RESEARCH ARTICLE OPEN ACCESS

Embedded Smart Video Surveillance Using Histogram

Medari Srikanth[1], M.Kattaswamy[2], B.Brahma Reddy[3], Balachandran G[4]
[1](PG Scholar, Department of Electronic Communication Engineering, Vignana Bharathi institute of technology, Hyderabad)
[2](Assistant Professor, Department of Electronic Communication Engineering, Vignana Bharathi institute of technology, Hyderabad)
[3](Head of the Department, Department of Electronic Communication Engineering, Vignana Bharathi institute of technology, Hyderabad)
[4](Associate Director, Cognizant Technologies Solutions, Hyderabad)

ABSTRACT
Video surveillance has long been in use to either monitor security sensitive areas or to know the statistic details for business purposes. Typical application includes banks, shopping complexes, Railway stations, crowded public places and borders. The advancement in computing power, availability of large-capacity storage devices and high speed network infrastructure paved the way for cheaper, multi sensor video surveillance systems. Traditionally, the video outputs are processed online by security officials and usually saved to tapes for later use only after a forensic event. However, assisting the human operators with identification of important events in video by the use of “smart” video surveillance systems has become a critical requirement. The making of video surveillance systems “smart” requires fast, reliable and robust algorithms. This mandates for this smart feature to run on camera itself; embedded software using spare processing capacity in Camera’s Digital signal processor (DSP). In this article we propose a novel algorithm which is small in footprint and quick in identifying the activity using Event Trigger and process the image using histogram which can be embedded in DSP system.
Keywords- Digital Signal Processor, Event Trigger, Histogram, Smart Surveillance, Video Surveillance

I. INTRODUCTION

Moving Object detection is one of the major video analytics components in video surveillance application. In typical Video Surveillance application, camera captures image, does intensity correction, compress the video ad transfer data to backend PC for further analysis. The video analytic components are executed in PC using offline method and basic surveillance is done by security official directly monitoring it online. Most of the techniques used for this problem deal with a stationary camera [3, 2] or closed world representations [7, 5] which rely on a fixed background or a specific knowledge on the type of actions taking place. The complexity of analytics makes back end processing by PC as preferred choice for the analytics. The video analytics at camera using embedded software helps a lot in first level surveillance assistance. With recent development in multi-cores and highly integrated processor, as well as efficient analytics algorithm, the first level surveillance can be achieved using embedded system. In this paper, we have proposed a computationally economical method for object detection which can be executed from camera or embedded electronics in camera itself. The proposed method uses pad based Event Trigger and histogram based object detection. The proposed method is fast, reliable and robust and it enables smart video surveillance..

II. BACKGROUND PAPER

The Detecting regions of change in images of the same scene taken at different times is of widespread interest due to a large number of applications in diverse disciplines. Despite the diversity of applications, change detection researchers employ many common processing steps and core algorithms [4]. The major methods of moving object detection include point detectors, background subtraction and segmentation and supervised learning [8]. Girisha and Murali [4] describe a method for segmenting motion objects from background that is based upon temporal frame differencing. Ko et al. [5] describe a segmentation technique in which the background at each pixel location is represented by a histogram of values in the region surrounding that location. Foreground is detected using a thresholded Bhattacharyya distance between the current histogram at the pixel location and that of the temporally recursive updated background distribution. We have avoided supervised classification approaches because of their likely demands on the processing capacity of the embedded processors. Also we avoid full frame background modeling and use temporal frame differencing to trigger our detector to avoid difficulties associated with morphological processing. We detect objects using event trigger and use the integral histogram approach [1, 6] in order to extract multiple histograms efficiently.

III. APPROACH

The first step in the Moving Object Detection is to quickly know whether we need to initiate moving object detection or not. Such initiation requires quick detection to know whether any activity took place in the captured video frames before doing any elaborate analytics. Since the processing power is limited, we propose a novel method to quickly detect activity within multiple frames. The proposed method uses 1/10th the size of the video resolution instead of entire frame size image for computation. This area is called 'pad'. This pad can be configured for size, location and array size. The pad size can be smaller or larger than the object.

The first step in the process is the construction of the thresholded pad temporal difference image D_t .

D_t(X,Y) = (| P_t(X,Y) - P_{t-1} (X,Y)| > θ_D (1)

Where ***P****t* is the pad image at time *t* and Ɵ*D* is a pixel difference threshold. With the low frame rate the pixel value changes from frame to frame can be significant. The mean difference $\hat{d}$t across the pad is calculated:

$\hat{d}$t = 1/N $\sum_{x,y\in=p} Dt(X,Y)$ (2)

Where *N is* the number of pixels in the pad image. A trigger occurs when ***D****t* rises through a mear differences threshold θ_D and the mean differences are increasing:

$\hat{d}$t > Ɵd ^ ($\hat{d}$t - $\hat{d}$t-1) > 0 (3)

Figure 1: Threholded pad difference image

Through this method, we can detect the activity start frame number (trigger point). In proposed method, we have used all frames for the trigger point identification. However, to further optimize the processing, we can choose every 3 to 5 frames for pad area analysis. This information can be configured in selected algorithm.

In the absence of a trigger we make a normalized 8 bin 'probably background' grey scale histogram q_B^t of the pad image at time *t*. The histogram is used to recursively update a time averaged 'probably background' histogram $q_b^{\sim t}$:

$q_b^{\sim t}$= (1-α) $q_b^{\sim t-1}$+αq_B^t (4)

Where α has a value typically of the order 0.2.

When a trigger condition is met we make a histogram Pt of the pad image. This contains information about both background and object. The first step is to construct a difference histogram q_d^t by subtracting the averaged 'probably background' histogram from the current pad histogram:

q_d^t = P^t - $q_B^{\sim t}$ (5)

Negative bin values in the difference histogram indicate pixel value ranges that are more representative of the background rather than the foreground. The Figure 2 Histograms shows a set of typical histograms associated with this stage of the process. Next, a difference masked pad image M_p is made by pixel wise multiplication of the pad thresholded temporal difference image with the pad image i.e.

M_p (X, Y) = D_t (X, Y). P_t (X, Y) (6)

The masked image will contain pixels associated with the object, shadows and zero values associated with the static region. The masked image is grey level sliced using the bin boundaries of the *n* = 8 bin histogram to produce a binary pad image B_p^u for each slice *u*:

B_p^u (X,Y) = б (b (M_p (X,Y)) -u) (7)

Where b (Mp(X,Y)) is a function that compares the grey scale value Mp(X,Y) against the bin boundaries and returns the bin number associated with it; б is the Dirac delta function. The lower boundary of the first bin is set to 1 rather than 0 out of the range [0, 255]. This removes the mask static pixel contribution. The bins of the proposal histogram q_o^t are set to be the sums of the corresponding binary slice if the associated difference histogram bin q_d^t is positive, otherwise they are set to zero:

$$q_o^t(u)= \begin{cases} \sum_{X,Y\in P} B_p^u(X,Y) & \text{if } q_d^t(u) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

The 'indicative object' proposal histogram is then normalized:

$q_o^{\sim t}$: $q_o^{\sim t}$(u) = q_o^t (u) / $\sum_{v=0}^{n}$ q_o^t(v) (9)

The proposal histogram indicates the grey scale values which are more characteristic of the part of the object entering the pad region than the background. We use this proposal histogram of the incoming object. Even though the proposal histogram is not a true representation of the object histogram, it is likely to be closer to it, in terms of the Bhattacharyya distance, than to histograms calculated from background in the region of the pad after the triggering event. The proposal histogram does not include bins that are dominant in the probably background histogram.

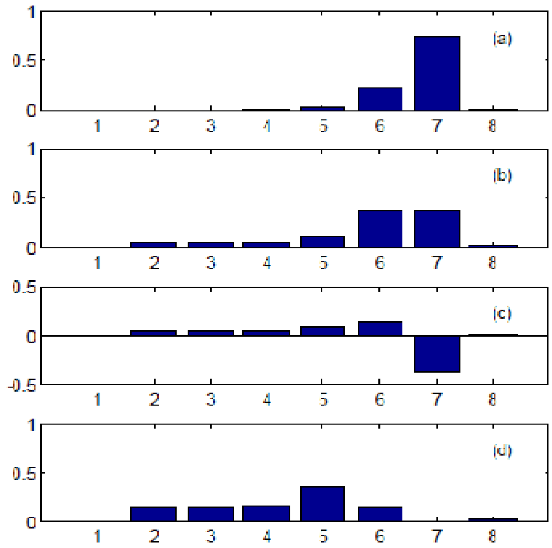

Figure 2: Histograms: a) Average probably background q_B^t, b) Pad image P^t, c) Differences q_d^t, d) Proposal $q_o^{\sim t}$

IV. IMPLEMENTATION

The implementation of the method captured video processed to MATLAB with a standard web camera in Laptop and OMAP DM3730 ARM Cortex A8 Processor Board.

Figure 3: Development system

The Image Acquisition tool box in MATLAB is used for image acquisition. The configuration for the image acquisition is given below

TABLE I
IMAGE ACQUISITION CONFIGURATION

Configuration Parameter	**Value**
Resolution	160 x 120
No of Frames	61
Frame Rate	15 / Sec
No of Trigger	1
Trigger Type	Manual
Device	Integrated Webcam

The capture video using above setting is use for further processing of object detection. Each stage of the processing like Pad marking, temporal difference Pad image, histogram, etc are stored as separate .avi files.

The MATLAB application is developed with single user screen where it displays the different processing and report the final object detection using histogram. Also the screen indicates the event trigger frame with details of the frame number in which the object enters the background.

V. CONCLUSION

We have addressed unique problems related to the analysis of a video data. The framework proposed is based on histogram from a video acquired by an ordinary webcam. The proposed method is used as smart video surveillance with abilities to integrate the intelligence in to the camera device itself. Finally, the quantification of the results by proposed method provides a confidence measure characterizing the quick, reliable and robust object detection. The method does not demand the use of a target model nor does it require the development of a full background image or classifier training. It works with moderate quality monochrome footage and can be used in a range of contexts.

The obtained results will be improved by further processing with more number of pads and initiating tracker for the identified object.

REFERENCES

[1] Chipwrights, "Programmable Visual Signal Processors", www.chipwrights.com
[2] W.E.L. Grimson, L. Lee, R. Romano, and C. Stauffer. Using adaptive tracking to classify and monitor activities in a site. In *CVPR98*, pages 22–31, 1998.
[3] Haritaoglu, D. Harwood, and L.S. Davis. W4S: A real-time system for detecting and tracking people in 2 1/2-d. In *ECCV98*, 1998
[4] Richard J.radke*, srinivas Andre, Omar al-kofahi, and badrinath roysam,"image change detection algorithms:A systematic survey" rensselaer polytechnic institute,USA.august 19,2004
[5] S.S. Intille, J.W. Davis, and A.F. Bobick. Real time closed world tracking. In *CVPR97*, pages 697–703, 1997.
[6] F. Porikli, "Integral Histogram: A Fast Way to extract Histograms in Cartesian Spaces", IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, pp829-836.
[7] M. Isard and A. Blake. A mixed-state condensation tracker with automatic model-switching. In *ICCV98*, pages 107–112, 1998
[8] Yilmaz, A., Javed, O., and Shah, M. 2006. Object tracking: A survey. *ACM Comput. Surv. 38*, 4, Article 13 (Dec. 2006), 45 pages. DOI = 10.1145/1177352.1177355http://doi.acm.org/10.1145/1177352.1177355