North Carolina Agricultural & Technical State University

Department of Electrical and Computer Engineering

Final Project

Moving object tracking using background subtraction from a stationary camera

By:

Biniam T. Gebru (***-**7-495)

Mrinmoy Sarkar (***-**3-260)

Solomon G. Gudeta (***-**8-736)

ELEN 657: Digital Image Processing

Dr. Jung H Kim

Spring, 2018

Table of Contents

ABSTRACT 1

1 TECHNICAL DESCRIPTION..2

1.1 Introduction to Background Subtraction 2

1.2 Background Subtraction Taxonomy 4

1.3 Approaches 4

1.3.1 Running-Average Model 5

1.3.2 Gaussian Mixture Model 6

2 SIMULATION & RESULTS 12

2.1 Simulation Description 12

2.1.1 Implementation I – Running Average 12

2.1.2 Implementation and result analysis of GMM based detection 14

2.2 Conclusion...15

2.3 Future works..16

ANNEX: MATLAB SOURCE CODE 17

REFERENCE..23

Abstract

Detection of moving objects in a video sequence is a difficult task and robust moving object detection in video frames for video surveillance applications is a challenging problem. Object detection is a fundamental step for automated video analysis in many vision and digital image processing applications. Object detection in a video is usually performed by object detectors or background subtraction techniques. Frequently, an object detector requires manual labeling, while background subtraction needs a training sequence. In this project proposes, we attempt to solve the problem of detecting moving object with fixed camera and identifying a non-moving background with a moving object of interest using the latter technique.

1 Technical Description

1.1 Introduction to Background Subtraction

The detection of change is a low-level vision task used as a first step in many computer vision applications such as video surveillance, low-rate video coding, human-computer interaction, augmented reality or medical diagnosis to mention only a few of them. Given an image sequence, the goal is to identify for each frame the set of pixels that are significantly different from the previous frames. Depending on the application, the requirements and constraints of the detection algorithm are different. Likewise, the definition of what is significantly different, may depend on the application domain. In the video surveillance domain, change detection has been frequently used in order to segment foreground objects from the background. Foreground objects are the objects of interest in an automated surveillance system. The segmented foreground objects are then associated between frames in order to perform a scene analysis and detect events of interest. As background is understood what is normally observed in the scene. Therefore, it is assumed that the background can be well described by means of a statistical model, the background model. Nevertheless, there are some background characteristics as moving foliage or sudden illumination changes, which might make difficult the task of background modelling and maintenance.

A comprehensive study of the main challenges and some principles that might be used to tackle them can be found in [1]. The segmentation of foreground objects by means of detecting the changes with reference to a background model is commonly known as background subtraction. Figure 1.1 depicts a basic schema of a general background subtraction system. The main challenges a background subtraction algorithm has to deal with are [1][2]:

- Gradual illumination changes, which are mainly experienced in outdoor environments along the different times of the day and affect the appearance of the objects in the observed scene.
- Sudden illumination changes, which are mainly experienced in indoor environments by the switching on and off of artificial light sources, and in outdoor environments by unstable weather conditions when clouds suddenly hide the sun.

- Shadows, which are mainly casted by moving objects and complicate the accurate segmentation of objects (static objects belonging to the background also cast shadows; nevertheless, these are not that problematic for the background subtraction process since they are always casted at the same position -or at slow moving positions in outdoor scenarios depending of the sun position- and can be more easily accommodated in the background model).
- Dynamic background, which are those parts of the background exhibiting different appearances because of containing some kind of moving objects as waving trees, rippling water, escalators and so on, which are not of further interest for a scene interpretation. • Camouflage, produced by objects whose appearance is difficult to differentiate from the appearance of the background.
- Bootstrapping, which is required because of the general unfeasibility of training a background model with a completely empty scene.

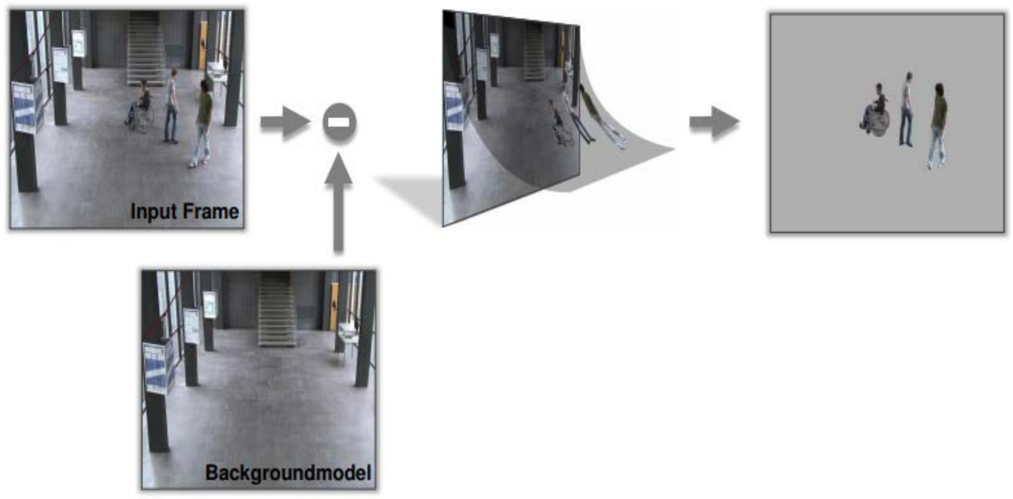

Figure 1.1 General background subtraction system

Actually, in [1] the authors also pointed out some challenges which they claimed that a background maintenance system should be able to handle:

- Moved objects, which refers to the detections corresponding to background objects that have been moved.
- Sleeping person, which refers to foreground objects appearing in the scene and remaining motionless after a while.
- Walking person, which refers to objects that have been learned as part of the background and at some point in time start moving and leave the scene.

1.2 Background Subtraction Taxonomy

The number of background subtraction approaches which have been proposed in the literature is large, and so are the different taxonomies which can be used to classify them. Attending to the spatial level considered, background subtraction approaches can be divided into three classes:

- Pixel-level algorithms only use features gathered at each single pixel position. These methods are very fast, but they do not use any kind of inter-pixel relationships. There have been many proposals in the literature for these kind of methods; among them, Running Gaussian Averages , Median Filtering , and Gaussian Mixture Models, have been of special relevance and have originated a vast number of derived systems.

- Block-level based approaches divide an image into blocks and compute block related features to describe the background. Block-level approaches are usually more robust against noise than pixel-level approaches; on the other hand they provide courser detections of the foreground objects and are computationally expensive. Some example of these kind of approaches are the Normalized Vector Distance based approach in and the Local Binary Pattern texture based approach in.

- Region-level based algorithms divide an image into a set of regions which are then classified as background or foreground. There is a very limited number of purely region level based algorithms since finding meaningful regions in an image by means of spatial consistency criteria can be computationally expensive. Therefore, region-level based approaches are usually combined with another kind of approach which is used to determine the regions followed by the region classification itself. Nevertheless, there are some examples of purely region-level based algorithms as the one presented in, which is based on the Partial Directed Hausdorff distance, and the more recently proposed in, where the authors propose to model foreground and background objects by means of Spatial-Color Gaussian Mixture Models.

1.3 Approaches

In this section we provide a brief review of the background subtraction approaches employed in our project. The discussion below provides the basis for the subsequent implementation. The widely used background subtractions methods are:

- Running-average model
- Median Model
- Running Gaussian average model
- Gaussian Mixture model
- Non-parametric models- Kernel density estimation methods

In this project and in the subsequent discussion we focus on two methods; namely: the running-average and Gaussian mixture models.

1.3.1 Running-Average Model

The most basic approach for modelling the background of a video sequence is to average the value X_t observed at time t at every pixel position (x, y) in the consecutive video frames. This average can be approximated by using the following recursive computation:

$$B_{t+1}(x,y) = B_t(x,y) + \alpha\left(X_t - B_t(x,y)\right) \quad \ldots\ldots 1.1$$

Where B_t is the resulting background image and α is a learning rate that controls how fast the background model is adapted to the changes in the scene. The background image, B_t , can be used to compute a difference image D_t, which contains the value of the difference between each pixel in the incoming images $I_t(x,y)$ and their corresponding values in B_t . The foreground mask F_t is generated according to the following decision rule:

$$F_t(x,y) = \begin{cases} 1 & if\ |D_t(x,y)| > \tau \\ 0 & Otherwise \end{cases} \quad \ldots\ldots 1.2$$

where τ is the selected thresholding value. The value of τ should be chosen dynamically so as to adapt to changing viewing conditions, which might affect the noise introduced by the camera. As an extension to the global thresholding procedure formulated in Equation 1.2, there are several procedures that can improve the detection of foreground, such as, e.g., local thresholding or hysteresis thresholding. An early approach using this kind of background model is presented in [4], where a selective updating strategy is used in order to allow for a fast adaptation of the background model while being able to maintain moving objects in the foreground. The used background updating equation is as follows:

$$B_{t+1}(x,y) = B_t(x,y) + \left(\alpha_1\left(1 - F_t(x,y)\right) + \alpha_2 F_t(x,y)\right)\left(X_t - B_t(x,y)\right) \ldots\ldots 1.3$$

1.3.2 Gaussian Mixture Model

The background models presented up to now, use a single description for the appearance of the background. In order to describe more complex distributions, Gaussian Mixture Models (GMMs) can be used. The basic idea of this approach is to classify each pixel by using a model of the appearance of the pixel which consists of the combination of different classes. Figure 2.3 shows an example of the empirical distribution of the intensity values produced by a source consisting of three Gaussian modes with different prior, mean and variance values, and its corresponding GMM. The dominant mode could correspond to the background model of an observed scene, the second could be interpreted as projected shadows, and the third one to the foreground objects passing by.

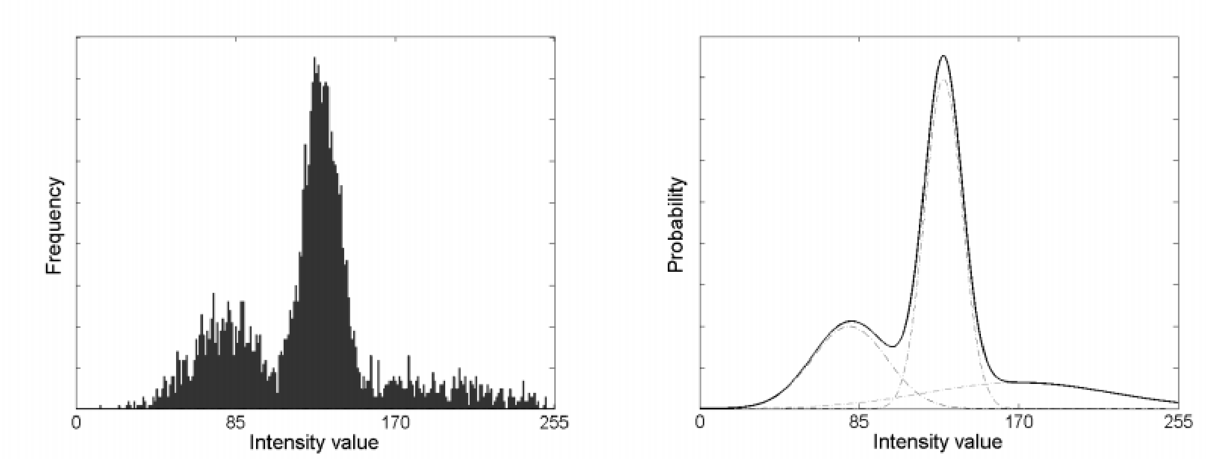

Figure 1.2 Gaussian Mixture Model. Left: empirical distribution of intensity values. Right: corresponding three-components Gaussian Mixture Model

GMMs have the advantage of coping with multi-modal background appearances, as e.g. waving trees, and are able to adapt to the observed scene in real-time with low memory requirements. The original formulation of the GMM for the task of background subtraction was provided in [5], where a mixture of three Gaussian distributions was used to model at a pixel level the appearance of the road, shadows and vehicles in a traffic monitoring application. This system was generalized in [6], and further developed in [7] in order to cope with moving cameras.

1.3.2.1 Background subtraction using Gaussian Mixture Model (GMM)

Background subtraction using GMM models each pixel as a mixture of Gaussians. The GMM model is then evaluated to classify the pixel to background or foreground component [M]. For static background and static lighting, pixel history will be relatively constant. Assuming independent Gaussian noise is incurred during pixel sampling process, pixel history could be described by a single Gaussian distribution centered at its mean value. However, due to presence of moving objects, illumination changes and scene changes, therefore, multimodal representation is crucial in modeling pixel history. This technique results in a stable moving object detector which reliably deals with lighting changes, long term scene changes and repetitive motion clutter.

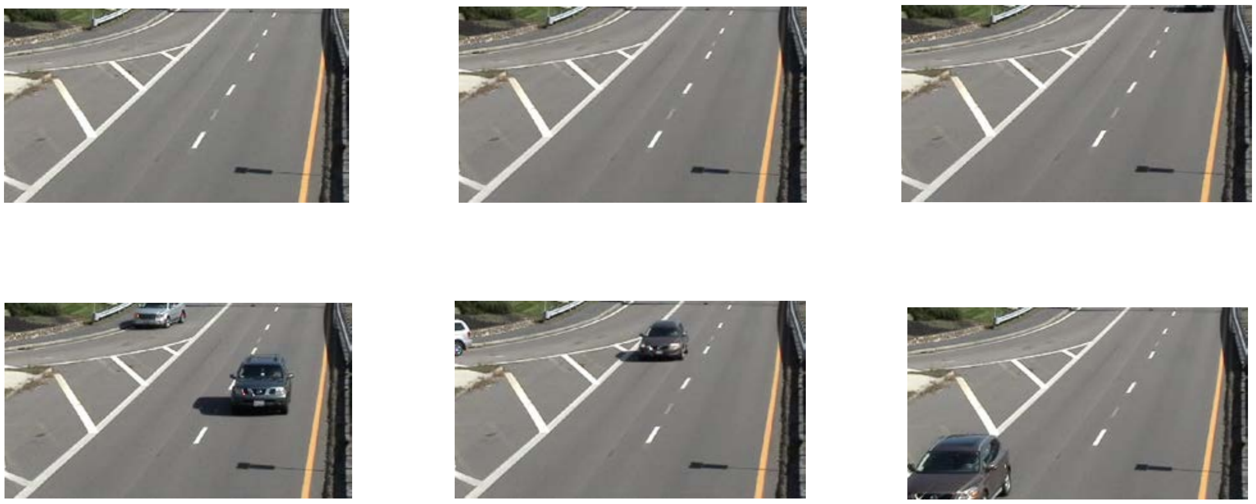

Figure 1.3: Frame sequences in a video where (a) Frame 2, (b) Frame 50, (c) Frame 100, (d) Frame 150, (e) Frame 200, (f) Frame 250

Consider the intensity values of a pixel $\{x_0, y_0\}$ of image sequence I over time t $\{X_1 \dots, X_t\} = \{I(x_0, y_0, i) : 1 \leq i \leq t\}$, then the probability of observing the current pixel value is

$$P(x_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \mathbb{C}_{i,t}), \dots\dots\dots\dots\dots\dots 1.4$$

where K is number of Gaussian distributions, $\omega_{i,t}$ is an estimate of the weight of i_{th} Gaussian in the mixture at time t, is the mean value of the i_{th} Gaussian mixture at time t, $\mathbb{C}_{i,t}$is the covariance matrix of the i_{th} Gaussian in the mixture at time t, and where n is a Gaussian probability density function. The model should track any kind of lighting changes in a static scene. Thus, parameters of Gaussian distributions should be updated online based on recently observed values of each pixel in the scene. A new pixel value will, in general, be represented by one of the major components of the mixture model and used to update the model.

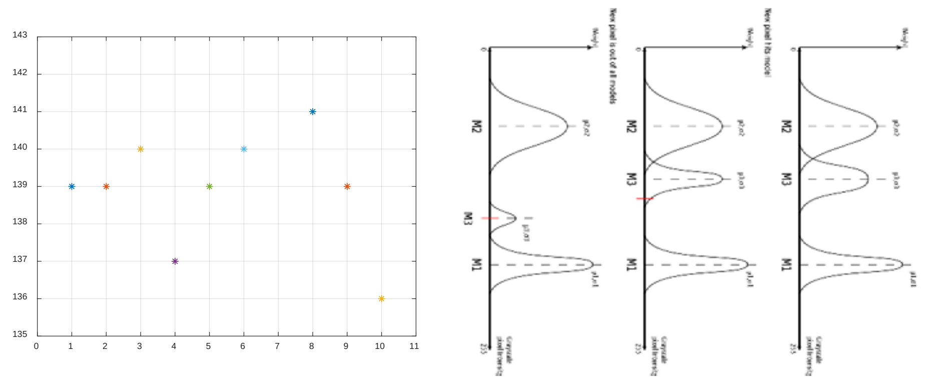

Figure 1.4 : Pixel intensity value variation over time and mixture of Gaussian used to model the pixel process

Considering the time varying characteristics of pixel values and computational cost into account approximated method is essentially used to integrate new data. The prior weights of the K Gaussian distributions at time t are adjusted as follows

$$\omega_{k,t} = (1-\alpha)\omega_{k,t-1} + \alpha * M_{k,t} \;\ldots\ldots\ldots\ldots\ldots\ldots\ldots 1.5$$

Where α is learning rate and $M_{k,t}$ is 1 for the model which matched and 0 for non-matching models. $1/\alpha$ is the time constant which determines the speed at which the Gaussian distribution parameters change. Weights are normally renormalized after every approximation. $\omega_{k,t}$ is equivalent to expectation of posterior probability of pixel values that matched model k given observations from time 1 through t. The μ and σ parameters for unmatched distributions remain the same. For distributions which matches new observations, the parameters are updated as follows

$$\mu_t = (1-p)\mu_{t-1} + pX_t \ldots\ldots\ldots\ldots\ldots\ldots\ldots 1.6$$

$$\sigma_t^2 = (1-p)\sigma_{t-1}^2 + p(X_t-\mu_t)^T(X_t-\mu_t) \ldots\ldots\ldots 1.7$$

Where

$$p = \alpha\eta(X_t|\mu_k, \sigma_k) \ldots\ldots\ldots\ldots\ldots\ldots\ldots 1.8$$

When moving object become static i.e. part of back ground, the Gaussian distribution describing previous back ground still exists with the same μ and σ^2. However, its weight gets lower and lower and the object gets reincorporated into the background.

```
Input: weight, mean, variance, Imageframe, learningrate, p, numberofgaussians
Output: Detected moving object
For each Imageframe
    Compute distance pixeldis of pixel values from mean of their associated components
    For each pixel
        Initialize match flag
        For each gaussian component
            If pixeldis <= deviationthresh*sd
                Set match flag
                Update weight, mean, variance and p
            Else
                Update weight
            End
        End
            Normalize weight
            Update background model:
        If match flag not set
            Find index of minimum weight
            Replace lowest rank mode with a new one with x as mean
            Update mean value of this gaussian
            Update variance of this gaussian
        End
            Compute rank
            Sort rank
            Update foreground
    End
    Remove noise in background
    Draw bounding box around detected objects
    Display result
End
```

Algorithm 1: Moving object detection using mixture of Gaussians model

When detecting a moving object variance of the background is expected to be lower than variance of the moving object, accordingly, ranking is used to decide what portion of model best represents background process.

First, matched Gaussians are ordered based on ω/σ from the less probable to the most probable background distribution. This creates an open-ended list where the most likely background Gaussians occupy top of the list. Then the first B Gaussian distributions are chosen as the background model where B is determined based on portion T of the recent data has been accounted for. T is the threshold quantifying the minimum fraction of the background model i.e. it is the minimum prior probability that the background is in the scene.

```
Input: weight, mean
Output: background component
Initialize background as background
For all Gaussians
background = background + mean*weight
end
```

Algorithm 2: Updating background component

Foreground pixel is any pixel that is more than 2.5 standard deviations away from any of the B distributions. If none of the K distributions match that pixel value, the least probable component is replaced by a distribution with the current value as its mean, an initially variance and a low weight parameter. When the value of P is too small, it leads to slow adaptations in the means and the covariance matrices, which leads to tracker failure. However, simply cutting out the likelihood term from p could rectify the problem.

```
Input: weight, var, pixel, rank, pixeldis
Initialize: matchflag, foreground, backgroundcomponent, foregroundthreshold, deviationthresh, k
Sort the rank in descending order and retrieve indices
While matchflag = 0 and k <= backgroundcomponent
If weight(pixel,rank(k)) >= foregroundthreshold
If pixeldis(pixel,rank(k)) <= deviationthresh*var(pixel,rank(k))
Matchflag = 1
Else
Foreground = pixel
End
End
End
```

Algorithm 3: Foreground pixel estimation

After identifying foreground pixels in each new frame image morphology operation is deployed to remove noise pixels while strengthening the true foreground pixels. First, non-connected components are removed by morphological erosion. Then, the remaining components are dilated using morphological dilation to account for wrongly removed foreground pixels.

2 Simulation & Results

2.1 Simulation Description

For simulation we consider two methods known as running average and gaussian mixture model based method. We have used 3 MATLAB buildin prototype video. The names of those videos are 'visiontraffic.avi', 'car-perspective-3-hires.m4v' and 'atrium.mp4'. In these three videos different type of objects like human, car, toy-car etc., are moving and our algorithms successfully detects them. In the following section our approach is discussed and simulation results are shown.

2.1.1 Implementation I – Running Average

For running average method, we need to predefine some parameter. The list of parameters is shown in the following table. And the output image for different phase is shown in figure 1-6.

Parameter	**Value**	**Remark**
Window length	100	The no. of images that will be used to estimate the background.
Threshold	50	Used to convert gray level image to binary image
Median filter size	5x5	Used to denoising the image
Blob size	20x20	Used to dilute the mask of the foreground image

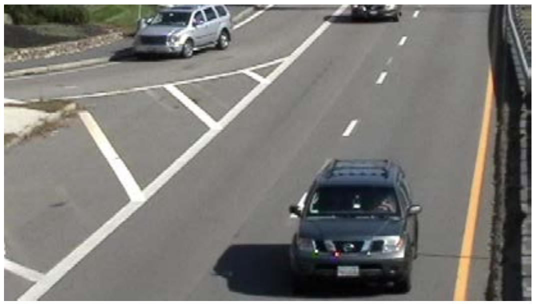

Figure 1: Current image frame

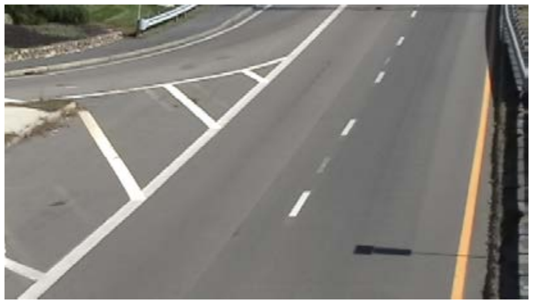

Figure 2: Estimated background image

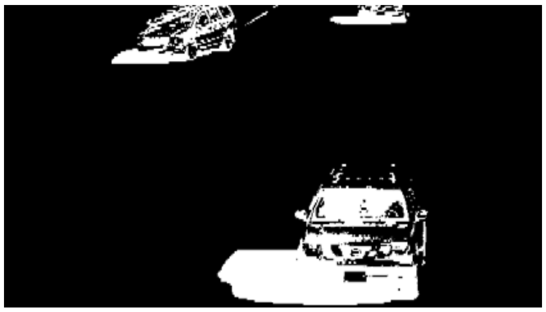

Figure 3: Subtracted binary mask image

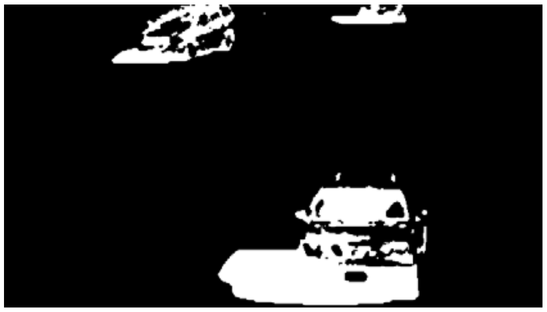

Figure 4: Filtered binary mask image

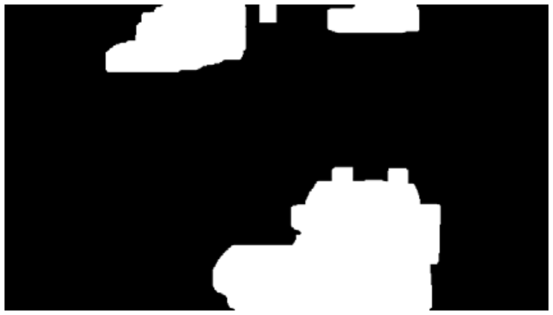

Figure 5: Diluted binary mask image

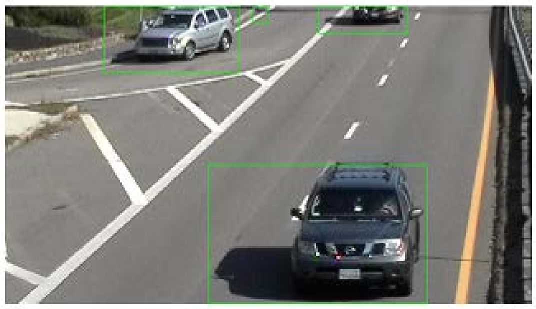

Figure 6: Output image with detected object

2.1.2 Implementation and result analysis of GMM based detection

Gaussian mixture model is a parametric model. Its detection performance depends on parameters such as learning rate, back ground component threshold, foreground threshold, number of Gaussians and position of camera i.e. when camera is nearer and far away. The value parameters of Gaussian mixture model used in this work to detect moving objects is tabulated below

Nº	Parameter	Value	Remark
1	Number of Gaussians	3	Can vary from 3 to 5
2	Background component	3	Prior probability of background
3	Deviation threshold	2.5	
4	Learning rate	0.01	Can vary from 0 to 1
5	Initial standard deviation	6	
6	Intensity level	8 bit	
7	Foreground threshold	0.25	Threshold on weight parameter

When there are multiple moving objects in a scene, illumination of the scene changes than when there is single object. The objects could also occlude each other. Some of the objects might suddenly stop or join other objects in the scene. The detection is robust to presence of occlusions, presence of multiple objects. However, it takes a few seconds before moving objects which was stationary to be included in the foreground and similarly for stationary objects which was moving. When there is occlusion, an object can be detected as two or more objects while two objects can be detected as single object if they are very close to each other. Our algorithm is tested on built in videos such as 'visiontraffic.avi' and 'atrium.mp4' in MATLAB and has shown very good performance.

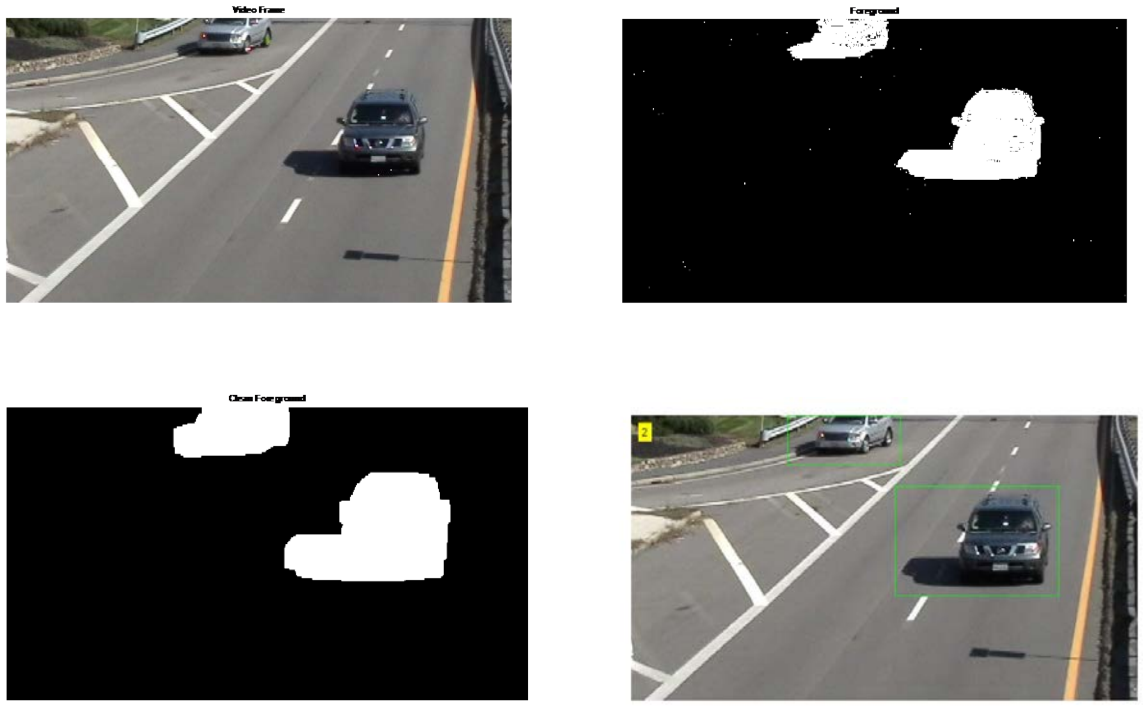

Figure 2.1: Detecting moving car: (a) Input frame, (b) Foreground object, (c) Filtered foreground object, (d) Detected car

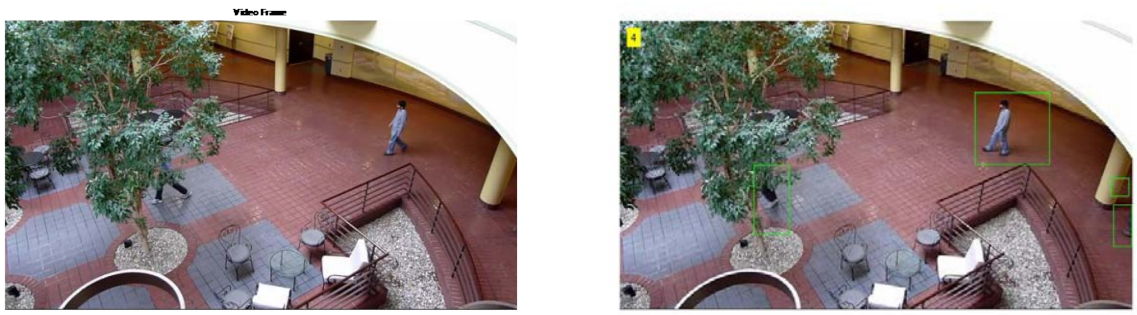

Figure 2.2.: Detecting a walking person: (a) Input frame, (b) detected person

2.2 Conclusion

In this project, different background subtraction techniques are applied to detect moving object(s). Comparable results are found from the employed techniques. We have also compared the performance of our implementation to performance of built in matlab function vision.ForegroundDetector. the Approximate moving average method has similar computational performance to that of built in matlab function although it introduces more false detections proportionally. Our gaussian mixture model implementation has similar detection performance to that of matlab function. However, computationally it is very slow.

2.3 Future works

In this work shadow detection and removal is not considered. Gaussian mixture model parameters are manually set. It doesn't recognize the moving object.

Annex: MATLAB Source Code

Running average method

```
% ECEN-657 Digital Image Processing
%-------------------------------------------------------------------------
% MATLAB Final Project
%-------------------------------------------------------------------------
% To be submitted to: Dr. Jung H Kim
%-------------------------------------------------------------------------
% By: Biniam T. Gebru
%     Mrinmoy Sarkar
%     Solomon Genene
%-------------------------------------------------------------------------
%   Spring, 2018
%-------------------------------------------------------------------------
%% Built in prototype videos in matlab
% Try these videos as argument to VideoReader
% 'visiontraffic.avi'
% 'car-perspective-3-hires.m4v
% 'atrium.mp4'
%% clear and prepare workspace
clc
clear all
close all
%% Read video from a file
v = VideoReader('visiontraffic.avi');%'atrium.mp4');%'visiontraffic.avi''car-
perspective-3-hires.m4v 'atrium.mp4'
videoPlayer = vision.VideoPlayer('Name', 'Detected object');
videoPlayer.Position(3:4) = [700,400];  % window size: [width, height]
%% Initialize detection parameters
video = readFrame(v);
M = size(video,1);
N = size(video,2);
windowlen=100;
th = 50;
oldImg = zeros(M,N,3,windowlen);
frameno = 0;
%% initialize detection tools
blobAnalysis = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...
     'AreaOutputPort', false, 'CentroidOutputPort', false, ...
     'MinimumBlobArea', 150);
%% Process frames
while  hasFrame(v)
    % read frame
    video = readFrame(v);
    figure(1)
    imshow(video)
    % Model background
    indx = mod(frameno,windowlen)+1;
    oldImg(:,:,:,indx) = double(video);
    background = uint8(sum(oldImg,4)/windowlen);
    figure(2)
    imshow(background)
    frameno = frameno+1;
    % Subtract background: generates fore ground image
    maskimg = uint8(abs(double(video)-double(background)));
    maskgray = rgb2gray(maskimg);
    maskbw = maskgray > th;
    figure(3)
    imshow(maskbw)
    % Use morphological opening to remove noise in the foreground
    FilteredImage=medfilt2(maskbw,[5 5]);
    figure(4)
    imshow(FilteredImage)
    % connect disconnected foreground components
    SE = strel('rectangle',[20 20]);
    dilutatedmask = imdilate(FilteredImage,SE);
    figure(5)
    imshow(dilutatedmask)
    % Detect the connected components with the specified minimum area,
    bbox = step(blobAnalysis, dilutatedmask);
    % Draw bounding boxes around the detected objects
    result = insertShape(video, 'Rectangle', bbox, 'Color', 'green');
    % Display detected object
    figure(6)
    imshow(result)
    step(videoPlayer, result)
%     if frameno==165
%         break;
%     end
end
```

GMM method

```
% ECEN-657 Digital Image Processing
%-------------------------------------------------------------------------
% MATLAB Final Project
%-------------------------------------------------------------------------
% To be submitted to: Dr. Jung H Kim
%-------------------------------------------------------------------------
% By: Biniam T. Gebru
%     Mrinmoy Sarkar
%     Solomon Genene
%-------------------------------------------------------------------------
%      Spring, 2018
%-------------------------------------------------------------------------
%% Built in prototype videos in matlab
% Try these videos as argument to VideoReader
% 'visiontraffic.avi'
% 'car-perspective-3-hires.m4v
% 'atrium.mp4'
%% clear and prepare workspace
clc
clear all
close all
%% Read video from a file
```

```
v = VideoReader('visiontraffic.avi');%'visiontraffic.avi''car-perspective-3-hires.m4v
'atrium.mp4'
videoPlayer = vision.VideoPlayer('Name', 'Detected object');
videoPlayer.Position(3:4) = [700,400];  % window size: [width, height]
%% Initialize detection parameters
video = readFrame(v);
M = size(video,1);
N = size(video,2);
windowlen=100;
th = 50;
oldImg = zeros(M,N,3,windowlen);
frameno = 0;
%% initialize detection tools
blobAnalysis = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...
     'AreaOutputPort', true, 'CentroidOutputPort', true, ...
     'MinimumBlobArea', 150);
 centroid =[];
%% Process frames
while  hasFrame(v)
    % read frame
    video = readFrame(v);
    % Model background
    indx = mod(frameno,windowlen)+1;
    oldImg(:,:,:,indx) = double(video);
    background = uint8(sum(oldImg,4)/windowlen);
    frameno = frameno+1;
    % Subtract background: generates fore ground image
    maskimg = uint8(abs(double(video)-double(background)));
    maskgray = rgb2gray(maskimg);
    maskbw = maskgray > th;
    % Use morphological opening to remove noise in the foreground
    FilteredImage=medfilt2(maskbw,[5 5]);
    % connect disconnected foreground components
    SE = strel('rectangle',[20 20]);
    dilutatedmask = imdilate(FilteredImage,SE);
    % Detect the connected components with the specified minimum area,
    [~,cc,bbox] = step(blobAnalysis, dilutatedmask);
    centroid =[centroid; cc];
    % Draw bounding boxes around the detected objects
    result = insertShape(video, 'Rectangle', bbox, 'Color', 'green');
    % Display detected object
    step(videoPlayer, result)
end
plot(centroid(:,1),centroid(:,2),'>')
figure
scatter(centroid(:,1),-1.*centroid(:,2),'>')


%% This is updated version: originally written by Ben Sethon
%% Built in prototype videos in matlab
% Try these videos as argument to VideoReader
% 'visiontraffic.avi'
% 'car-perspective-3-hires.m4v
% 'atrium.mp4'
%% Clear and prepare workspace
clc
close all
clear all
%% Read video from file
v = VideoReader('visiontraffic.avi');
fr_bw = readFrame(v);
%% initialize foreground and background
[height, width,~] = size(fr_bw);
foreground = zeros(height, width);
% background = zeros(height, width);
```

```
%% set video player
videoPlayer = vision.VideoPlayer('Name', 'Detected object');
videoPlayer.Position(3:4) = [700,400];  % window size: [width, height]

%% Initialize mixture of gaussian model parameters
numberofgaussian = 3;
backgroundcomponent = 3;
deviationthreshold = 2.5;
learningrate = 0.01;
foregroundthreshold = 0.25;
initialstandarddev = 6;
d = zeros(height,width,numberofgaussian);              % distance of each pixel from mean
p = learningrate/(1/numberofgaussian);                 % initialize p
rank = zeros(1,numberofgaussian);                      % rank of components weight and
standard deviation
%% Initialize weight, Mean, standard deviation
intensitylevel = 255;
M = rand(height,width,numberofgaussian)*intensitylevel;
w = ones(height,width,numberofgaussian)./numberofgaussian;
sd = initialstandarddev.*ones(height,width,numberofgaussian);
%% Initialize boundary box parameters
se = strel('square', 4); % morphological filter for noise removal
se1 = strel('square',20);
blobAnalysis = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...
    'AreaOutputPort', false, 'CentroidOutputPort', false, ...
    'MinimumBlobArea', 150);
%% process each frame
while  hasFrame(v)

    fr_bw = readFrame(v);
    % compute distance of pixel from gaussian mean
    d = abs(double(fr_bw)-M);
    % for each pixel
    for i=1:height
        for j=1:width
            matchflag = 0;

            for k=1:numberofgaussian
                if (abs(d(i,j,k)) <= deviationthreshold*sd(i,j,k))
                    matchflag = 1;
                    % update weights, mean, sd, p
                    w(i,j,k) = (1-learningrate)*w(i,j,k) + learningrate;
                    p = learningrate/w(i,j,k);
                    M(i,j,k) = (1-p)*M(i,j,k) + p*double(fr_bw(i,j));
                                        sd(i,j,k) =   sqrt((1-p)*(sd(i,j,k)^2) +
                    p*((double(fr_bw(i,j)) - M(i,j,k)))^2);
                else
                    w(i,j,k) = (1-learningrate)*w(i,j,k);
                end
            end
            %% Normalize the weight
            w(i,j,:) = w(i,j,:)./sum(w(i,j,:));
            %% Update the background
%             background(i,j)=0;
%             for k=1:numberofgaussian
%                 background(i,j) = background(i,j)+ M(i,j,k)*w(i,j,k);
%             end
            %% if no match
            % if no components match, create new component
            if (matchflag == 0)
                [min_w, min_w_index] = min(w(i,j,:));
                M(i,j,min_w_index) = double(fr_bw(i,j));
                sd(i,j,min_w_index) = initialstandarddev;
            end
            %% compute and sort the rank
            rank = w(i,j,:)./sd(i,j,:);
%             rank_ind = [1:1:numberofgaussian];
            [rank, rank_ind]=sort(rank,'descend');

            %% determine foreground
            matchflag = 0;
            k=1;
            foreground(i,j) = 0;
            while ((matchflag == 0)&&(k<=backgroundcomponent))
                if (w(i,j,rank_ind(k)) >= foregroundthreshold)
                    if (abs(d(i,j,rank_ind(k))) <=
deviationthreshold*sd(i,j,rank_ind(k)))
                        foreground(i,j) = 0;
                        matchflag = 1;
                    else
                        foreground(i,j) = fr_bw(i,j);
                    end
                end
                k = k+1;
            end

        end
    end
%     break;
    %% Use morphological opening to remove noise in the foreground
    filteredForeground = imopen(foreground, se);
    filteredForeground = imdilate(logical(filteredForeground), se1);
    bbox = step(blobAnalysis, filteredForeground);

    %% Draw bounding boxes around the detected objects
    result = insertShape(fr_bw, 'Rectangle', bbox, 'Color', 'green');

    step(videoPlayer, result)

end


%% Built in prototype videos in matlab
% Try these videos as argument to VideoReader
% 'visiontraffic.avi'
% 'car-perspective-3-hires.m4v
% 'atrium.mp4'
%% clear and prepare workspace
clc
clear all
close all
%% Train the detector
foregroundDetector = vision.ForegroundDetector('NumGaussians', 3,'NumTrainingFrames',
50);
videoReader = vision.VideoFileReader('atrium.mp4');%'visiontraffic.avi',atrium.mp4,'car-
perspective-3-hires.m4v'
for i = 1:150
    frame = step(videoReader); % read the next video frame
    foreground = step(foregroundDetector, frame);
end
figure; imshow(frame); title('Video Frame');
figure; imshow(foreground); title('Foreground');
se = strel('square', 3);
filteredForeground = imopen(foreground, se);
se = strel('square',20);
filteredForeground = imdilate(filteredForeground, se);
figure;
imshow(filteredForeground);
title('Clean Foreground');
%% Initialize detection components
```

```
blobAnalysis = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...
'AreaOutputPort', true, 'CentroidOutputPort', true, ...
     'MinimumBlobArea', 150);
% bbox = step(blobAnalysis, filteredForeground);
% result = insertShape(frame, 'Rectangle', bbox, 'Color', 'green');
% numObjects = size(bbox, 1);
% result = insertText(result, [10 10], numObjects, 'BoxOpacity', 1,'FontSize', 14);
% figure;
% imshow(result);
% title('Detected objects');
%% Initialize video player
videoPlayer = vision.VideoPlayer('Name', 'Detected object');
videoPlayer.Position(3:4) = [700,400];  % window size: [width, height]
se = strel('square', 4); % morphological filter for noise removal
se1 = strel('square',20);
centroid =[];
%cam = webcam;
%for idx = 1:100
%% Process frames
while ~isDone(videoReader)

    frame = step(videoReader); % read the next video frame
    %frame = snapshot(cam);

    % Detect the foreground in the current video frame
    foreground = step(foregroundDetector, frame);

    % Use morphological opening to remove noise in the foreground
    filteredForeground = imopen(foreground, se);

    filteredForeground = imdilate(filteredForeground, se1);

    % Detect the connected components with the specified minimum area,
    [~,c,bbox] = step(blobAnalysis, filteredForeground);
    centroid =[centroid; c];
    % Draw bounding boxes around the detected objects
    result = insertShape(frame, 'Rectangle', bbox, 'Color', 'green');

    % Display the number of objects found in the video frame
    numObjects = size(bbox, 1);
    result = insertText(result, [10 10], numObjects, 'BoxOpacity', 1,'FontSize', 14);

    step(videoPlayer, result);  % display the results
end

release(videoReader); % close the video file
figure
plot(centroid(:,1),centroid(:,2),'>')
figure
scatter(centroid(:,1),-1.*centroid(:,2),'>')
%clear('cam')
```

Reference

1. Toyama, K., Krumm, J., Brumitt, B., and Meyers, B. (1999). Wallflower: *Principles and Practice of Background Maintenance.* In Proceedings of the IEEE International Conference on Computer Vision, volume 1, page 255, Los Alamitos, CA, USA. IEEE Computer Society.

2. Brutzer, S., Hoferlin, B., and Heidemann, G. (2011). *Evaluation of Background Subtraction Techniques for Video Surveillance.* In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), pages 1937–1944, Colorado Spring, USA.

3. Wren, C., Azarbayejani, A., Darrell, T., and Pentland, A. (1997). Pfinder: *Real-time tracking of the human body.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 19:780– 785.

4. Kilger, M. (1992). *A shadow handler in a video-based real-time traffic monitoring system.* In Proceedings of the IEEE Workshop on Applications of Computer Vision, pages 11–18.

5. Friedman, N. and Russell, S. (1997). *Image segmentation in video sequences: A probabilistic approach.* In Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI).

6. Stauffer, C. and Grimson, W. (1999). *Adaptive Background Mixture Models for Real-time Tracking.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, volume 2, pages 246–252, Fort Collins, CO, USA.

7. Hayman, E. and Eklundh, J. (2003). *Statistical Background Subtraction for a Mobile Observer.* In Proceedings of the International Conference on Computer Vision, pages 67–74.