REVISION: A System for Motion Detection in Real-Time Video Streams

D.J.S. Sako[1], E.O. Bennett[1], C.G. Igiri[1], F.B. Deedam-Okuchaba[1], P. Obe[1]
[1]Department of Computer Science,
Rivers State University, Port Harcourt, Nigeria.

DOI: 10.56201/ijcsmt.v8.no1.2022.pg31.44

Abstract

This paper discusses the implementation of REVISION, a GUI-enabled security and surveillance system that detects motion in video streams received from attached webcams in real time. REVISION has the ability to automatically analyze the video images, alert and archive the images when motion is detected from the view of the camera. This provides relief to the normal video surveillance system which continuously records the situations even when there is nothing happening in front of the camera; a situation that results in consumption of large amount of memory and a time-consuming reviewing process requiring more manpower and human resources. The system is designed and modelled using Object-Oriented analysis and design methodology, and implemented using Java programming language. The initial testing of the system shows that it worked as expected.

Keywords: *Image Processing, Threshold, CCTV, Surveillance, Motion detection*

I. INTRODUCTION

There has been many researches on visual surveillance in the past decades. Motion detection and tracking from video imagery has been an active field in these researches. Although motion detection is a challenging field in computer vision, it has increased social interests of people via its application in human computer interface, surveillance systems, security and many other areas of human endeavors. Videos are actually sequences of images, each of which called a frame, displayed in fast enough frequency so that human eyes can percept the continuity of its content. Object detection in videos involves verifying the presence of an object in the image sequences and possibly locating it precisely for recognition. Object tracking is to monitor an object's spatial and temporal changes during a video sequence, including its presence, position, size, shape, etc. (Guo, 2001).

Video surveillance has extensively been in use to observe security sensitive areas such as banks, (Upasana, etal 2015), department stores, highways, busy public places and borders (Vidyashree and Rao, 2017). In view of safety and protection, there has been installations of thousands of surveillance cameras in the world, which operations, recorded continuously, are being collected and processed by human operators slowly. Although many local image

processing functions are possible to improve the system application, this requires a lot of processing resources and high-power-consuming hardware. Hence there is a need to develop an automated surveillance system that only records when there is something happening in front of the camera. Such recordings are done in form of taking snapshots in milliseconds of every event of detected motion.

The purpose of motion detection is to provide an automatic detection in the region of interest. This region is always embodied in a region of awareness or in terms of the camera geometry, the field of view. The region of interest in the present case is the environment with moving object and activity.

Digital video processing combines streaming video with image processing to identify and track a moving target, provide autonomous vehicle guidance and detect motion in security applications (Gregg, 2018). Object motion detection is becoming a demanding application in our day today (Hampapur et al., 2004). Surveillance is required for intelligent gathering, the prevention of crimes, protection of processes, persons, groups, or objects for the investigation of crimes. (Benfold and Reid, 2009), These are achieved through deterrence, observation and reconstruction which are the core objectives of a typical surveillance system. Surveillance can assist rebuilding of an incidence through the availability of footage for forensics experts. It can also be used for personal security.

This paper discusses the design and implementation of REVISION, a system for motion detection in video streams received from attached webcams in real time, using enhanced motion detection algorithm, with the ability to automatically analyze the video images and archive the images with moving objects and also alert automatically when motion is detected from the view of camera. This implies that our proposed system gives the camera the capability to capture when needed rather than capture all the time and this leads to huge reduction in storage space and monitoring manpower. The system uses the concept of "Motion Detection by Vision" to achieve motion detection capability where no hardware sensors are required.

In the next section, we discuss some related work. We present the approaches and methods used in this study in section 3. We describe the implementation of our proposed system in section 4. Finally, we conclude in section 5.

II. RELATED WORK

A sequence of methods and systems for detection of motion have been designed and developed in the past decades. Haritaoglu, et al., (2000) focused on Scene Modeling and Maintenance for Outdoor Surveillance using fast background subtraction. The distinguished part of that work was that it was based on unlabeled data, i.e., the training features set contains both useful foreground parts and background clutter and the correspondence between the parts and detected features were unknown. Unfortunately, the method does not work very well due to the highest degree of variability that a pedestrian can have depending on clothes and pose he was assuming.

Shan and Wang (2006) described algorithms for motion detection and tracking, respectively, in a video surveillance system. Firstly, the shadow and ghost detection processing is fused with updating the background subtraction model, after the frame is transformed to HSV color space from RGB color space in order to reduce the effect of illumination changes, shadows,

and ghosts. And the result of detection is used as initialization for tracking. Secondly, the gradient field is combined with region information to locate the boundary of the object accurately, instead of the traditional level-set method, which only utilizes the gradient field to propagate a front.

Jodoin, et al (2008) introduced accurate and fast motion detection in the existence of camera jitter by applying background subtraction to video scene dynamics as an alternative of scene photometry. In their work, an object is assumed moving if its dynamical behavior is unlike from the standard dynamics observed in a reference sequence. Evolution in the field of object's motion detection is generated by the introduction of the Weizmann challenge (Gorelick et al, 2007) that presented an object detection challenge, where the proposed methods are applied on a large dataset of 9 different object categories performing 10 natural actions. An important step forward for enhancing detection accuracy is the work of Vikas et al., (2013), where the authors proposed a block-based method capable of dealing with noise, dynamic backgrounds, illumination variations and while still obtaining smooth curve of foreground objects. Specifically, image sequences are analyzed on an overlapping segment-by-segment basis. A low-dimensional texture descriptor obtained from each block is passed through an adaptive cascade classifier, where every stage deals with a distinct problem. A probabilistic foreground mask generation approach then used blocks overlaps to incorporate interim block-level decisions into final pixel-level foreground segmentation. Lu et al (2008) proposed a novel real time motion detection algorithm that integrated the temporal differencing method, optical flow method, double background filtering (DBF) method and morphological processing methods to achieve better performance. The effectiveness of the proposed algorithm for motion detection was demonstrated in a simulation environment.

Kavitha, and Tejaswini (2012) presented a technique for motion detection that incorporates several innovative mechanisms that store, for each pixel, a set of values taken in the past at the same location or in the neighborhood. It then compares this set to the current pixel value in order to determine whether that pixel belongs to the background, and adapts the model by choosing randomly which values to substitute from the background model.

Iyapo et al (2018) built a motion detection alarm and security system using an embedded microcontroller system capable of detecting motion of an intruder in a restricted area and then triggering an alarm system. Passive infrared sensor detected the motion of the person using the person body heat. The passive infrared (PIR) sensor which is the motion detector is attached to a microcontroller which activates the alarm system and any other attached output device to notify the house owner. SuganyaDevi et al (2013) presented motion detection in video frames using self-adaptive background frame Matching method to select the background pixel at each frame with regard to background model generation with moving camera. Two images frames are required to detect any movement. The first frame, called reference frame, represents the reference frame values for comparison purpose, and the second frame, called the input frame, contains the moving object. The two frames are compared and the differences in pixel values are determined. Singha (2014) presented a new algorithm for detecting moving objects from a static background scene based on frame difference. Firstly, the first frame is captured through the static camera and after that sequence of frames is captured at regular intervals. Secondly, the absolute difference is calculated between the consecutive frames and the difference image is stored in the system. Thirdly, the difference image is converted into gray image and then translated into binary image. Finally, morphological filtering is done to remove noise.

III. PROPOSED SYSTEM

The proposed system works in the following ways:

i. **Capture images:** Before a motion is detected, live images have to be captured with areas that are monitored and kept under surveillance. This can be achieved by using a webcam (WC) which continuously provides sequence of images in a particular speed of frames per second.
ii. **Compare frames:** The current frames captured are being compared with previous frames to detect motion. That is, comparing the live images being provided by the webcam with each other so as to detect changes in these frames and hence predict the occurrence of some motion.
iii. **Store images:** When motions are being detected in images, the images are then stored in memory so that the user can view it in the near future. It helps the user in providing proof of some inappropriate activity.
iv. **Trigger alarm:** The system triggers an alarm, according to the user settings, to notify the user that a motion has been detected. The alarm continues until the webcam losses the detected motion or the motion stops.

The proposed system pipeline is shown in figure 1 Is

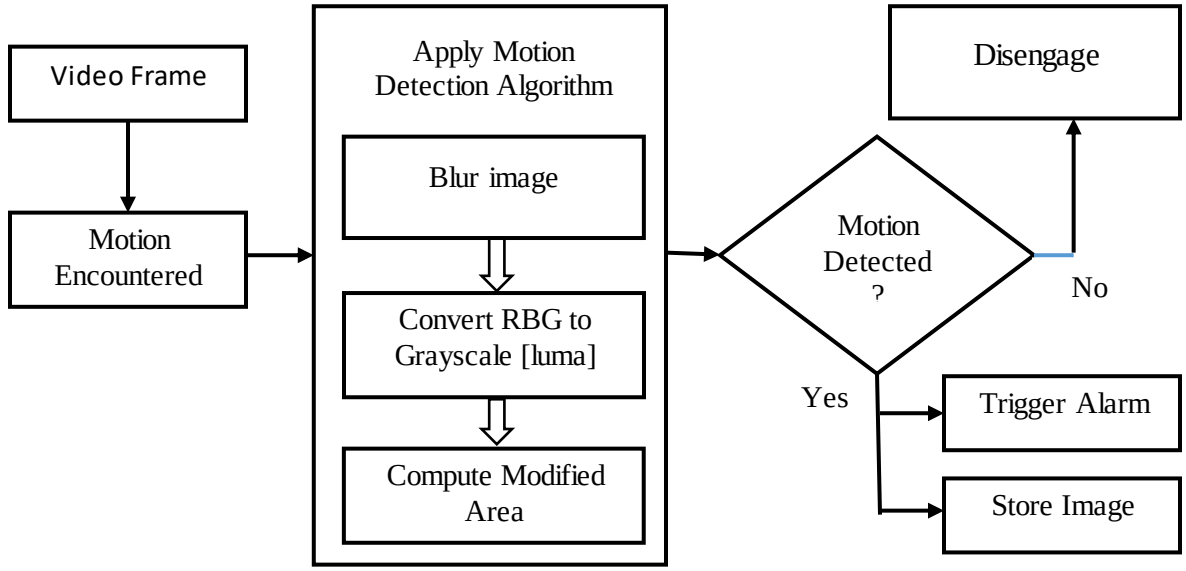

Figure 1: Proposed system motion detection pipeline

A. Approaches to Motion Detection

The following fundamental steps, as outlined by Firyn (2019) and adopted in this work, describe how the proposed system will be able to achieve its aim of motion detection using web camera.

Blur Image

Image blurring is performed to reduce noise by implementing a filter which performs a box blur on the image. Box blur is based on the moving window in which average values of every channel in Red, Green, Blue (RGB) space are being calculated. Implementing a 3x3 box blur.on the image in which there are 9 pixels by considering every neighboring pixel of the Pixel(x,y) in an image, as shown in figure 2, using algorithm 1 for image blurring, will produce:

Sum = image[x - 1, y + 1] + image[x + 0, y + 1] + image[x + 1, y + 1] + image[x - 1, y + 0] + image[x + 0, y + 0] + image[x + 1, y + 0] + image[x - 1, y - 1] + image[x + 0, y - 1] + image[x + 1, y - 1].

The sum of the colors in figure 2 will be:
sum = image [3,5] + image[4,5] + image [5,5] + image[3,4] + image[4,4] + image[5,4] + image[3,3] + image[4,3] + image[5,3]

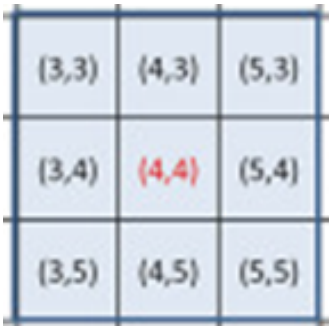

Figure 2: Neighbour pixels of P(4,4)

Algorithm 1: **Image Blurring**

```
Input: Image
Output: newImage
Box_blur(Image)
{
        set newImage to image
        set sum, count to 0
        obtain values for Width and Height of image
        for x /*row*/, y/*column*/ on newImage
        {
                //Kernel would not fit!
                If x < 1 or y < 1 or x + 1 == Width or y + 1 == Height then
                        continue
                for every neighboring pixel of the Pixel(x,y) in the image within
                the radius, r.
                {
                        Color c=image[x,y]  // image .getpixel(x, y)
                        //add the color to total
                        sum += color
                        count++
                }
                //Set the final color, newImage[x,y], to the average of n pixels
                newImage[x,y] = sum/count
                setPixel(current x, current y, newImage[x,y])
        }
Return newImage
}
```

Convert Pixels Gray Scale

Converting RGB data from every pixel to grayscale is done so that work can be done on luminance data as it is much easier to do conversion from RGB to luminance. As indicated by Firyn (2019) we will need to perform motion detection for every channel separately and then average the output, which is generally more complex if we stay within RGB. The following luma equation weighs red, green and blue according to their wavelengths and is used to convert RGB to luma,(grayscale) defined in sRGB specification (Poynton, 2012).

$$Y = 0.299R + 0.587G + 0.114B \quad (1)$$

Original red, green, and blue values are replaced with the new gray scale value, Y, as shown in Algorithm 2.

Algorithm 2: **Gray Scale Algorithm**

Input*: image Img*
Output: grayscale image, GrayImg

```
//Get each pixel pixel(x,y) in image
For each x /*row/, y/*column/ on Img
{
        Color c=Img.getpixel(x,y)
        //Extract red (R), green (G), and blue (B) from color c
        R = c.Red
        G = c.Green
        B = c.Blue
        Y =0.299R + 0.587G + 0.114B
        Img.setPixel(x,y)=new Color(Y)
}
GrayImg=img;
Return GrayImg
```

Calculate Modified Area

We compare all pixels with the previous values (from previous image) to calculate how many of these pixels are different. We increase the number of different pixels by one if pixel intensity difference threshold is greater than a given pixel threshold; in which case a pixel is deemed to be different. We set the configurable value of this pixel intensity difference threshold (pixel threshold) above which pixel is classified as "moved". Pixel threshold ranges from 0 (minimum) to 255 (maximum), with default set to 25. We add the combined pixels of the current modified image and the previous modified to thresholds. At the end, we calculate percentage value of modified area by using formula defined as:

$$area = count * 100 / (width * height) \quad (2)$$

where *count* is the number of modified pixels (with luma above pixel threshold), *width* is the width of the image width and *height* is the height of the image.

Decide if there is Motion

The percentage value of modified area is compared with a configured value representing the percentage fraction of detected motion area threshold; the percentage threshold of image that has different pixels for motion to be detected and above which the image is classified as "moved". Motion detection is fired and motion detector is engaged. Minimum value for this is 0 and maximum is 100, which corresponds to full image covered by spontaneous motion. The default value is 20% (i.e. 0.2). A lower percentage area value means that there is no motion and motion detector is disengaged. The modified area is also compared with the maximum percentage fraction of detected motion area threshold; below which it is classified as "moved". The maximum is optionally used to help filter false positives caused by sudden exposure changes. Motion strength is defined as: 0 = no motion, 100 = full image covered by motion; hence it is set to 100.

Algorithm 3: **Proposed Motion Detection Method**

```
Input: Input video (live streams from webcam)
Output: Images
Begin
    Initialize x=0, y=0, count=0, Thresholds, pixelThreshold, areaThreshold,
    areaThresholdMax, imageWidth, imageHeight
    /*
    count: the number of modified pixels (with luma above threshold),
    pixelThreshold: an integer value between 0 - 255, default set to 25.
    areaThreshold: value between 0-100, default set to 0.2.
    areaThresholdMax: value between 0-100, default set to 100.
    */
    Take live streams from webcam
    Arrange streams in BufferedImage
    Blur image to reduce noise
    Convert frames into gray scale using (1)
    Loop whle x < imageWidth
        Loop while y < imageHeight
            Get CombinedPixels(x, y)
            Set pixelThreshold = 25
            If CombinedPixels >= pixelThreshold Then
                Add CombinedPixels to Thresholds
                Increment count   // number of different pixels
            Increment y
        End Loop
        Increment x
    End Loop
    Calculate modifiedArea using (2)
    Set areaThreshold to 0.2
    Set areaThresholdMax to 100
    If (modifiedArea >= areaThreshold AND
    modifedArea <= areaThresholdMax) then
        Motion is Detected
        Fire Alarm
        Save Captured Image
    else
```

Disengage Motion Detection
End

The proposed motion detection algorithm of the system, Algorithm 3**,** is derived from the above sub-algorithms and steps which picture each phase of engagement and processes of the image frames.

B. System Modelling

Figure 3 shows the class diagram of the proposed system, with each class showing its attributes and methods. The use case diagram, shown in figure 4, illustrates in a very simple way the main functions of the system and the different kinds of users that will interact with it. For the proposed system, the Actors involved are the User and the Camera.

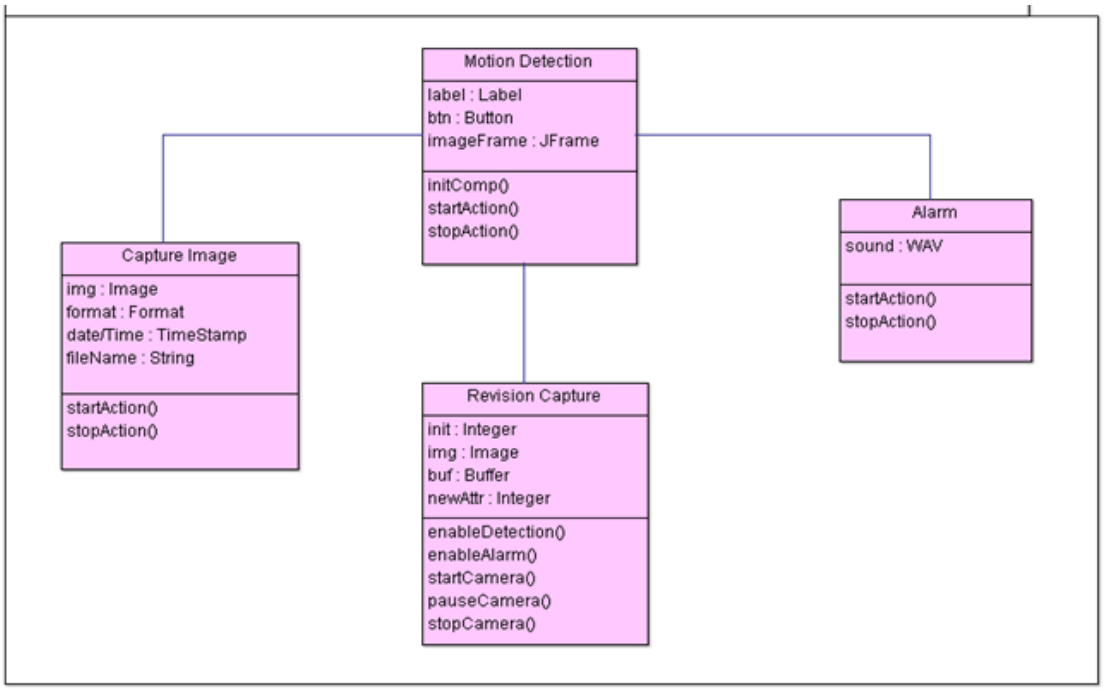

Figure 3: Class diagram of the Proposed System

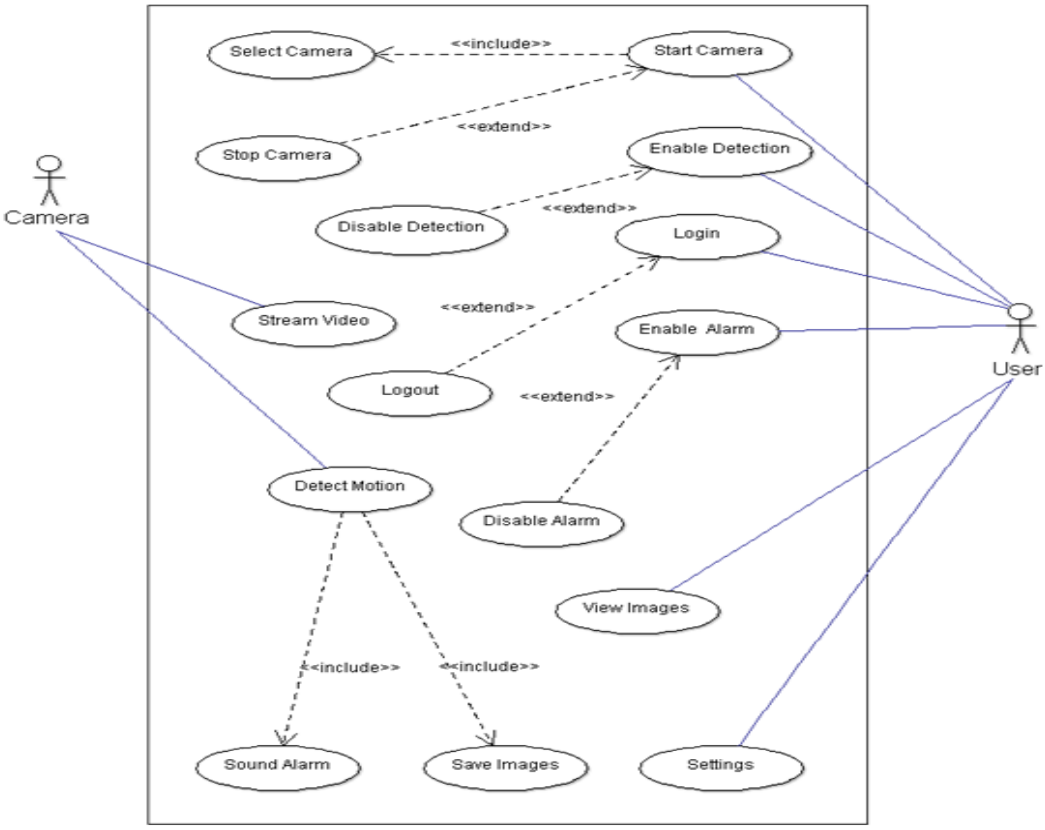

Figure 4: Use Case Diagram of the Proposed System

The sequence diagrams for login, motion detection, saving images and alarm alert are presented in figures 5a, 5b, 5c and 5d respectively. Figure 6 shows the behavioral state machine of the proposed system. It shows the different states that a single object in the system passes through during its life in response to events, along with its responses and actions.

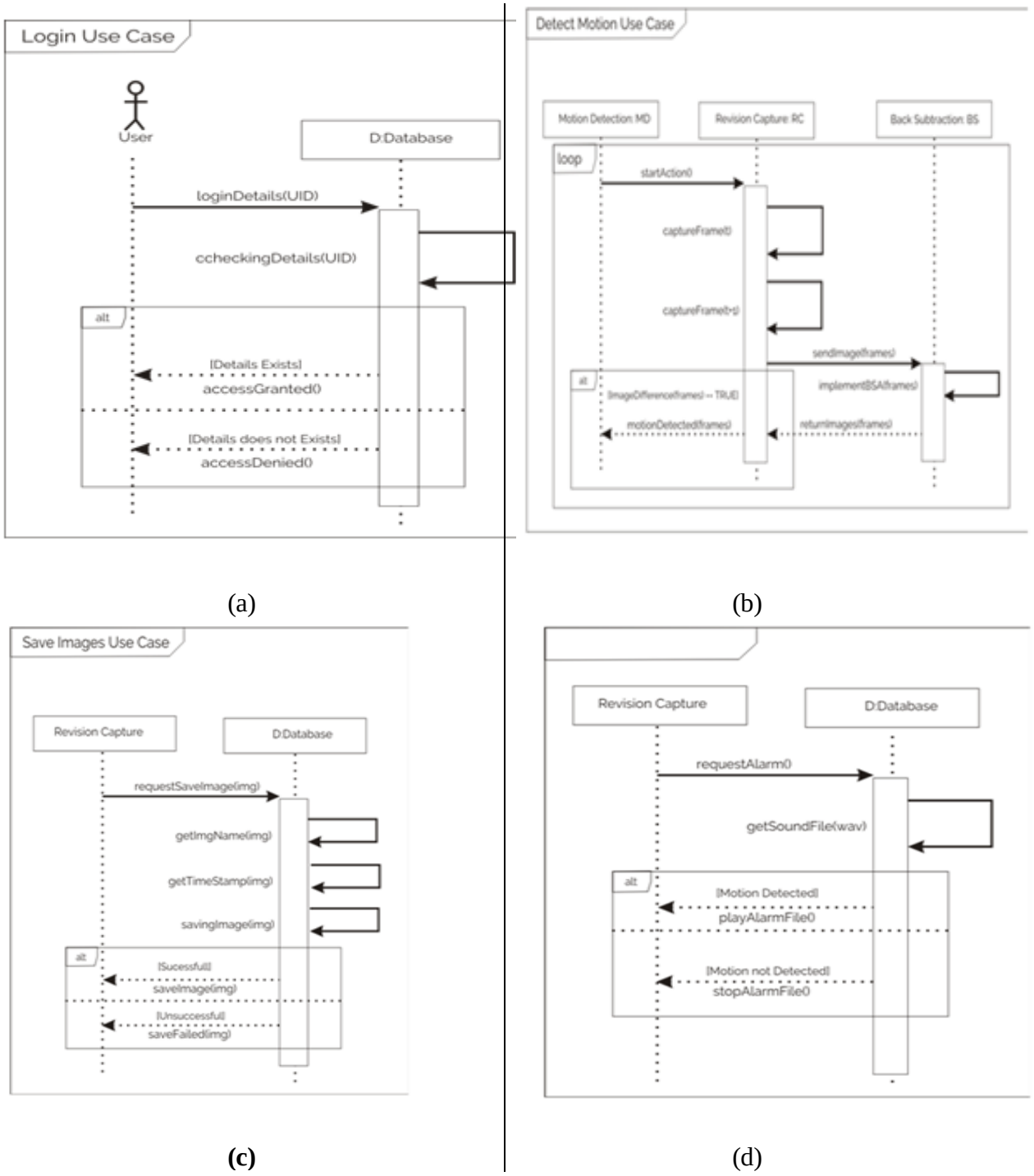

Figure 5: Sequence Diagram for the Proposed System
(a) Login (b) Motion Detection (c) Save Image (d) alarm sequence

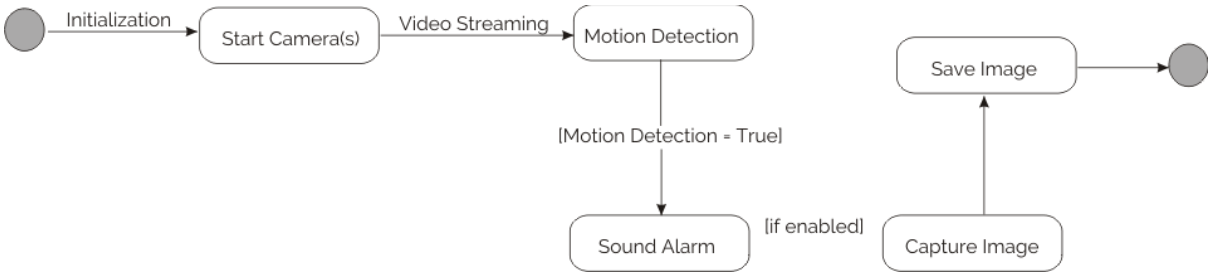

Figure 6: State Machine Diagram of the Proposed System

The Interface of the proposed system is designed using Java Swing Builder in Netbeans IDE and shown in figure 7.

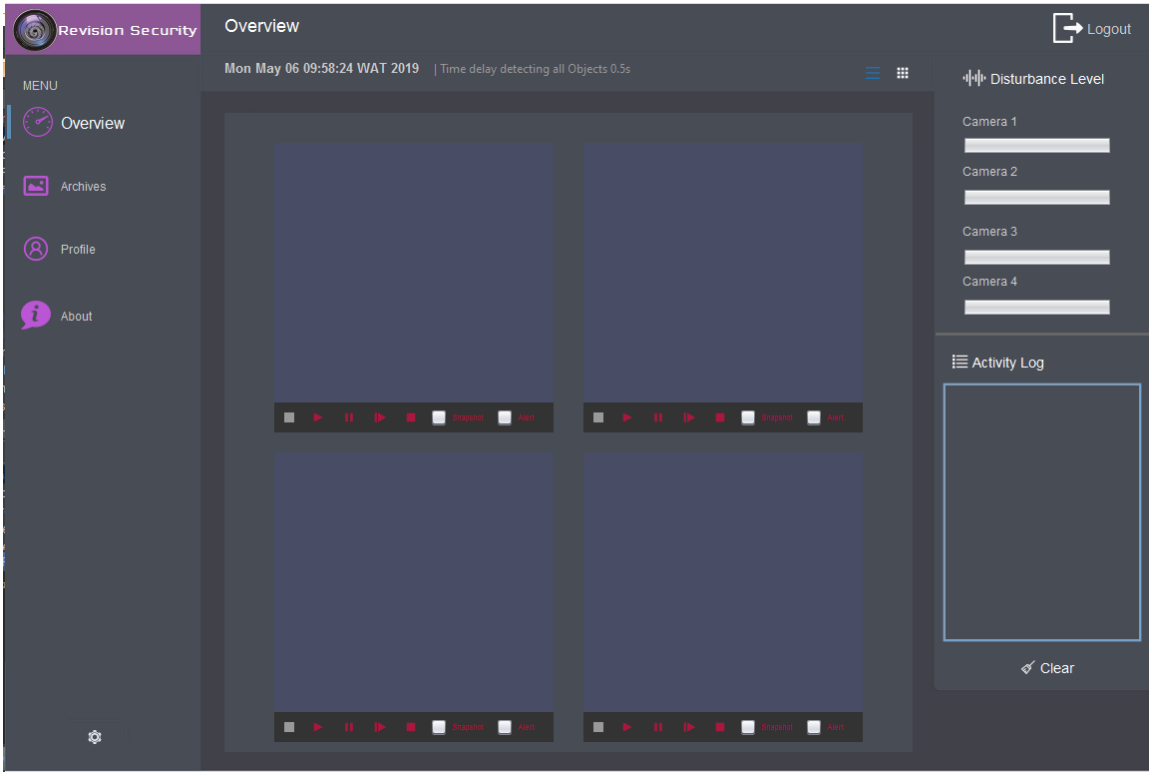

Figure 7: The System's User Interface Design

IV. IMPLEMENTATION

The system is implemented using Java Programming Language. The dashboard as shown in figure 8 contains four panels, 2 of which are assigned to one camera. Each of the panels contains control buttons to turn on the camera, pause the camera, stop the camera, take a snapshot and trigger alarm when a motion is detected. Any images captured when a motion is detected and being saved into the system gallery which can be viewed later by the user.

Figure 9 shows the gallery that contains all saved images, with navigation buttons for viewing previous, next image or even deleting current image.

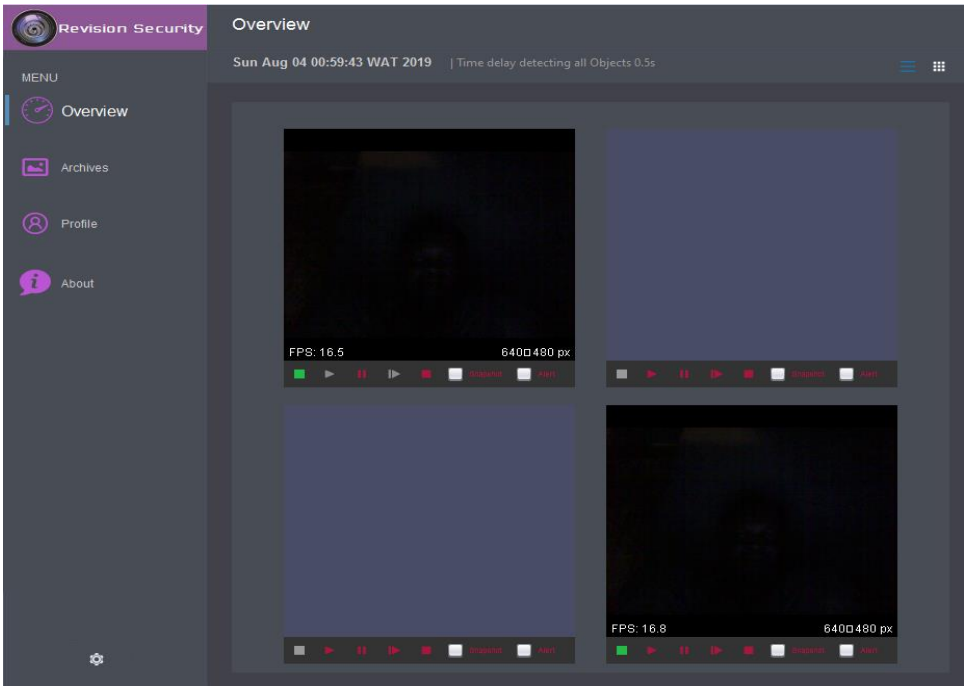

Figure 8: Display on Camera

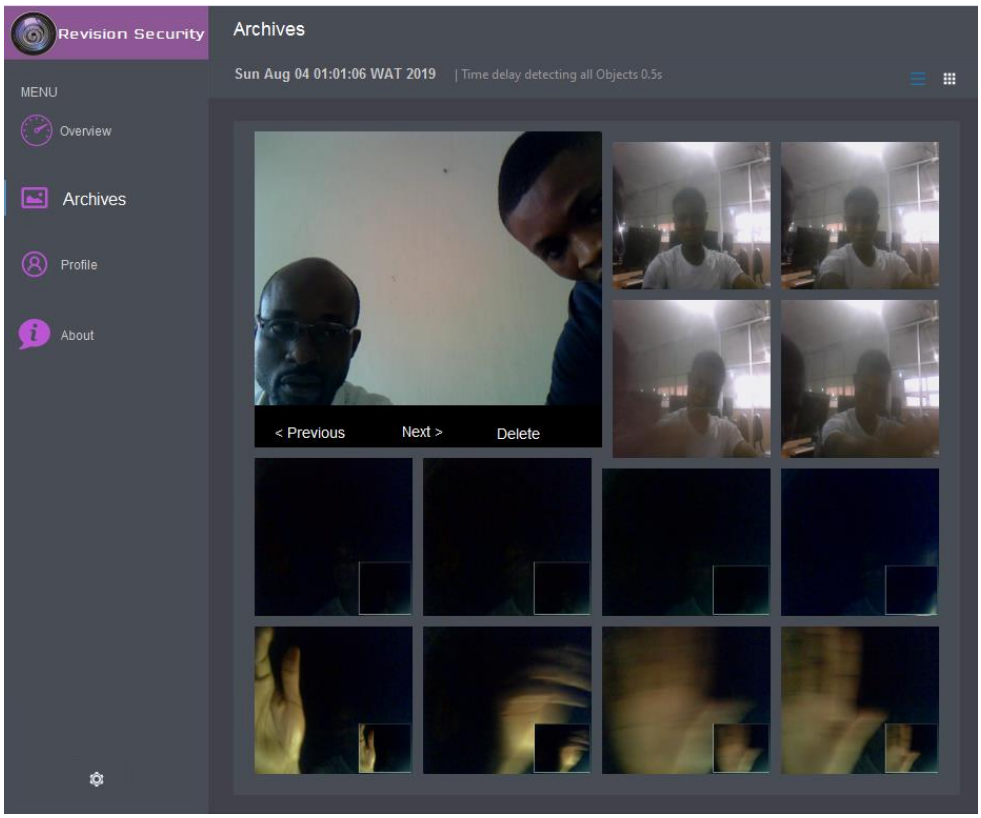

Figure 9: Gallery of Stored Images

IV. CONCLUSION

We have presented the implementation of the proposed REVISION system in which moving objects in a video scene are detected in real time by the method of motion detection. The full implementation of this system is an application built with Java Programming Language with full motion detection capability using connected webcams. The detected images are saved into the system and can be viewed by the user on a later time. The system also triggers alarm once a motion is detected. Automatic motion detection capability helps to reduce archive space resulting from continuous recording and monitoring of the security system by a human. However, the system can be further enhanced by implementing it using IP cameras from remote locations.

REFERENCES

Benfold, B. and Reid, I. (2009) Guiding visual surveillance by tracking human attention. *Proceedings of the British Machine Vision Conference (BMVC)*, 1-11.

Firyn, B. (2019). On what Image Processing Algorithm is WebCam Capture implemented? https://github.com/sarxos/webcam-capture/issues/714

Gorelick, L., Blank, M., Shechtman E, Irani M., and Ronen B. (2005), Actions as Space-Time Shapes, *IEEE International Conference on Computer Vision (ICCV)*.

Gregg, R.L. (2018). Real-Time Streaming Video and Image Processing on Inexpensive Hardware with Low Latency. https://digitalcommons.unl.edu/elecengtheses/93

Guo, Z. (2001). Object Detection and Tracking in Video. http://medianet.kent.edu/surveys/IAD01F-objdetection/index.html

Hampapur, A, Brown, L., Connell, J., Lu, M., Merkl, H., Pankanti, S., Senior, A., and Shu, C. (2004) The IBM smart surveillance system, demonstration, Proc. IEEE, CVPR.

Hare, J.S., Samangooei, S., and Dupplaw, D.P. (2011). OpenIMAJ and ImageTerrier: Java libraries and tools for scalable multimedia analysis and indexing of images. *Proceedings of the 19th ACM International Conference on Multimedia*. 691–694

Haritaoglu, I., Harwood, D., and Davis, L.S. (2000), A Fast Background Scene Modeling and Maintenance for Outdoor Surveillance. *Proceedings of the 15th International Conference on Pattern Recognition (ICPR)*, Barcelona. 179-183.

Iyapo, K.O., Fasunla, O.M., Egbuwalo, S.A., Akinbobola, A.J. and Oni, O.T. (2018). Design and implementation of motion detection alarm and Security system. *International Journal of Engineering and Advanced Technology Studies*. 6(1), 26-38.

Jodoin, P., Konrad, J., Saligrama, V. and Veilleux-Gaboury, V.(2008). Motion Detection with an Unstable Camera. *Proceedings of the IEEE International Conference on Image Processing, ICIP, San Diego, California,* 229–232.

Kavitha, K. and Tejaswini, A. (2012). Background Detection and Subtraction for Image Sequences in Video, *International Journal of Computer Science and Information Technologies*, 3(5), 5223-5226.

Lu, N., Wang, J., Wu, Q.H. and Yang, L. (2008). An Improved Motion Detection Method for Real-Time Surveillance, *IAENG International Journal of Computer Science*, 1(6).

Mazumdar, A. (2013). Understanding box blur. http://amritamaz.net/blog/understanding-box-blur

Poynton, C. (2012). Digital Video and HDTV: Algorithms and Interfaces. 2nd Edition. Morgan–Kaufmann Publishers.

Shan, Y, and Wang, R. (2006). Improved algorithms for Motion Detection and Tracking. *Optical Engineering*, 45(6). http://dx.doi.org/10.1117/1.2213962

Singla, N. (2014). Motion Detection Based on Frame Difference Method. *International Journal of Information & Computation Technology*. 4(15), 1559-1565.

SuganyaDevi, K., Malmurugan, N. and Manikandan, M. (2013). Object Motion Detection in Video Frames Using Background Frame Matching, *International Journal of Computer Trends and Technology*, 4(6), 1928-1931.

Upasana, A., Manisha, B., Mohini, G. and Pradnya, K. (2015). Real Time Security Using Human Motion Detection. *International Journal of Computer Science and Mobile Computing* 4(11), 245-250.

Vidyashree H M and Rao, G.R (2017). A Survey on Motion Detection, Tracking and Classification for Automated Video Surveillance. *International Journal of Engineering Research & Technology (IJERT),* 5(22), 1-4.

Vikas R., Conrad S., and Brian C. L. (2013), Improved Foreground Detection via Blockbased Classifier Cascade with Probabilistic Decision Integration, *IEEE Transactions on Circuits and Systems for Video Technology*, 23(1), 83–93.