,
45, 70014 Conversano (BA), Italy

ABSTRACT

The implementation of a stand-alone system developed in JAVA language for motion detection has been discussed. The open-source OpenCV library has been adopted for video surveillance image processing thus implementing Background Subtraction algorithm also known as foreground detection algorithm. Generally the region of interest of a body or object to detect is related to a precise objects (people, cars, etc.) emphasized on a background. This technique is widely used for tracking a moving objects. In particular, the BackgroundSubtractorMOG2 algorithm of OpenCV has been applied. This algorithm is based on Gaussian distributions and offers better adaptability to different scenes due to changes in lighting and the detection of shadows as well. The implemented webcam system relies on saving frames and creating GIF and JPGs files with previously saved frames. In particular the Background Subtraction function, find Contours, has been adopted to detect the contours. The numerical quantity of these contours has been compared with the tracking points of sensitivity obtained by setting an user-modifiable slider able to save the frames as GIFs composed by different merged JPEGs. After a full design of the image processing prototype different motion test have been performed. The results showed the importance to consider few sensitivity points in order to obtain more frequent image storages also concerning minor movements. Sensitivity points can be modified through a slider function and are inversely proportional to the number of saved images. For small object in motion will be detected a low percentage of sensitivity points. Experimental results proves that the setting condition are mainly function of the typology of moving object rather than the light conditions. The proposed prototype system is suitable for video surveillance smart camera in industrial systems.

KEYWORDS

Video Surveillance, OpenCV, Image Processing, OpenCV, Image Segmentation, Background Subtraction, Contour Extraction, Camera Motion Sensitivity.

1. INTRODUCTION

Video tracking of moving multiple objects is a topic of interest in the research community. This function was applied for vehicle tracking using Kalman filter [1]. Concerning this function three main approaches to detect and segment the vehicles such as background subtraction method, features based method, and frame differencing and motion based method were studied [2]. In literature a probabilistic based image segmentation model was adopted for human activity detection by highlighting the importance of segmentation accuracy in motion detection [3] which is the main topic of the proposed paper. A moving object system for video surveillance based on a modified region growing algorithms was discussed in [4] by improving high performance of the detection system. Other authors focused their attention on Camera Field of View (CFOV) approach able to detect the static and moving without noise and false detections in the scene [5]. These last two works prove that the motion detection algorithms are important issues for video surveillance image processing. Concerning image detection from a webcam OpenCV libraries

DOI : 10.5121/sipij.2018.9301

provides different functions of image detection and image processing [6], which can be applied for a video surveillance application also for background subtraction approach. Background modeling [7] is often used in various applications to model the background thus detecting moving objects in video surveillance [8], [9], optical motion capture [10],[11] and multimedia [12]. The easiest way to model the background is to capture a background image that does not include any moving objects. In some environments, the background is not available and could change in critical situations such as lighting changes, objects introduced or removed in the field of camera view. To take account of these robustness and adaptation problems, many basic modeling methods have been developed [8],[13]. These background modeling methods can be classified into the following categories:

• Basic Background Modeling [14],[15];
• Statistical Background Modeling [16];
• Fuzzy Background Modeling [17];
• Background Estimation [9],[18].
Other approaches can be found in terms of:
• forecast [19];
• recursion [8];
• adaptation [20];
• mode approach [21].

All of these modeling approaches are used in the context of background subtraction characterized by the following steps and main problems:

• modeling in the background;
• initialization in the background;
• background maintenance;
• foreground detection;
• choice of the size (pixel, block or cluster);
• choice of type of function (color characteristics, edge functions, motion functions and texture functions).

By developing a background subtraction method, all these choices determine the robustness of the method compared with the critical situations encountered in the video sequence such as [9]:

• Image of noise due to a source of poor quality images (NI),
• Jitter Room (CJ),
• Automatic Camera Adjustments (CA),
• Time of day (TD),
• Light switch (LS),
• Automatic start (B),
• Camouflage (C),
• Opening in the foreground (FA),
• Moving background objects (MO),
• Insertion of background objects (IBO),
• Multimodal Background (MB),
• Waking foreground object (WFO),
• Sleeping foreground object (SFO)
• Shadows (S).

Concerning background subtraction OpenCV implements the class called Background Subtractor MOG2 [22]. It is a Gaussian Mixture-based Background Segmentation Algorithm. This algorithm takes the background pixels and assigns a Gaussian Distribution to each one: the algorithm tries

to identify the background by the information from the Gaussian mixture. We list in table 1 some important features and application related the background subtraction approach:

Table 1. Background subtraction features

		Wren et al. [23]	Stauffer et al. [24]	Eveland et al. [25]	Gordon et. al. [26]
Algorithm Characteristics	Color Input	X	X		X
	Depth Input			X	X
	Time Adaptive	X	X	X	
	Luminance Normalized Color	X			X
	Multimodal BG Stats		X		
Robustness to Various Phenomena	Low Visual Texture	X	X		X
	Depth Discontinuities	X	X		X
	Non-Empty Scene Init	X	X	X	
	Illumination Changes	X	X	X	
	BG Object Changes	X	X	X	
	Shadows & Inter-Reflect	X		X	X
	Color Camouflage			X	X
	Depth Camouflage	X	X		X
	Rotating Fan		X		
	Active Display			X	X
	High-Traffic Areas			X	X

.

According with the state of the art authors applied the BackgroundSubtractorMOG2 algorithm in order to study th background subtraction sensitivity in video surveillance detection systems, by providing an approach useful for setting camera parameters.

The paper is structured as follows:

(i) Design of the video surveillance processor by Unified Modeling Language (UML);

(ii) Testing and sensitivity results of a prototype webcam system;

(iii) Conclusions.

In Fig. 1 is illustrated the proposed architecture applied to the video surveillance prototype processor.

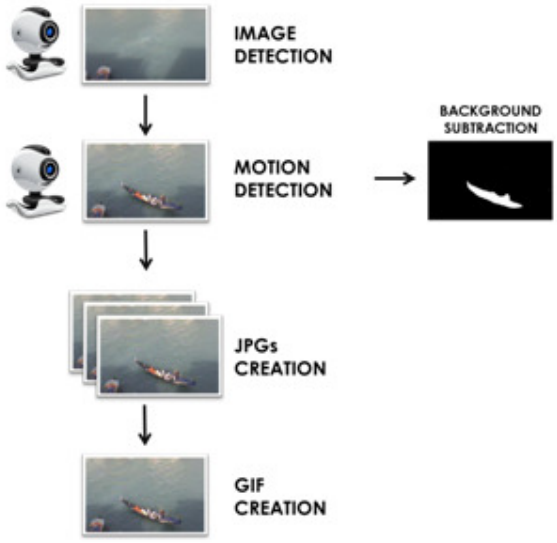

Figure 1. Prototype system architecture.

3. DESIGN AND IMPLEMENTATION OF THE PROTOTYPE VIDEO SURVEILLANCE PROCESSOR

In this section we discuss the design of the video surveillance processor. The tool UMLet 14.2 [27] has been used for the design layouts of the whole prototype system.

The design is developed in the following five layouts:

- use case diagram –UCD- of the whole prototype system (Fig. 2 and Fig. 3);
- class diagram -CD- (Fig. 4);
- Activity Diagram –AD- (Fig. 5 and Fig. 6).

The design of the use cases diagram in Fig. 2 highlights the main functionalities of the following main actors:

• User: user who must monitor the quality process of circuit welding;

• Software (actor according to UML2 standard): actor programmed for real time image processing.

Figure 2 shows the main processing steps such as:

• camera opening;
• camera closing;
• slider management (controller);
• motion retrieval;
• JPGs saving;
• Gif saving;

Figure 2. Generic Use cases diagram (UCD).

The main functions of the actor “Software” are illustrated in Fig. 3 where the video surveillance processor will apply the background subtraction method and will extract the contours of the moving object.

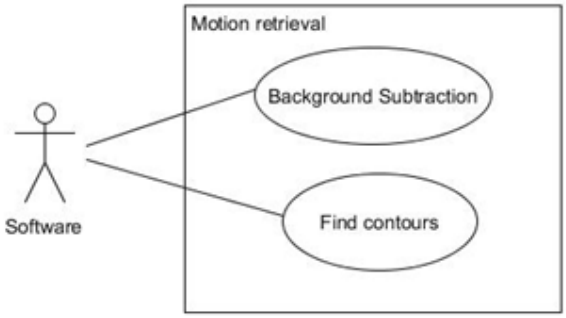

Figure 3. Uses Cases Diagram: motion retrieval.

Classes and objects are the 'building blocks' of the whole prototype structure. An object is a discrete entity, having well-defined boundaries, which encapsulates the state and behavior of a class. A class is the descriptor of a set of objects that share the same attributes, operations, methods, relationships and behaviors. They show the connected system entities according to the static relationships that characterize them. The “Main” class is the core of the application. The first instructions are necessary for the initialization of the *MainWindow* components of the application interface) and of Surveillance, which encapsulate the methods useful for the image processing:

MainWindow Window = new MainWindow();
Surveillance SurveillanceSystem = window.getSurveillanceSystem();

The instructons will also delete the "MyFrame" folder in a preventive manner, in order to clean up the memory before the restarting process about a new data acquisition.

At this point a thread pool will be started. It is necessary for the correct management of the threads that will save the video. Before the starting of the video capture process, it is necessary to check that the webcam is working correctly using the following instruction:

Window.getSurveillanceSystem.getWebCam.isOpened();

In the case of incorrect reading of the webcam the application is immediately interrupted, otherwise you expect 500ms for the initialization of the device and you enter the while loop that will stop the processor only when the application is closed by the user.
The actual reading phase from the webcam takes place by the following instruction:

SurveillanceSystem.getWebCam().read(SurveillanceSystem.getWebcamImage());

where *SurveillanceSystem.getWebCam ()* is an instance of a VideoCapture native OpenCV class required to open a video stream by a webcam.
At this point, after a brief initialization phase of the slider, called through the *Window.pointsInitialize ()* method, is applied the acquisition procedure consisting of the following three basic steps:
• Application of the background subtraction: in this phase, an object of type *BackgroundSubtractorMOG2* (class relative to OpenCV) will be initialized:

BackgroundSubtractor.apply (Mat image, Mat fgmask, double learningRate);

where the input parameters correspond to:
o Image: the next image on which the subtraction algorithm is applied;
o fgmask: the output mask stored in 8 bits;
o learningRate: the amount of speed that the background subtractor uses to adapt to changes.

• Search for movement contour points: it is possible to check if there is an actual difference between the background and the image just captured by the webcam. The function extracts the outlines from an image using the algorithm. The contour extraction is very important for the shape analysis, and for object detection and recognition. To do this, the *findContours ()* function of OpenCV has been adopted.
• Saving the acquired image if it satisfies the movement detection criterion expressed by the following logic condition:

if (SurveillanceSystem.getEdges (). size ()> SurveillanceSystem.getPointsMovementThreshold ()) {SurveillanceSystem.saveScreenshot (Window.getFrame ());}

Where *SurveillanceSystem.getEdges ()* is the outline vector that was filled in the previous step by the function *findContours ()*. The threshold value (*PointsMovementThreshold*) is initially chosen automatically by the prototype software as function of the size of the captured frame. Then it is manually modifiable through a slider.

This "*MainWindow*" class is used to initialize and to manage all the variables necessary for the correct execution and display of the window. In particular it is composed of:

• An instance of the *JFrame* class, which contains all the elements of the application's graphical interface;
• An instance of the *CameraPanel* class, on which the webcam streaming is displayed;
• An instance of the *JLabel* class containing the constantly updated timestamp;
• An instance of the *Surveillance* class able to regulate the acquisition and the screenshots storage;

• An instance of the *JSlider* class that adjusts the number of motion points by changing the percentage of the size of the background:

```
int rate = (Sistem.getPointsMovementThreshold () * 100) /
SurveillanceSystem.getMaxWidth ();
```

Once instantiated, the Surveillance class takes care of loading the openCV library:

```
System.loadLibrary (Core.NATIVE_LIBRARY_NAME);
```

Subsequently, a series of variables will be instantiated to regulate the acquisition and to save screenshots. Among these variables, *pointsMovementThreshold* : this variable allows to vary the image capture threshold by defining how large the movement should be considered significant. Successively the prototype system is connected to the webcam and the *BackgroundSubtractor* function is initialized through the instructions:

```
this.WebCam = new VideoCapture (0);
this.BackgroundSubstractor = Video.createBackgroundSubtractorMOG2 ();
```

The Surverillance class defines the *saveScreenshot* (JFrame frame) function which saves the current contents of the JFrame in a jpg file:

```
BufferedImage image = (BufferedImage) frame.createImage (frame.getSize (). Width,
frame.getSize (). Height);
Graphics g = image.getGraphics ();
frame.paint (g);
g.dispose ();
ImageIO.write (image, "jpg", new File (s.getName () + "\\" +
Constants.IMAGE_NAME_PREFIX + this.PseudoNumber + "-" + (this.fileID ++) + ".jpg"));
this.IsPresentJPG = true;
```

The *CameraPanel* class is responsible for converting a matrix of pixels into an image. In particular, a matrix of pixels is represented by the "*Mat*" class, while an image can be encapsulated in the "*BufferedImage"* type. In the *matToBufferedImage* method (Mat matrix, boolean flag), two following important instructions of the class are considered:

```
Imgcodecs.imencode (". Jpg", matrix, mb);
if (! flag)
this.image = ImageIO.read (new ByteArrayInputStream (mb.toArray ()));
else
this.mask = ImageIO.read (new ByteArrayInputStream (mb.toArray ()));
```

The first instruction specifies the save format, the source matrix and the destination matrix (*mb*). The second instruction uses a stream of bytes to receive the array transformed into a byte array and, by the read method of the static *ImageIO* method, the image is acquired.
The variable "*flag*" allows to understand if the input "matrix" corresponds to the webcam or to the 8 bits mask.
The *Recorder* class is a thread that combines screenshots saved from the "*Surveillance*" class into a GIF file. Before starting, a check is carried out by the following instructions:

```
if (argumentsNum> Constants.MINIMUM_JPEGS_FOR_GIF)
```

where *Constants.MINIMUM_JPEGS_FOR_GIF* is a preset variable defining the minimum number of saved screenshots.

The *Utilities* class provides useful methods, used by several classes. It is certainly important, in order to obtain a reliable video surveillance system, to understand the moment in which the video recording starts. For this reason, the *updateTimestamp ()* function is used: this function returns a string containing the time in the format "DAY-MONTH-YEAR NOW.MINUT.THAYS" and uses the *Java Calendar* class. The *deleteJPGFiles* function is applied to delete JPGs after creating the GIF, while the *deleteFolder* function is able to clean the MyFrame folder.

The *GifSequenceWriter* class is a kind of library that allows the construction of GIF files.

Figure 4. Class Diagram.

The activity diagram of Fig. 5 describes the specific behaviors of the prototype video surveillance software. The activity diagram data flow is represented by the oriented arrows indicating the timeline of the activities.

In Fig. 6 is illustrated the activity diagram related the GIFs saving process.

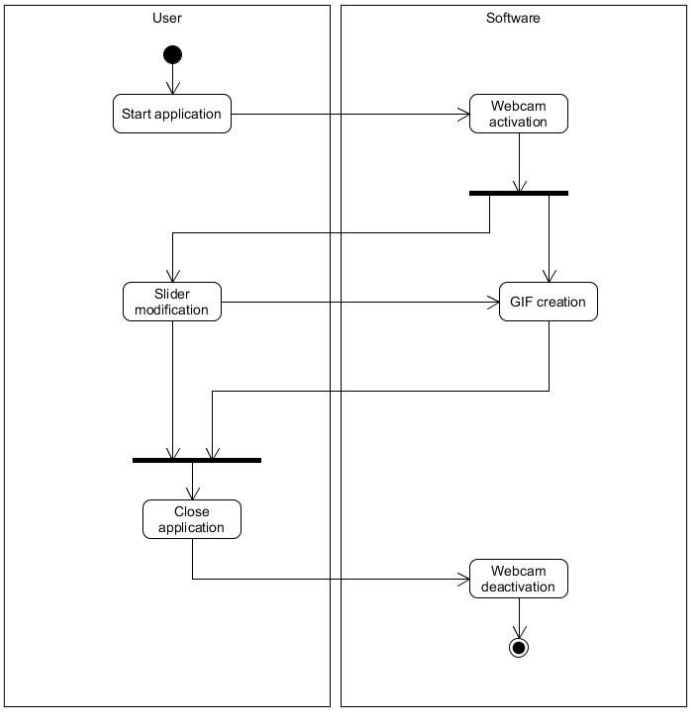

Figure 5. Generic Activity Diagram.

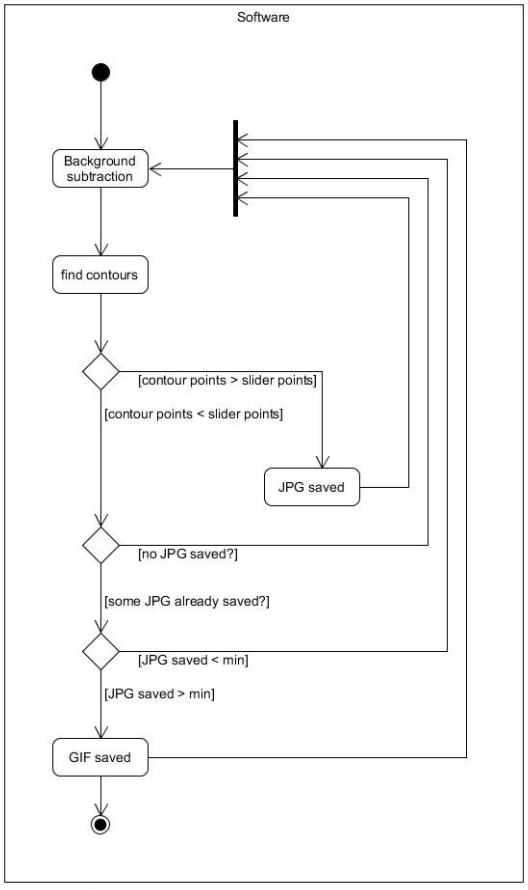

Figure 6. Activity Diagram: GIF creation process.

3.1 GRAPHICAL INTERFACE CONTROLLING MOTION CAMERA SENSITIVITY

The graphical interface (GUI) of the prototype video surveillance processor has been developed by java language using the Eclipse Mars 2.0 platform and the Swing – Window Builder tool (see Fig. 7). As webcam has been used a Logitech High Definition C720. The developed GUI allows to perform all the test.

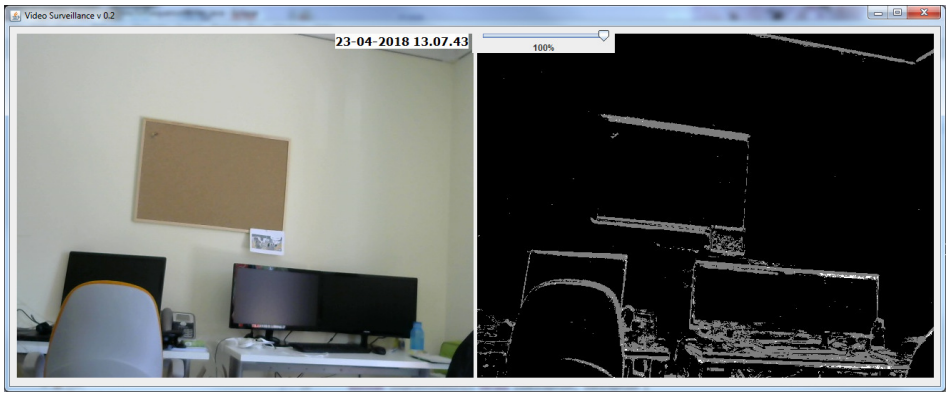

Figure 7. GUI of the prototype.

The GUI is characterised by the following basic elements:

• Streaming webcam (GUI left): the webcam view has been embedded into a JPanel (Swing element). The image is displayed in the *Main.java* class with the following line of code:

Window.getCameraPanel (). MatToBufferedImage (Surveillance.getWebcamImage (), false);

• Streaming foreground (GUI right): a foreground streaming computed with the background subtraction algorithm has been inserted in a JPanel (8 bits in black and white) to understand how the motion is detected from the webcam. The used code line is the following one:

Window.getCameraPanel (). MatToBufferedImage (Surveillance.getForeground (), true);

• Date and time (GUI top left): it represents the date and time of the moment in which the application was opened.
• Sensitivity points slider (GUI top right): the slider allows to select the percentage of points taken into consideration for comparison with the other contour points; user can choose a percentage of significant points for the movement detection, manually modifiable by the slider, by considering the webcam resolution as a reference percentage (as example, for 50% of significant points of an 640x480 image will be considered 320 points).

4. EXPERIMENTAL RESULTS AND TESTING

In this section will illustrate the experimental results indicating the sensitivity behaviour of the adopted background subtraction approach. In order to acquire the basic parameter of the BackgroundSubtractorMOG2 algorithm has been tested the object of Fig. 8 related to a cat model moving a paw. The preliminary parameter to set are the following ones:

- Mode 0: RETR_EXTERNAL = 0 (generic parameter for the recovery of extreme external contours);

- Mode 1: RETR_LIST = 1 (it recovers all the contours without establishing any hierarchical relationship);
- Mode 2: RETR_CCOMP = 2 (it recovers all the contours and organizes them in a two-level hierarchy; at the highest level, there are external borders of the components; at the second level there are the boundaries of the holes; if there is another contour within a hole of a connected component, it is still positioned at the highest level);
- Mode 3: RETR_TREE = 3 (it recovers all the contours and reconstructs a complete hierarchy of nested contours).
- Method 1: CHAIN_APPROX_NONE = 1 (it stores all the points in the profile);
- Method 2: CHAIN_APPROX_SIMPLE = 2 (it compresses the horizontal, vertical and diagonal segments and it leaves only their end points);
- Method 3: CHAIN_APPROX_TC89_L1 = 3 (approximation parameter);
- Method 4: CHAIN_APPROX_TC89_KCOS = 4 (approximation parameter).

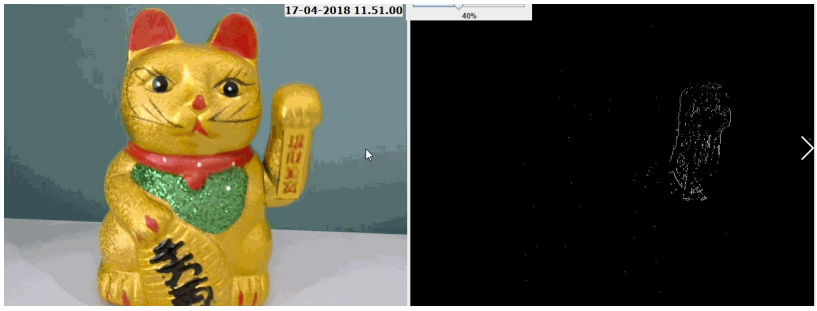

Figure 8. First motion test: (left) cat model and (right) contour extraction of the paw part.

The number of the created JPEG images is defined by the logic condition discussed before: if the contour points are major than the point sensitivity number will be stored the JPEG image, otherwise, if the contour points are minor than the point sensitivity number will be stored the GIF image as a merged image of different JPEG stored images. A high number of stored JPEGs means that more possible moved contours could be detected. For a high GIF image number (high sensitivity response) will occur possible false motion detection. For this reason it is preferable to obtain an average value of GIF number composed by a high number of JPEGs. The choice of the best parameters is defined by the results of Fig. 9 and Fig. 10: the data which follow the major slant angle of the interpolation logarithmic line, are suitable for the parameter setting (in the analysed case mode 1 and method 2). The slant angle in fact represents the capacity of the prototype system to detect movements.

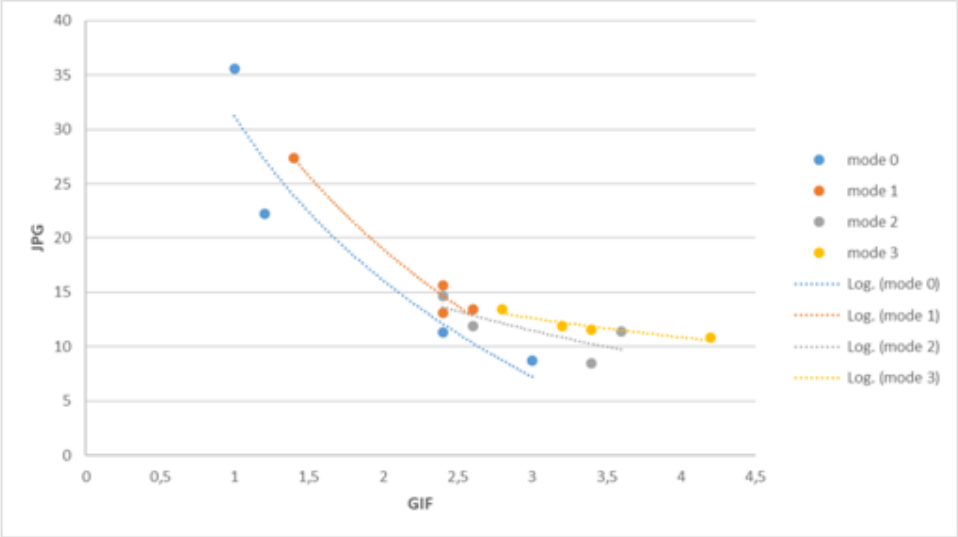

Figure 9. Preliminary test of motion sensitivity for different modes parameters.

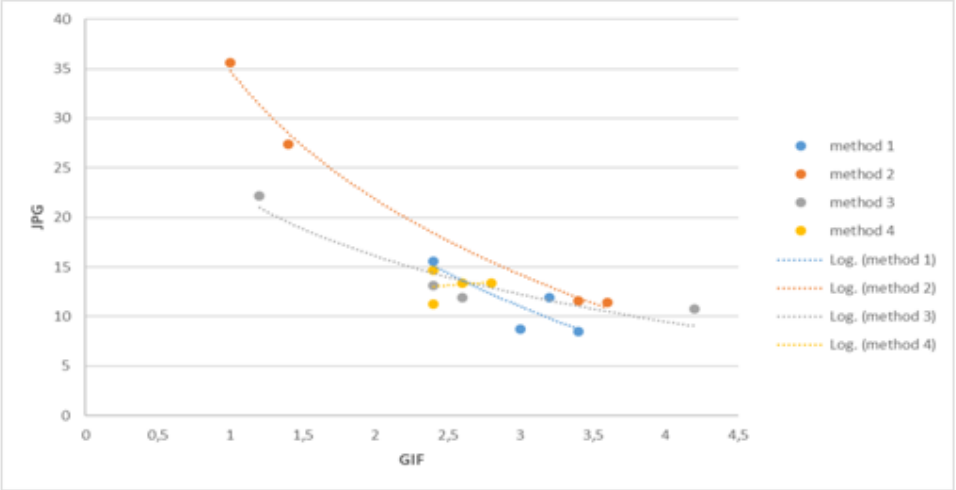

Figure 10. Preliminary test of motion sensitivity for different method parameters.

After the preliminary parameter setting procedure, different motion test have been performed by fixing RETR_LIST=1 and CHAIN_APPROX_SIMPLE=2 according with preliminary test. The first performed test typology is related to a fixed human body with moving harms. In Fig. 11 is illustrated the GUI of this test and an example of the segmentation result. In Fig. 12, Fig. 13, Fig. 14 and Fig. 15 are illustrated the experimental results by histogram plot reporting slider value (20 %, 40 %, 60 %, 80 %, 100 %), GIFs number (x axis), and JPEGs number (y axis).

A good sensitivity condition will be performed by a slider allowing to obtain an average number of GIF images (an high value could represent false detection) and an high number of corresponding JPEG images. In Fig. 16, Fig. 17, Fig. 18, Fig. 19, and Fig. 20 are illustrated sensitivity test results concerning walking man. In Fig. 21, Fig. 22 and Fig. 23 are shown experimental sensitivity results about running man contour detection. Finally in Fig. 24, Fig. 25 and Fig. 26 are illustrated testing results about a launched object (backpack) simulating running dog or cat. The illustrated figures are the most significant and represent test under different

conditions such as camera distance from the moving object and light condition (artificial and natural). In Table 1 are summarized all the experimental results discussed in the conclusion section.

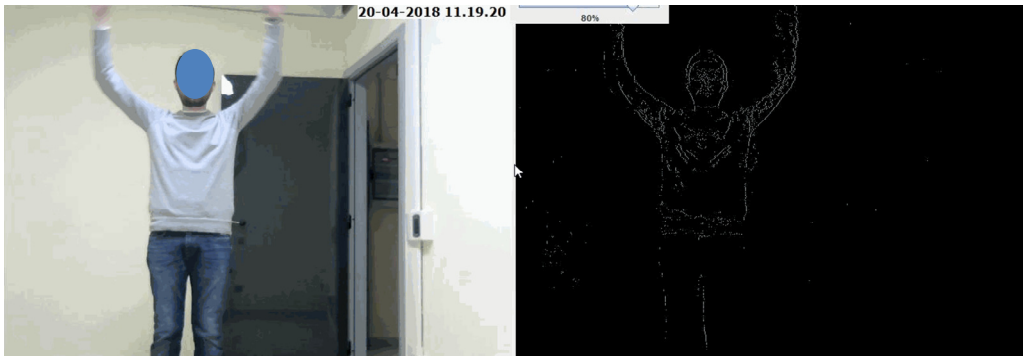

Figure 11. Test 1,2,3,4: fixed human body with moving arms.

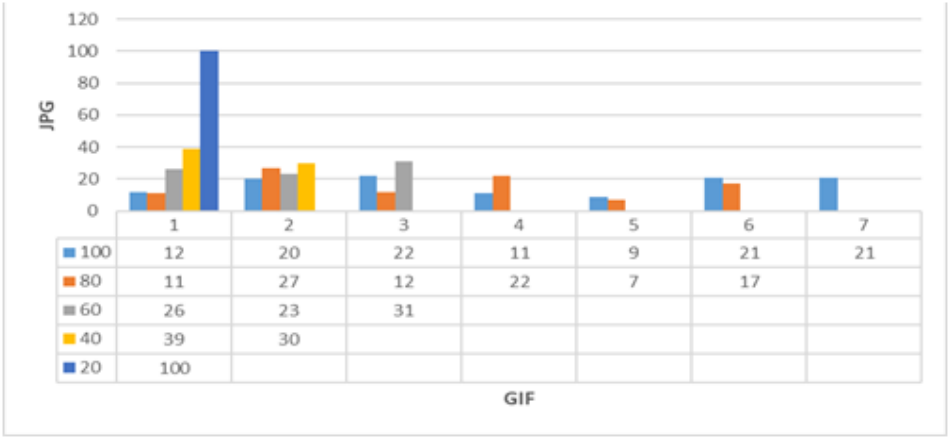

Figure 12. Test 1: Fixed human body with moving arms at 1 meter of distance and using artificial light.

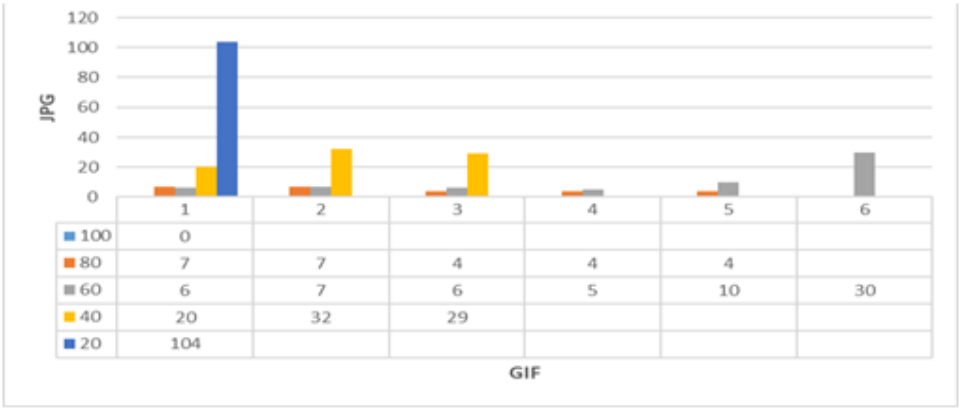

Figure 13. Test 2: Fixed human body with moving arms at 3 meter of distance and using artificial light..

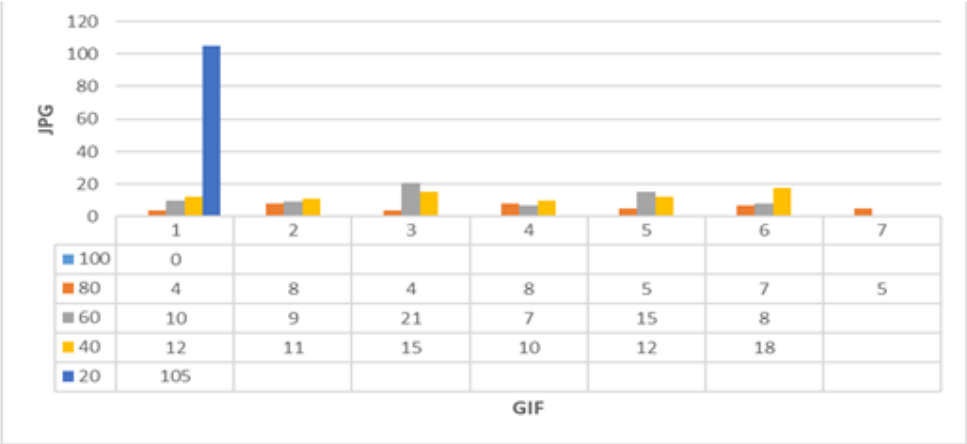

Figure 14. Test 3: fixed human body with moving arms at 1 meter of distance and using natural light.

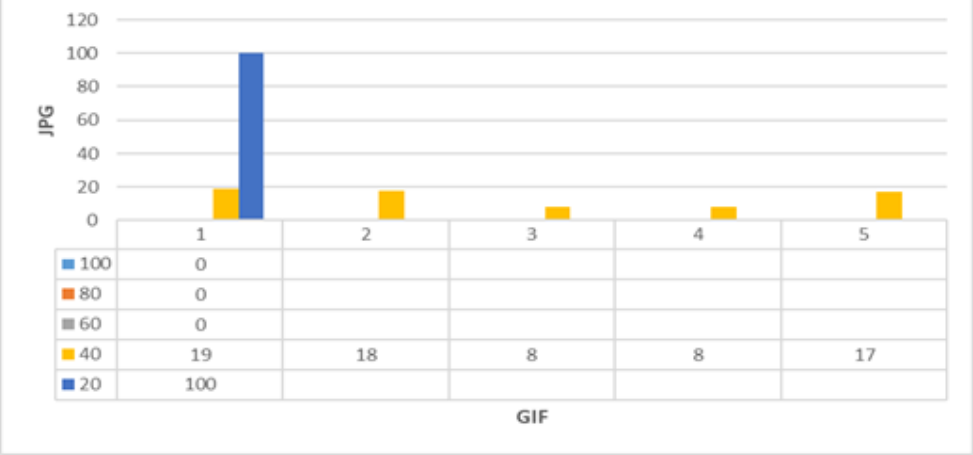

Figure 15: Test 4: fixed human body with moving arms at 3 meter of distance and using natural light.

Figure 16. Test 5,6,7,8: walking man.

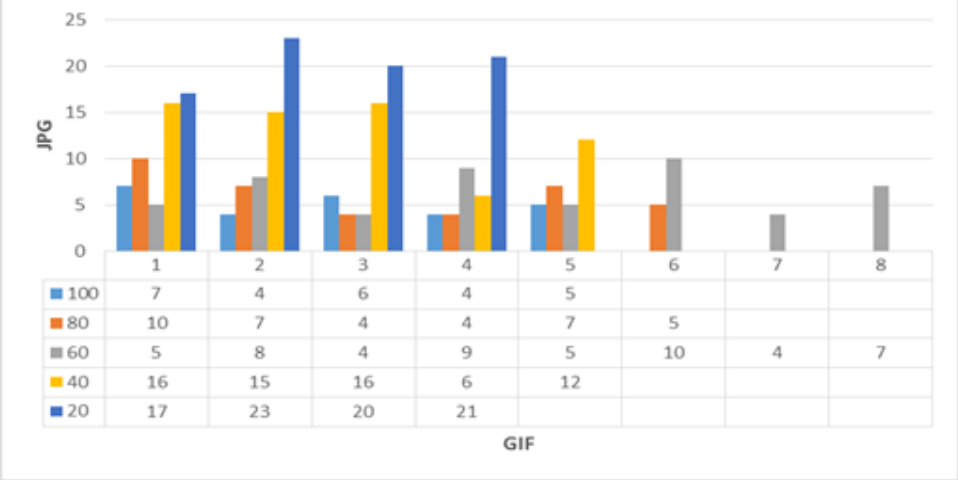

Figure 17. Test 5: Walking man at 1 meter of distance with artificial light.

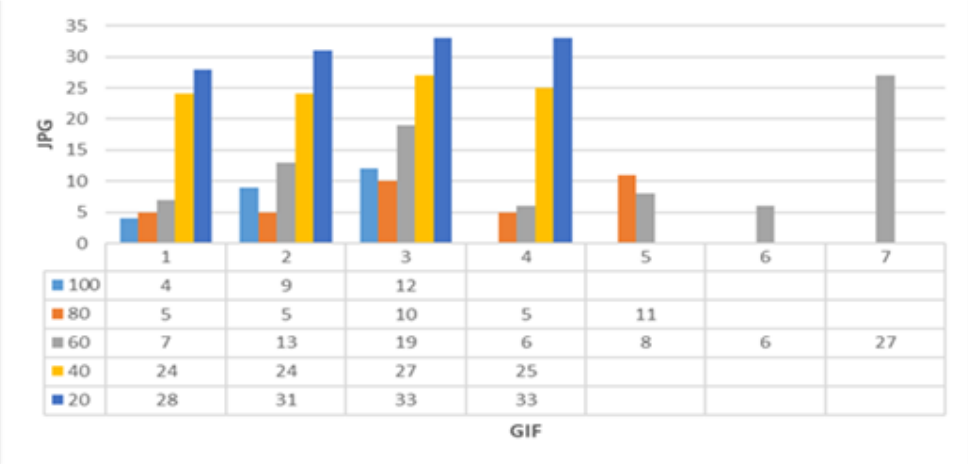

Figure 18. Test 6: Walking man at 3 meter of distance with artificial light.

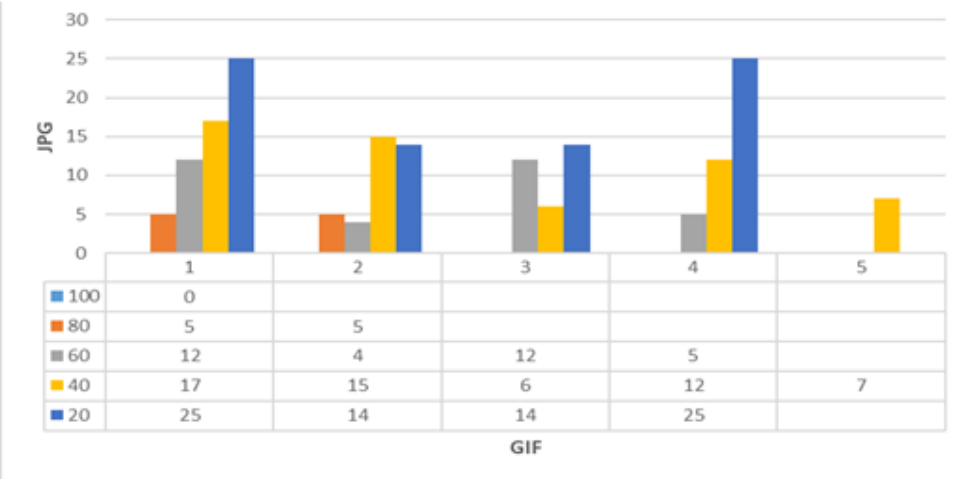

Figure 19. Test 7: Walking man at 1 meter of distance with natural light.

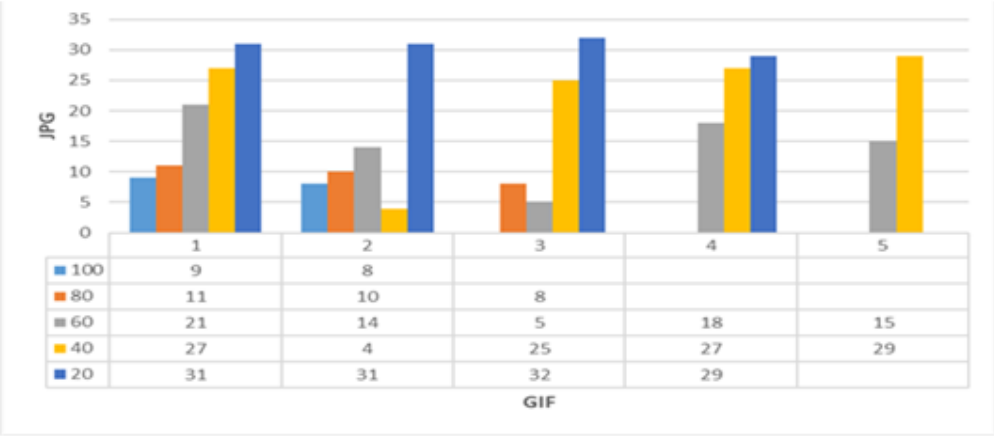

Figure 20. Test 8: Walking man at 3 meter of distance with natural light.

Figure 21. Test 9, 10, 11, 12: running man.

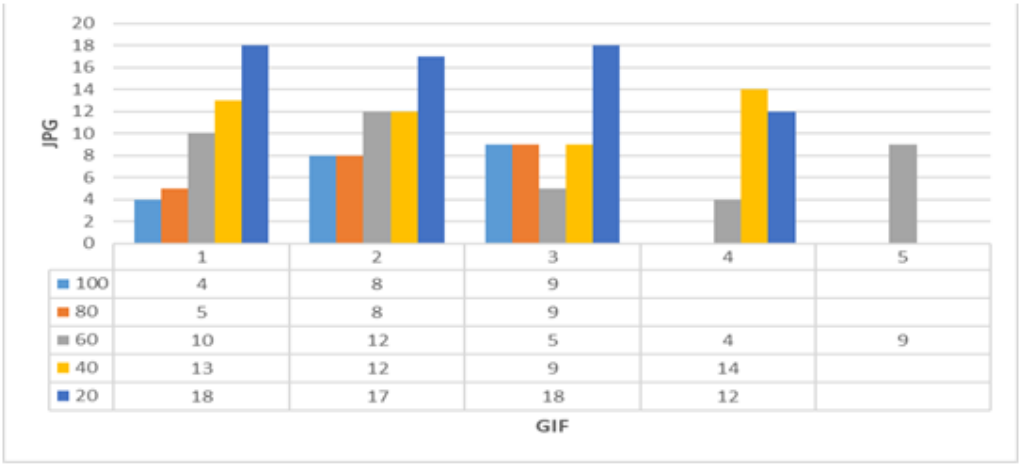

Figure 22. Test 10: Running man at 3 meter of distance with artificial light.

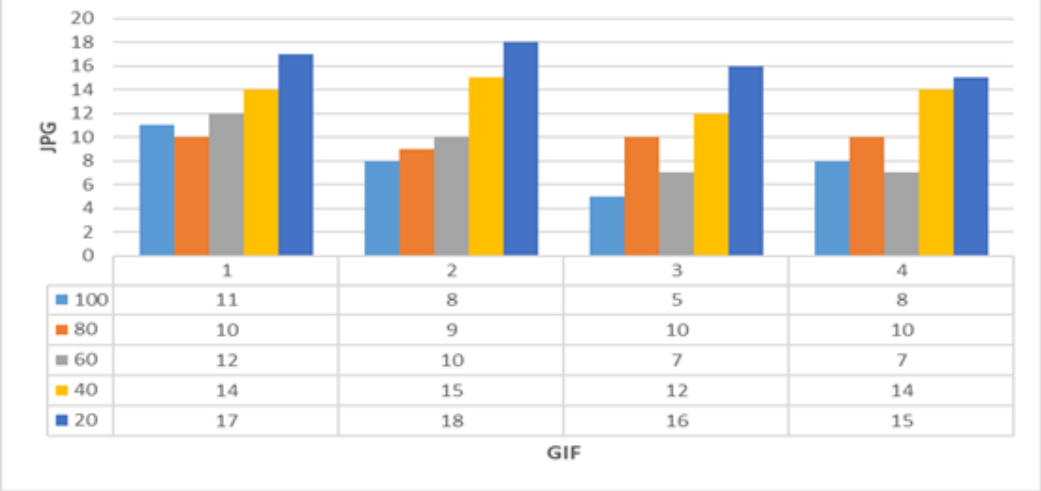

Figure 23. Test 12: Running man at 3 meter of distance with natural light.

Figure 24. Test 13,14: object launched (backpack) simulating running dog or cat.

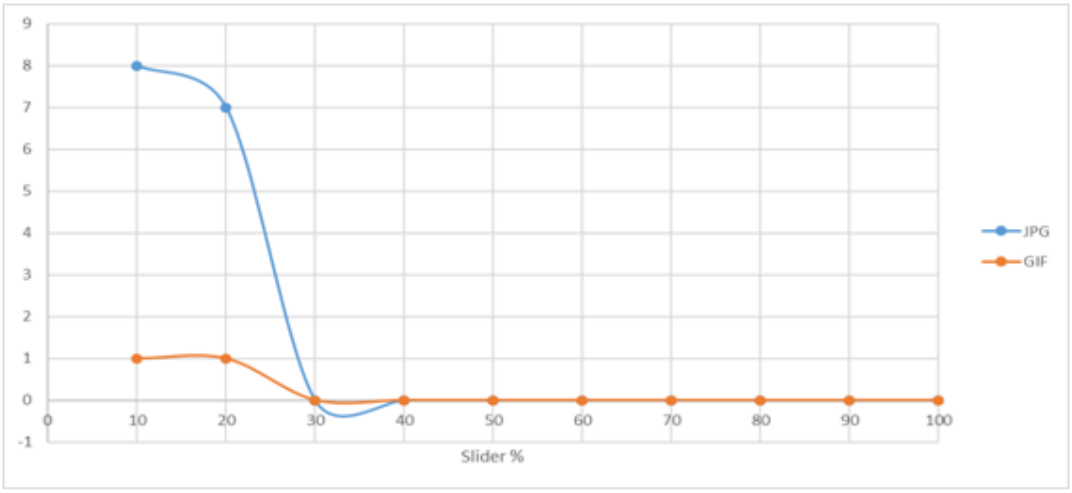

Figure 25. Test 13: object launched (backpack) simulating running dog or cat with artificial light.

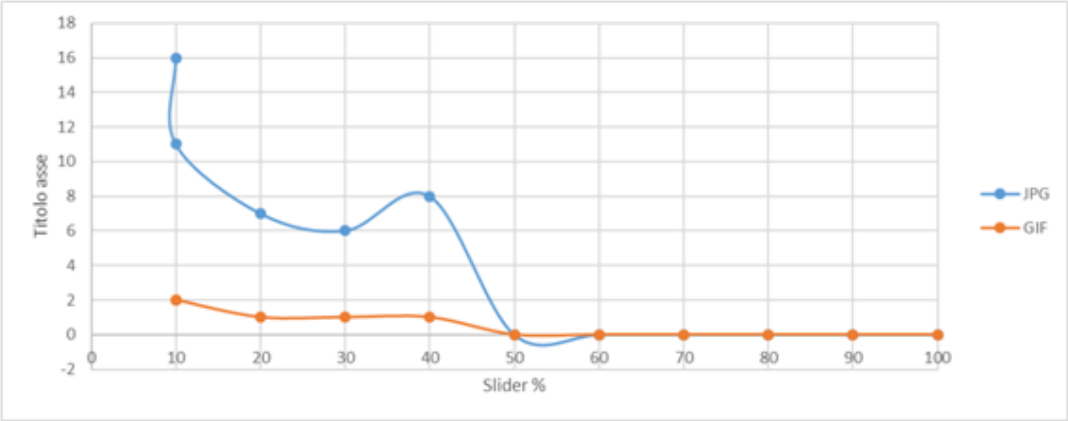

Figure 26. Test 14: object launched (backpack) simulating running dog or cat with natural light.

Table 1. Test summary.

Test	Conditions	Best slider value
1	Fixed human body with moving arms at 1 meter of distance and using artificial light.	60 %
2	Fixed human body with moving arms at 3 meter of distance and using artificial light.	60 %
3	Fixed human body with moving arms at 1 meter of distance and using natural light.	None (false detection)
4	Fixed human body with moving arms at 3 meter of distance and using natural light.	60 %
5	Walking man at 1 meter of distance with artificial light.	20 %
6	Walking man at 3 meter of distance with artificial light.	20 %
7	Walking man at 1 meter of distance with natural light.	20 %
8	Walking man at 3 meter of distance with natural light.	20 %
9	Running man at 1 meter of distance with artificial light.	None (false detection)
10	Running man at 3 meter of distance with artificial light.	20 %, 40 %
11	Running man at 1 meter of distance with natural light.	None (false detection)
12	Running man at 3 meter of distance with natural light.	20 %
13	object launched (backpack) simulating running dog or cat with artificial light	10 %
14	object launched (backpack) simulating running dog or cat with natural light	10 %

5. CONCLUSION

The goal of the proposed paper is the study of motion detection sensitivity results of a prototype video surveillance system developed within the framework of an industrial project. The proposed Logitech HD C720 webcam prototype processor is based on image segmentation and on background subtraction approaches implementing BackgroundSubtractorMOG2 algorithm of OpenCV. Experimental results prove that the setting conditions are related mainly to the typology of moving object to detect and less on light detection condition. A good compromise between an average number of GIF images and an high number of automatically generated JPG ones represent the best condition of no false detections. Experimentations showed that the condition of 1 meter of distance between the webcam and the moving object provides in some cases false motion detection, the walking and the running man are better detected for a slider of 20 %, the motion of an human body part is better detected for a slider of 40 %, and movements of a possible pet are better detected for a slider of 10 %. These results proves that the parameter setting are very important for video surveillance systems. The proposed procedure could be applied for all sensitivity setting procedure involving other detection algorithms and camera hardware. Other hardware and optical lenses could change drastically the best setting parameters. The paper investigates the whole prototype system design by providing important information about the development of the image processing code which could be easily implemented in an industrial smart camera system.

ACKNOWLEDGEMENTS

The work has been developed in the frameworks of the Italian projects (developed for CONSORZIO C.E.P.I.M.): “Studio applicazione e sperimentazione di soluzioni innovative per sistemi di videosorveglianza integrata e sicurezza in remoto con riconoscimento e archiviazione dati: ‘CEPIM SURVEILLANCE SECURITY’ [Study application and experimentation of innovative solutions for video surveillance integrated systems and remote security with recognition and data archiving: "CEPIM SURVEILLANCE SECURITY"]”. The authors would like to thank the following collaborators of the research team: V. Antonacci, G. Birardi, B. Bousshael, R. Cosmo, V. Custodero, L. D’Alessandro, G. Fanelli, M. Le Grottaglie, G. Lofano, L. Maffei, G. Lonigro, A. Lorusso, S. Maggio, N. Malfettone, L. Pellicani, R. Porfido, M. M. Sorbo, F. Tarulli, E. Valenzano.

REFERENCES

[1] Salarpour, A., Salarpour, A., Fathi, M., & Dezfoulian, M. (2011), “Vehicle tracking using Kalman filter and features,” *Signal & Image Processing : An International Journal (SIPIJ)*, Vol. 2, No. 2, pp1-8.

[2] Raad, A. H., Ghazali, S., & Loay, E. G. (2014), “Vehicle detection and tracking techniques: a concise review,” *Signal & Image Processing : An International Journal (SIPIJ)*, Vol. 5, No. 1, pp1-12.

[3] Kishore, D. R., Mohan, M., & Rao, A. A. (2016), “A novel probabilistic based image segmentation model for real time human activity detection,” *Signal & Image Processing : An International Journal (SIPIJ)*, Vol. 7, No. 6, pp11-27.

[4] Sujatha, G. S., & Kumari, V. V. (2016), “An innovative moving object detection and tracking system by using modified region growing algorithm,” *Signal & Image Processing : An International Journal (SIPIJ)*, Vol. 7, No. 2, pp39-55.

[5] El-Taher, H. E., Al-hity, K. M., & Ahmed M. M. (2014) "Design and imnplementation of video tracking system based on camera field of view," *Signal & Image Processing : An International Journal (SIPIJ)*, Vol. 5, No. 2, pp119-129.

[6] Massaro, A., Vitti, V., & Galiano, A. (2018) "Automatic image processing engine oriented on quality control of electronic boards," *Signal & Image Processing: An International Journal (SIPIJ)*, Vol. 9, No. 2, pp1-14.

[7] Bouwmans, T., El Baf, F., & Vachon, B. (2008) "Background modeling using mixture of gaussians for foreground detection-a survey*," Recent Patents on Computer Science*, Vol. 1, No. 3, pp219-237.

[8] Sen-Ching S., & Kamath, C. (2005) "Robust background subtraction with foreground validation for urban traffic video," *Eurasip Journal on applied signal processing*, Vol. 14, pp1-11.

[9] Toyama, K., Krumm, J., Brumitt B., & Meyers, B. (1999) "Wallflower: principles and practice of background maintenance," *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, Vol. 1, pp255-261.

[10] Carranza, J., Theobalt, C., Magnor, M. A. & Seidel, H. P. (2003) "Free-viewpoint video of human actors," *ACM transactions on graphics (TOG)*, Vol. 22, No. 3, pp569-577.

[11] Mikić, I., Trivedi, M., Hunter, E., & Cosman, P. (2003) "Human body model acquisition and tracking using voxel data," *International Journal of Computer Vision*, Vol. 53, No. 3, pp199-223.

[12] Warren, J. (2003) "Unencumbered full body interaction in video games," in *Master's Thesis, Parsons School of Design*, 2003.

[13] Elhabian, S. Y., El-Sayed K. M., & Ahmed, S. H. (2008) "Moving object detection in spatial domain using background removal techniques-state-of-art," *Recent Patents on Computer Science*, Vol. 1, No. 1, pp 32-54.

[14] McFarlane N. J., & Schofield, C. P. (1995) "Segmentation and tracking of piglets in images," *Machine Vision and Applications*, Vol. 8, No. 3, pp187-193.

[15] Zheng, J., Wang, Y., Nihan N., & Hallenbeck, M. (2006) "Extracting roadway background image: Mode-based approach," *Transportation Research Record: Journal of the Transportation Research Board*, Vol. 1944, pp82-88.

[16] Elgammal, A., Harwood, D., & Davis, L. (2000) "Non-parametric model for background subtraction," *Proceeding of European Conference on Computer Vision*, *Berlin, Springer*, pp751-767.

[17] Sigari, M. H., Mozayani, N., & Pourreza, H. (2008) "Fuzzy running average and fuzzy background subtraction: concepts and application," *International Journal of Computer Science and Network Security*, Vol. 8, No. 2, pp138-143.

[18] Chang, R., Gandhi, T. & Trivedi, M. M. (2004) "Vision modules for a multi-sensory bridge monitoring approach," *Proceeding of The 7th International IEEE Conference on Intelligent Transportation Systems*, pp971-976.

[19] Wang H. & Suter, D. (2006) "A novel robust statistical method for background initialization and visual surveillance," *Proceeding of Asian Conference on Computer Vision, Berlin, Springer*, pp 328-337.

[20] Porikli F., & Tuzel, O. (2003) "Human body tracking by adaptive background models and mean-shift analysis," *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pp1-9.

[21] Porikli F., & Tuzel, O. (2005) "Bayesian background modeling for foreground detection," *Proceedings of the third ACM international workshop on Video surveillance & sensor networks*, pp55-58.

[22] "cv::BackgroundSubtractorMOG2 Class Reference" 2018. [Online]. Available: https://docs.opencv.org/3.4.0/d7/d7b/classcv_1_1BackgroundSubtractorMOG2.html

[23] Wren, C. R., Azarbayejani, A., Darrell, T. & Pentland, A. P. (1997) "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, pp780-785.

[24] Stauffer C., & Grimson, W. E. L. (1999) "Adaptive background mixture models for real-time tracking," *Proceeding of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp246-252.

[25] Eveland, C., Konolige K., & Bolles, R. C., «Background modeling for segmentation of video-rate stereo sequences,» IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 266-271, 1998.

[26] Gordon, G., Darrell, T., Harville, M., & Woodfill, J. (1999) "Background estimation and removal based on range and color," *Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp459-464.

[27] "UMLet 14.2 Free UML Tool for Fast UML Diagrams" 2018. [Online]. Available: http://www.umlet.com/

Corresponding Author

Alessandro Massaro: Research & Development Chief of Dyrecta Lab s.r.l.