Open Source Video Analysis Tool for Motion Detection

Y.N. Prajapati, M.K. Srivastava, Sandhya Sharma

***Abstract*— We present a fast moving Open Source detection application by extending the functionality of open source tools that are available freely on the Internet. This application can be placed on a cloud infrastructure and performs fast processing so that the costs needed to use the cloud resources can be minimized.**

Keywords: - Open source, Motion Detection, Video Analysis

I. INTRODUCTION

Open source video motion detection is a resource intensive process that may require high-speed processors, large primary and secondary memory space, and massive electrical power, to execute the process. It has been widely used for security video surveillance and video scene classification applications, where those applications are usually installed in specific purpose devices.

Cloud-computing allows Open Source video motion detection to be performed with cheaper price because computing resources can be shared by many users. A video surveillance application can be installed in an IaaS facility, which then can be virtually used by users based on the computing resources usage. For example, the application can be installed in the Google Compute Engine and the video data can be uploaded and stored in the RESTful (Google Storage Service).

In order to implement open source video motion detection in lesser price, we use open source tool those are completely free. One of such tools is Open CV [1], have libraries used for image processing those are used for motion detection. Open CVs libraries, analysis every pixel of every image of video so that moving object pr movement within the video scene can be tracked.

However, analyzing pixel values, particularly for high-resolution video data, is time consuming. Although cloud servers may provide the computing powers to resolve the computing requirements, it is possible to reduce the processing time to detect moving object by using video compression features because: first, most of modern video cameras now compress recorded video data prior to storage or transmission; second, we believe that the video data processed at the cloud servers are in compressed format because of the high requirement of bandwidth usage when sending video data in uncompressed format.

Manuscript Received on December, 2014.

Y.N.Prajapati, (SRM-University, NCR Campus Modinagar, A.P, India.

M.K.Srivastava, (SRM-University, NCR Campus Modinagar, A.P, India.

Sandhya, Sharma, (SRM-University, NCR Campus Modinagar, A.P, India.

To use video compression features, particularly in MPEG and ITU-T H.26x video format, FF mpeg [2] contains useful libraries, especially the libavcodec library. FF mpeg provides

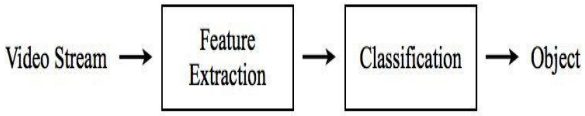


Fig.1.Common Method to Detect Moving Object

interfaces to access some compression features, such as motion vectors and macroblock type, which were used in many research reports on object detection and tracking, e.g., [3], [4]. FFmpeg has also been proposed for video processing in the cloud by [5]. However, their proposal was about how to perform fast video compression in the cloud by applying the FFmpeg application directly, without considering the capabil- ity of the FFmpegs libavcodec.

Therefore, in this paper, we study and propose a mechanism to extract compression features by using FFmpeg, and then to implement an algorithm to detect moving object within a video scene. We also compare the performances of using both compressed and pixel domain to detect moving object.

A. Video Motion Detection

In general, moving object detection method consists of two main processes: feature extraction and classification (Fig. 1). Feature extraction is a process to extract relevant information from the input video data, and classification is a process to group extracted features to discriminate detected objects with other components within the images. In this section, we explain moving object detection method by using both motion vector and pixel value.

Motion Vector (MV)-based Motion Detection:

FF mpeg is a popular open source tool to process video stream. It provides tools to encode, decode, transpose, multiplex, demultiplex, and filter video stream. It contains libraries that can be used to build customized multimedia applications. FFmpeg also provides interfaces to access video compression features. In this case, during decoding, we can parse the compression features easily from video stream for analysis. One of such features is motion vector (MV).

MV is a key component in the motion estimation process in hybrid video encoding system to indicate the position of current MB in the reference frame. Hence, by encoding this pointer and some differential data to compensate the errors, the encoder can obtain better compression efficiency. MV consists of two parts: horizontal and vertical part. It exists

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

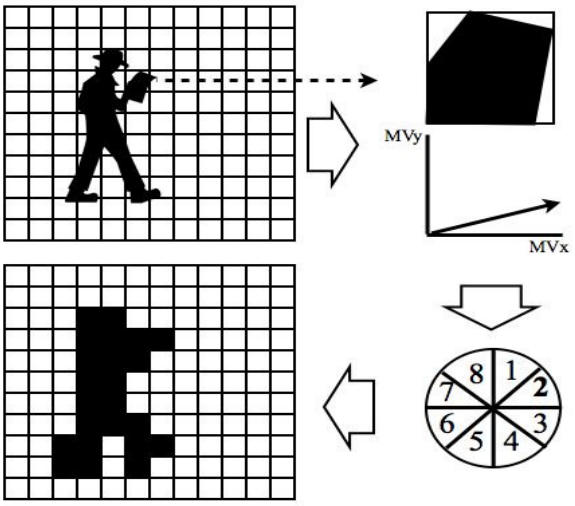


Fig. 2. Illustration of MV-based classification in Macro block (MB) layer.

MB itself contains motion vec- tor, transform coefficients, partition types, and other relevant information. MBs size is normally 16 by 16 pixels and can be divided into partitions to increase compression efficiency. In MPEG-2 standard, a 16x16 pixel MB can be partitioned into one 16x16 pixel partition or two 16x8 pixel partitions. Meanwhile, H.264 defines 4 partitions (16x16, 16x8, 8x16, and 8x8) and 3 sub-partitions for 8x8 partition (8x4, 4x8, and
4x4). In inter-predicted MB, each partition has one MV. Hence, an MB with two 16x8 partitions, for example, has two MVs. We detect motion occurrence in video scene by using a method that classifies MBs in a frame based on the value of MVs of each MB as proposed in [3], [4].

We classify the MBs by finding similarities among closely located MBs' MVs. For example, if two neighboring MVs have similar size and direction, then they are classified into the same class. To remove noise or unwanted MBs, we perform MV thresholding so that we will get clusters of connected MBs, which are called blobs. Each blob is the mask of the moving object being detected. Fig. 2 shows an example of the result of this process on a frame.

Fig. 3 shows the flow diagram to detect moving object. For either P or B frame, we scan every MB and evaluate the MVs magnitude. If the magnitude is a positive value, then we evaluate its direction. We classify directions into 8 equal regions. Based on the value of MV's direction in angle, an MB is assigned an integer number that identifies one of these 8 MV direction regions. MBs are scanned spatially from top-left position horizontally towards the bottom-right corner. If an MB has the same direction identity as the previously scanned MB, then these MB are assigned an identity called area identity number. The same rule applies for other both horizontally and vertically adjacent MBs. MBs with the same area identity number belong to the same area. Then, each area is assigned a bounding box that represents the moving object being detected. Fig. 4 illustrates the process.

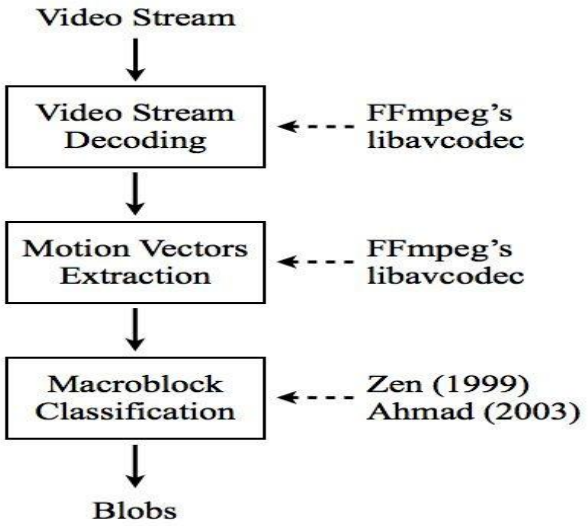


Fig. 3 Diagram of MV-based Motion Detection Process

2) Pixel-based Motion Detection:

To analyze video stream in pixel-based mode, we have to decode the stream first. Decoding can be performed easily by using the FFmpeg's libraries. Then, we extract the features of the reconstructed images.

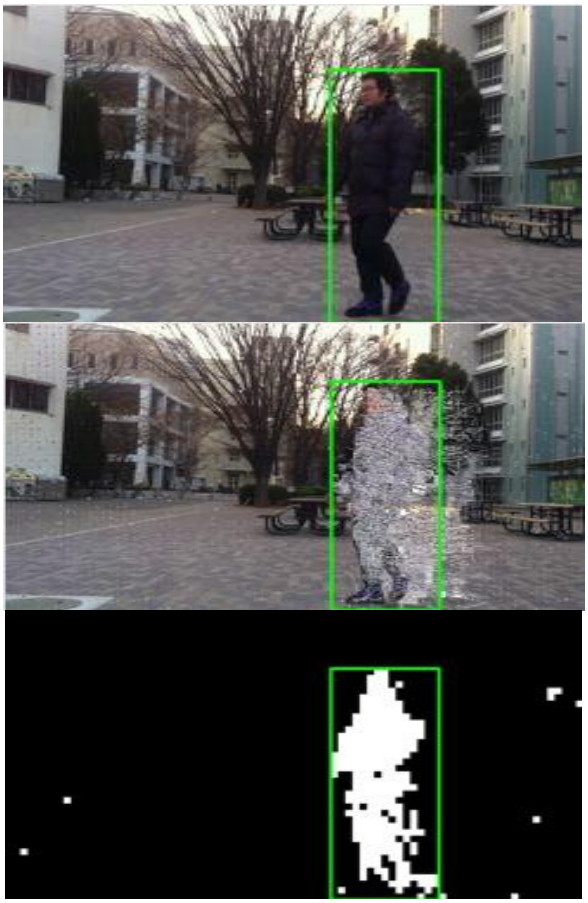
To perform feature extraction, we can use either FF mpeg or Open CV. However, since the detection algorithm has been implemented in Open CV, we decide to use Open CV to do the feature extraction process.

Open CV is a well-known open source tool for image pro- cussing. It has plenty of tools and libraries to do typical image processing jobs, e.g., filtering and segmentation. Open CV also provides some examples to assist us developing an image processing application. We use one provided sample on blob tracking to implement the pixel-based object detection appli- cation. For simplicity, we select the simplest known method to use so that the time complexity is as low as possible. In this case, we use the foreground object detection algorithm in the Open CV's libraries [6], [7].

Fig. 5 shows the simplified diagram of this detection process. FFmpeg is used only to decode the video stream. Then, the object detection function uses the decoded data to detect any moving object by using the available programming interfaces provided by Open CV.

B. Evaluation

We evaluate the performance of both MV-based and pixel-based video motion detection based on the available tools: FFmpeg and OpenCV. We built a computer application that runs on Windows, using MatLab framework, and uses FFmpeg version 0.7.6 and OpenCV version 2.3.0 to process the video data. Fig. 6 shows the user interface of the application. Experiments were conducted on a Mac Pro computer with user 2.2 GHz 2-core processors and 2 GB of RAM. We evaluate the processing time of both MV-based and pixel-based method on a standard definition (SD) video stream.



(a) Moving object (b) Motion vectors (c) Object's mask

Fig. 4. Illustration of the detection process

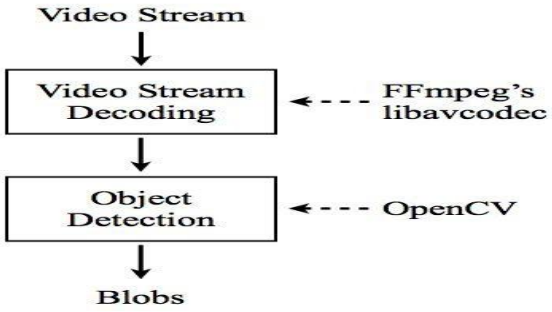


Fig. 5. Pixel-based motion detection method

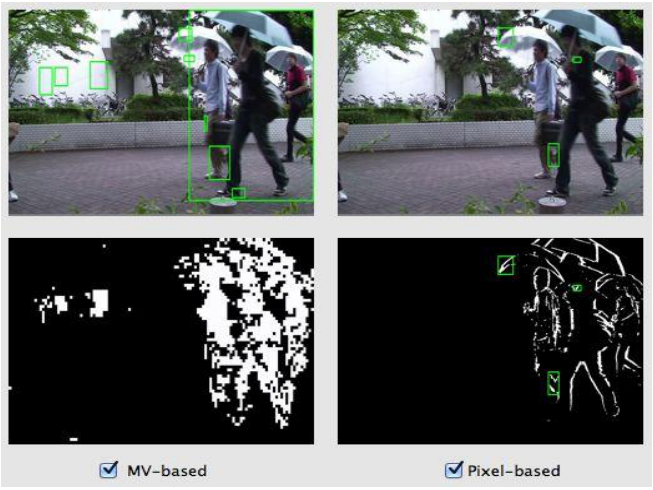


Fig. 6. The user interface of video motion detection application

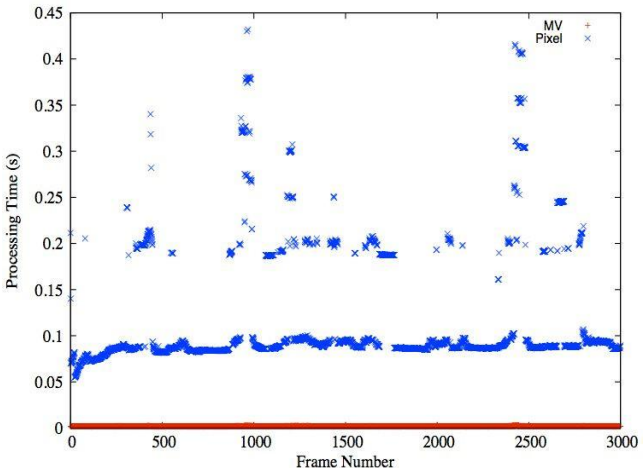


Fig. 7. Average Processing Time for SD Video

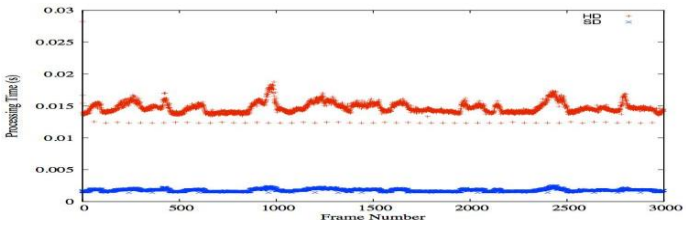


Fig.8. Average Processing Time for HD Video

(HD) one, with five repetitions for each video stream. The resolution of the SD video is 640x360 pixels The frame rate of both videos is29.97 fps, and the frame length is 140 frames or about 1.65 minutes length.

1) Processing Time for SD Video: Fig. 7 shows the average processing time for SD video. The average processing time required by OpenCV-based or pixel-based object detection program is about 0.12 s, and the MV-based method requires about 0.002 s. Hence, the ratio is almost 65 times. The variation of the processing time is negligible compared to its average value.

2) Processing Time for HD Video: Fig. 8 shows the average processing time for the HD video. The average processing time required by OpenCV-based or pixel-based program is about

0.82 s, and the MV-based method requires about 0.02 s, so that the ratio is almost 56 times.

3) Comparison of Processing Time of SD and HD Video: Because the graphs of the average processing time for MV- based method on both Fig. 7 and Fig. 8 are too small to show, we draw the graphs in greater scale and compare them in one figure side-by-side as shown on Fig. 9.

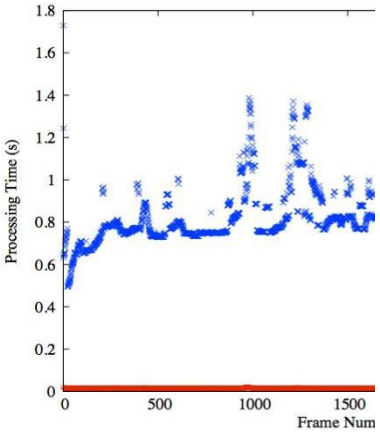


Fig. 9. Average Processing Time of MV-based Motion Detection

In addition, we also draw the comparison of the average processing time of SDand HD video for pixel-based method on Fig. 10. From Fig. 9 and Fig. 10, we can see that when the size of the image increases from 640x360 pixel to 1920x1080 pixel, the time required to process video data by using MV-based method increases almost 10 times. Meanwhile, processing HD video data using the pixel-based method needs almost 7 times of the SD video data. The number of pixels and MBs in one frame of the HD video is 9 times bigger than of the SD video. However, in MV-based processing, the increase of processing time is larger than of the pixel-based processing. However, the average processing time of the MV-based method for the HD video is still lower than the presentation time of the frame, which is about 0.033 s. Therefore, MV-based method is suitable for real-time detection and tracking, even when we use high resolution video.

Although the pixel-based method was not probably optimized yet, basically the pixel-based method will work slower than the MV-based method because of the following reasons. First, the amount of data to process, which is the number of MBs, is much smaller than the amount of data that is processed in pixel-based method. Second, the MV-based method runs in parallel with the decoding process, while the pixel-based method has to wait for the reconstructed image to start to run.

II. CONCLUSION

We have presented video motion detection methods, motion vector (MV) based and pixel-based methods, using open source tools, FFmpeg and OpenCV libraries. With the existing tools, we could build moving object detection application by simply extending the video decoding functions and implement- ing motion vector clustering algorithm.

Experiment results have shown that the MV-based method worked tens times faster than the pixel-based method. However, both methods were not optimized yet and further experi- ments are required to investigate the recall and precision rate. In addition, the exact costs that are required to implement the applications in the cloud-based system also need to be examined.

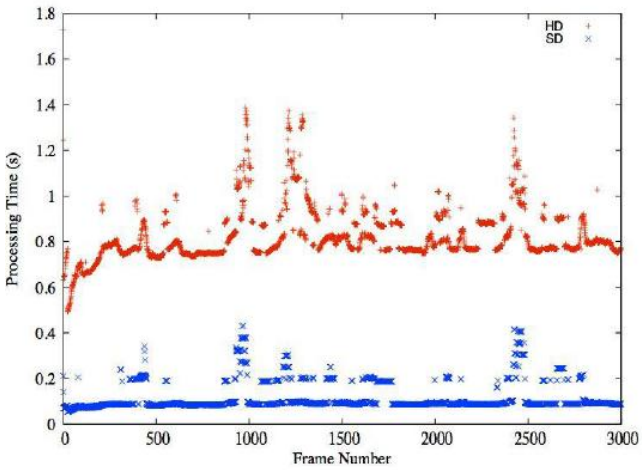


Fig. 10. Average Processing Time of Pixel-based Motion Detection

REFERENCES

[1] Open Source Computer Vision (OpenCV) [Online]. Available http://opencv.willowgarage.com/wiki/. Accessed February 21st, 2012.

[2] FFmpeg [Online]. Available http://ffmpeg.org/. Accessed February 21st,
2012.

[3] H. Zen, T. Hasegawa, and S. Ozawa, Moving object detection from MPEG coded picture, Proc. of 1999 International Conference on Image Processing (ICIP), Vol. 4, 1999, pp. 25-29.

[4] Ashraf M.A. Ahmad, Duan-Yu Chen, and Shu-Yin Lee, Robust object detection using cascade filter in mpeg videos, Proceedings of the IEEE 5th International Symposium on Multimedia Software Engineering (ISMSE),
2003, pp. 196203.

[5] Pereira, R. et al., An Architecture for Distributed High Performance Video Processing in the Cloud, Proc. of 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD), 2010, pp. 482-489.

[6] The OpenCV Video Surveillance / Blob Tracker Facility. [Online].
Available: http://opencv.willowgarage.com/wiki/VideoSurveillance/. [Accessed: February 1st, 2012].

[7] Chen, T. et al., "Computer Vision Workload Analysis: Case Study of
Video Surveillance Systems," Intel Technology Journal, May 2005, pp. 109-1