Clustering approach for Moving object detection from video sequence

Parth V. Parekh[1], Prof. Pinal J. Patel[2], Shrikrishna Padalkar[3], Vaibhavi S. Ghariya[4]

[1]Computer Science & Engineering Dept., Student, Government Engineering College, Gandhinagar, Gujarat, India
[2] Computer Science & Engineering Dept., Asst.Prof.. Government Engineering College, Gandhinagar, Gujarat, India
[3]Computer Science & Engineering Dept., Studetn. Nirma University Gujarat, India
[4]Computer Science & Engineering Dept., Student, Nirma University, Ahmedabad. Gujarat, India

ABSTRACT

Usually, the video based moving object detection deal with non-stationary image stream that changes over time. Robust and Real time moving object tracking is a problematic issue in computer vision research area. We examine the difficulties of video based detection of object and step by step, we analyze these issues. We had applied different approaches on video sequence and hurdles. And we had used Gaussian Mixture model as a resolution to those difficulties.

Keywords: *Gaussian Mixture Model, Moving Object Detection, Computer Vision, Statistical Video Processing.*

1. INTRODUCTION

Video based moving object tracking is one of the exigent missions in computer vision area such as visual surveillance, motion capture, human computer interactions etc. Video based tracking basically accord with non-stationary image, target object descriptions and the background which change over time. The most available algorithms are able to perform tracking simply in predefined and well controlled environment. The performance of these applications is dependent on the Finding the Moving Object algorithm being robust to illumination changes, small movements of background elements (e.g. swaying trees), the addition or removal of items in the background (e.g. parked car), and shadows cast by moving objects. Computational efficiency is also of high priority. The most common paradigm for performing Finding the Moving Object is to build an explicit model of the background. Moving objects are then detected by taking the difference between the current frame and this background model. Typically, a binary segmentation mask is then constructed by classifying any pixel as being from a moving object when the absolute difference is above a threshold. Finding the Moving Object algorithms differ in how they define and update the background model. Despite the success enjoyed by Finding the Moving Object algorithms, it is becoming clear that post-processing is required in order to improve their performance. This post-processing can range from shadow detection algorithms operating at the pixel level to connected component labeling which identifies object-level elements. The results of post-processing can be used to directly improve the quality of the segmentation mask and fed back into the Finding the Moving Object algorithm in order to facilitate more intelligent updating of the background model.

Finding the Moving Object, although being simply defined as a difference between the background image without objects of interest and an observed image, has many difficult issues to overcome, making it a problem that has inspired a wealth of research. For instance, the type of situation for which it is needed exposes many problems and a Finding the Moving Object algorithm that works well in one scenario may not necessarily work as well in another.

This paper looks at Finding the Moving Object with respect to videos, comparing an obtained background image to video frames, as opposed to Finding the Moving Object for still images Although there are many algorithms for Finding the Moving Object, they all follow a general pattern of processing as shown in Figure 1 . Firstly, video frames captured from a camera are input to the background subtractor. Preprocessing stages are used for filtration and to change the raw input video to a process able format. Background modeling then uses the observed video frame to calculate and update the background model that is representative of the scene without any object of interest. Foreground detection is where the pixels that show a significant difference to those in the background model are flagged as foreground.

Data validation is used to examine the found objects of interest and to eliminate any false matches. A foreground mask can then be output in which pixels are assigned as foreground or background. The foreground detection stage can be described as a binary classification problem whereby each pixel in an image is classified as foreground or background. Formally for every pixel p in image I, each pixel is either 0 (background) or 1(foreground). After this mask is obtained, background pixels are usually set to white or black to allow focus on the foreground object.

2. PROBLEMS AND SOLUTIONS

Many algorithms are based on the basic principle of subtraction pixel values in the observed image from pixel values in the background image. However the nature of the realistic environments in which these systems are used introduces many problems which cause the incorrect classification of a pixel as foreground. This section will briefly describe some of these problems and some of the solution that literature study suggests to alleviate them [2], [4].

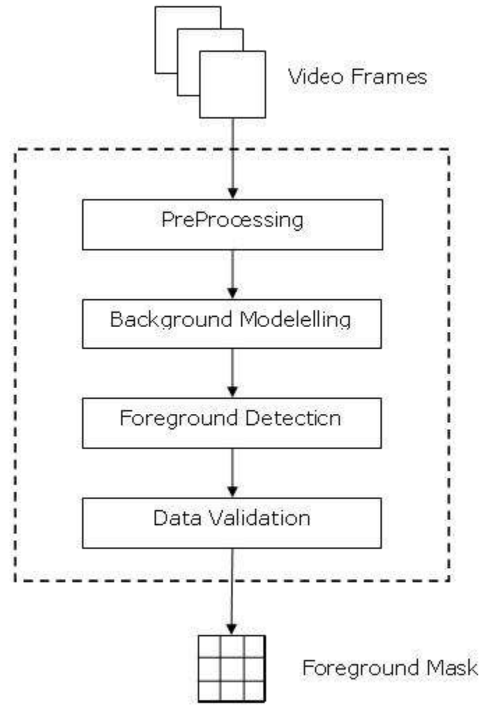

Fig. 1Basic outline of Finding the Moving Object Algorithm

Changes in Illumination alter the color composition of the background. In color and intensity based algorithms this change causes a large difference in the subtraction and therefore increases the number of false detections. E.g. Turning on a light or a cloudy day.

Relocation of a Background Object Cause changes in two regions, the new position of the object and the former. Both positions will be picked up as foreground due to change in color. E.g. A stationary object is moved to some nearby location.

Non-static Background causes fluctuation in the pixel values causing change in color based detection algorithm that results in false matches in these areas. E.g. Tree leaves moving due to wind.

"*Similar background and foreground color*" - These pixels will not be classified as foreground as they are not dissimilar enough. E.g. If someone is wearing clothes that are similar to background color.

Shadows Objects can cast shadow areas which are darker than the background color in that area they will be wrongly classified as foreground pixels due to illumination change in the shadow region. E.g. A person moving in sunlight.

Many methods have been suggested to appease the problems described above that can be added to pre and post-processing stages. Shadows are one of the biggest issues and as such have inspired a wealth of research in the area of shadow removal alone. During preprocessing, Smoothing of the images can be used to reduce the transient environmental noise such as rain. Many algorithms use a Gaussian blur first to average out fluctuating pixel values to alleviate big differences [?]. Alternatively when temporal data can be exploited in a video, if a pixel's value is constantly changing over time then it can be assumed it is part of a non-static background object. The background model can deal with events such as objects changing positions by implementing an effective update rule to change the model over time. Background modeling is an area of research itself. One example of an update process is to track object locations. If an object moves and then remain constantly in the same position over a length of time it can be considered to be a part of background. Illumination changes can be handled by exploiting illumination invariance within the color space used. Post processing can be used for data validation to eliminate false positive matched. This can be in the form of the rejection of isolated foreground pixels as they can be assumed to be noise or thresholding on foreground region size. As the subtraction usually only looks at a single pixel, this stage can also examine the value of the neighbors [4].

The first step in developing a background subtractor is to build a model of the background. Since there are no preset background images to use, the subtractor will have to generate a model automatically. Various methods for Finding the Moving Object had been studied and analyzed, The methods, their advantages and disadvantages are noted below.

3. MOVING OBJECT DETECTION

3.1 Movie Frame Difference Technique

Many Finding the Moving Object algorithms reduce down to simple subtraction of the pixel in the expected background image from the pixel in the observed image and any significant change indicates that an object of interest has been identified. This is the most nave approach. First takes the Frame 1st as a Base Frame, then compared the base with rest of the frame we will compare that frame one by one up to nth Frame, then we will find where the base frame is varying from other Frame, At that value substitute the value of that in the base Frame. But do not include the change that is going to happen in the other than frame then base frame. But this technique cannot provide decent result, if there is not significant different between successive frames.

3.2 Mean Method

In this method the mean of all pixels from frame-1 to frame-n is calculated. After that we will compare that mean model to each and every frame in that, we will specify one threshold value if the current pixel fall in that range then we can say that is belongs to the background otherwise we will specify it as foreground object.

Fig. 2 Base Frame

Fig. 3 n^{th} Frame

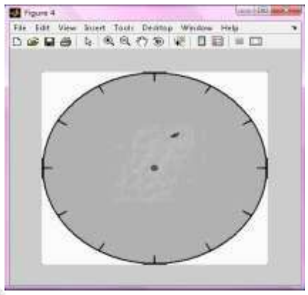

Fig. 4 Approximate Background

This technique gives good result then the previous technique. These techniques also required less no of frame/sec. But these technique will not work well when we significant change in successive frame.

3.3 Graph-Cut Method

This is the one of the technique for Finding the Moving Object. This is simplest and cheapest technique because in this technique it required the background of whatever video it has given to it to find the foreground and background object. This technique is not useful because in many video we cannot get the background without the object or we have video that start from anywhere in medial of it. In this type of situation these technique is not useful. If we get the background of any video then this technique work efficiently and it save time also.

3.4 Gaussian Mixture Model

4.

Finding the Moving Object is a commonly used class of techniques for segmenting out objects of interest in a scene for applications such as surveillance. It involves comparing an observed image with an estimate of the image if it contained no objects of interest. The areas of the image plane where there is a significant difference between the observed and estimated images indicate the location of the objects of interest. The name “Finding the Moving Object” comes from the simple technique of subtracting the observed image from the estimated image and thresholding the result to generate the objects of interest.

4. GAUSSIAN MIXTURE MODEL

4.1. Basic about GMM

A Gaussian Mixture Model copes up with multimodal background; hence it is widely used in finding the Moving Object. It calculates each pixel-value from all the sample pixels’ mean and variance [2]. GMM is created for each pixel and updated with each new frame. At every new frame some of the Gaussians matches the current value, for them, mean and variance is updated by the running average. Usually the intensity plot of a pixel is a multimodal plot as shown in the Fig. 5. Hence a single Gaussian is unable to capture its multimodal behavior causing the requirement for Gaussian Mixture model. Even the Literature studies shows that Gaussian Mixture Model is more suitable in such kind of system; hence we will use it in our system.

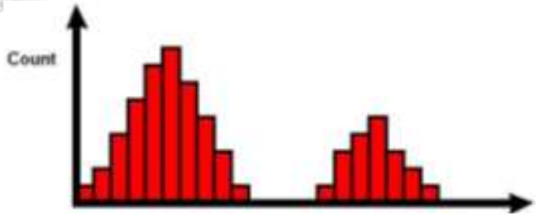

Fig. 5. Intensity plot of a pixel is a multimodal plot

4.2 GMM Updation

Basically GMM have three parameters that is going to update pixels by pixels and frame by frame. The Parameters are

Mean, Co-Variance and Mixing parameter.

4.2.1 Mathematical Proof of the Gaussian Mixture Model

This section describes the mathematical proof of the GMM. So, far we know that in GMM there are basically three parameters Co-variance, mean and mixing parameter. In these proof we show that how to update these three parameter accordingly [4].

4.2.2 Learning Gaussian Mixture Models

The GMM $G(t) = (C_i(t))_{i=1m}$ is a finite set of clusters of size m, where a cluster at t^th instant is given by,

$$\boldsymbol{C_i(t) = (\mu_i(t), \delta_i(t), \pi_i(t))} \dots\dots\dots\dots\dots\dots\dots \ (1)$$

Where, $\mu_i(t), \delta_i(t), \pi_i(t)$ are the respective mean vector, co-variance matrix and the mixing parameter of $C_i(t)$ at the t^{th} instant.

4.2.2.1 Initialization

The GMM is initialized with a single Cluster $C_1(1) = (X_1;\ \delta_{init}; 1:0)$ where X_1 is the data vector at $\boldsymbol{t = 1}$ and δ_{init} is the initial co-variance matrix whose values which are assigned from the domain knowledge.

4.2.2.1 Update

In this sub-section we deduce the equations for updating the GMM $G(t-1)$ learned till the $(t-1)^{th}$ instant to $G(t)$ with the current data vectorX_t. We consider the data vector to be belonging to the cluster

$$\boldsymbol{C_j(t-1), if\left(X_t - \mu_j(t-1)\delta_j(t-1)^{-1}(X_t - \mu_j(t^{-1}))\right)_{jn\gamma W}}.$$

Here γ is a user defined threshold and n is the dimension of the data vector ($X \epsilon R^n$).Now, we consider the following cases. In the first case, we assume that $\exists_j$: $\boldsymbol{X_t \in C_j(t-1)}$. Let $N_i(t)$ be the number of data vectors that has been assigned to $\boldsymbol{C_j(t)}$ till the t^{th}instant. Thus we have,

$$\boldsymbol{\pi_i(t) = \frac{N_i(t)}{t}} \dots\dots\dots\dots\dots\dots\dots \ \boldsymbol{(2)}$$

$$\boldsymbol{\pi_i(t) = \frac{(t-1)\pi_i(t-1) + \delta(i-j)}{t}} \dots\dots\dots\dots\dots\dots\dots\dots.\boldsymbol{(3)}$$

$$\boldsymbol{\pi_i(t) = (1-\alpha_t)\pi_i(t-1) + \alpha_t\delta(i-j)} \dots\dots\dots\dots\dots\dots\dots\dots.\boldsymbol{(4)}$$

Where, $\alpha_t = t, \boldsymbol{\delta(i-j)}$ is Kronecker's delta? Now we'll update the mean and co-variance in $\boldsymbol{C_j(t-1)}$ only. We may follow following procedure to update co-variance.

$$\boldsymbol{\mu_i(t) = \frac{1}{N_i(t),\ X \epsilon C_j(t)} \sum X} \dots\dots\dots\dots\dots\dots\dots\dots.\boldsymbol{(5)}$$

$$\boldsymbol{\mu_i(t) = \frac{N_j(t-1)\mu_j(t-1) + X_t}{t\pi_j(t)}} \dots\dots\dots\dots\dots\dots\dots\dots.\boldsymbol{(6)}$$

$$\boldsymbol{\mu_i(t) = \left(1 - \beta_j(t)\right)\mu_i(t-1) + \beta_j(t)\, X_t} \dots\dots\dots\dots\dots\dots\dots \ (7)$$

Where, $\boldsymbol{\beta_j(t)} = \frac{\alpha_t}{\mu_i(t)}$.

We can update the co-variance matrix in the same way. From the definition, we can compute co-variance matrix at the t^{th} instant as,

$$\delta_j^2(t) = \frac{\mathbf{1}}{\boldsymbol{N_i(t),\ X\,\epsilon C_j(t)}} \sum (\boldsymbol{X} - \boldsymbol{\mu_j(t)})(\boldsymbol{X} - \boldsymbol{\mu_j(t)})^{\boldsymbol{T}} \ldots\ldots\ldots\ldots\ldots\ldots.. (\mathbf{8})$$

$$\delta_j^2(t) = \frac{\mathbf{1}}{\boldsymbol{N_i(t),\ X\,\epsilon C_j(t)}} \sum \boldsymbol{XX^T} - \boldsymbol{\mu_i(t)\,\mu_j(t)^t} \ldots\ldots\ldots\ldots\ldots\ldots\ldots (\mathbf{9})$$

Now, further manipulating, by substituting the update rule for $\boldsymbol{\mu_j(t)}$, it can be shown that the updated co-variance matrix is given by,

$$\delta_j^2(t) = \left(1 - \boldsymbol{\beta_j(t)}\right) \delta_j^2(t-1) + \boldsymbol{\beta_j(t)}\left(\left(\boldsymbol{X_i} - \boldsymbol{\mu_j(t-1)}(\boldsymbol{X_j} - \boldsymbol{\mu(t-1)^T})\right)\right) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.. (\mathbf{10})$$

Case 2: In second case, it may happen that $\exists_j$: $\boldsymbol{X_j} \in \boldsymbol{C_j(t-1)}$, in such cases we initialize a new cluster $\boldsymbol{\beta_j(t) = (X_{t,}, \delta^2_{init}, \alpha_t)}$.

Case 3: If $G(t-1)$ contains less than m clusters, then we add $C_k(t)$ to it.

Case 4: Otherwise, $C_k(t)$ replaces the cluster with the lowest weight. More so, in this particular case, the mixing parameter of other clusters are panelized.

$\pi_i(t) = (1 - \alpha_i)\pi_i(t-1), i \neq k$.

4.2.2.3 Implementation Code of Moving Object Detection

We have developed the recursive equation for updating of GMM. The algorithm for Finding the Moving Object is as follows:

```
ReadInitialFrame();
InitailizeGMM();
while(FramesLeft)
(
        ReadNextFrame();
        UpdateGMM();
)
Reopen()
while(FrameLeft)
(
        ReadNextFrame();
        ApplyGMM();
        WriteOutputFrame();
)
```

After this step erosion/dilation is performed, this will help in removing noise. Dilation, in general, causes objects to dilate or grow in size; erosion causes objects to shrink. The amount and the way that they grow or shrink depend upon the choice of the structuring element

5. Experimental Results

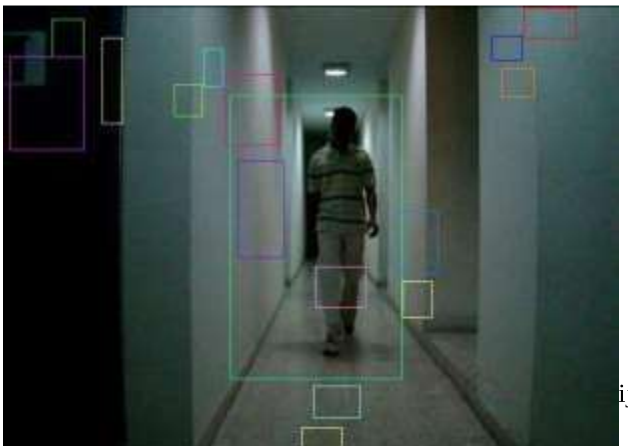

Fig. 6 Output of GMM Implementation

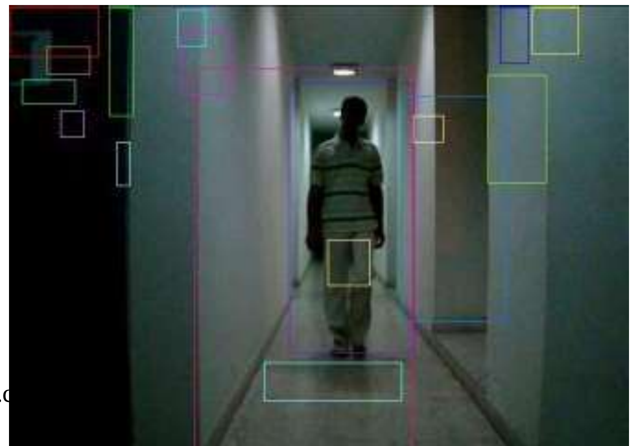

Fig. 7 Output of GMM Implementation

6. CONCLUSION

In this paper, we first presented different techniques to find the moving objects in video sequences and then presented a new approach based on Gaussian mixture model. The preliminary experimental results demonstrate the effectiveness of the algorithm even in some complicated situations like change in illumination, relocation of object, similar background and foreground color, shadow objects and non-static background.

7. REFERENCES

[1]. F.RieszandB.Sz-NAgy, "Functional analysis,"1955.

[2]. M.Piccardi, "Background subtraction techniques: Are-view,"April2003. www-staff.it.uts.edu.au/~massimo/ Background Subtraction Review-Piccardi.pdf.

[3]. P.Sinha, "Object recognition via image invariants, a casestudy," May1994.

[4]. S.Deane, "A comparison of background subtraction techniques, IEEE,April http://portfolio.ecs.soton.ac.uk/20/3/irp-report-ieee.pdf.

BIOGRAPHIES

Mr. Parth Parekh has pursued Diploma and Degree From Nirma University with First Class. He has published paper on Moving Average Methods. He has been a research scholar since his bachelor career. He has been part of a patent application filed by Nirma University on Efficient Face Recognition based Anti Theft System.