Background-Foreground Frame Classification

CS771A: Machine Learning Techniques
Project Report

Advisor: Prof. Harish Karnick

Akhilesh Maurya (12066) Deepak Kumar (12228) Jay Pandya (12319) Rahul Mehra (12537)
Group-21

April 10, 2016

Abstract

A large number of cameras are installed at the gate of IIT Kanpur for the purpose of video surveillance. All the video data is stored and hence adds up to the disk space used for storage. All the video frames which are recorded during the non-peak hours have very less number of vehicles in them and most of the frames have none. There is a need to remove these background frames from the videos so that we can reduce the overhead of storing the unwanted frames. For this we propose our methods for removal of background frames from the given surveillance video.

1 Introduction

In this project we present three methods to do background-foreground classification of the frames of the video. One is based on simple frame differencing in the video. Another uses CNN to extract the features from any frame and trains a model for the classification. The last method uses creation of a code-book model for the classification of the pixels.

2 Given Input Data

We were provided with 10 video clips. 3 with lengths 15min and the rest had length 1min. All these clips were of the morning period and hence were very clear and had nice intensity throughout the length of the video. We were also provided with the labelled bounding boxes around the objects within each frame of the video. This data was highly ambiguous as one of the larger videos with length 15min had lot of frames which had no bounding box around the moving vehicles. Also there was lot of difference in the labelling done by different people. Also two of the videos had two security officers roaming throughout the video which made our classification task a bit difficult. Hence we didn't use this data and labelled the frames on our own. We labelled the given video data as background or foreground and trained our models based on that.

3 Frame-Differencing

Extraction of moving object is an important step in the system of video surveillance, traffic monitoring, human tracking and other applications

Algorithm of the frame differencing:

A motion detection algorithm begins with the segmentation part where foreground or moving objects are segmented from the background [1]. The simplest way to implement this is to take an image as background and take the frames obtained at the time t, denoted by I(t) to compare with the background image denoted by B. Before this the images are converted into grayscale rather than colored. Here using simple arithmetic calculations, we can segment out the objects simply by using image subtraction technique of computer vision meaning for each pixels in I(t), take the pixel value denoted by P[I(t)] and subtract it with the corresponding pixels at the same position on the background image denoted as P[B].

In mathematical equation, it is written as:

$$P[F(t)] = P[I(t)] - P[B] \tag{1}$$

The background is assumed to be the frame at time t. This difference image would only show some intensity for the pixel locations which have changed in the two frames as shown in Fig1. Though we have seemingly removed the background, this approach will only work for cases where all foreground pixels are moving and all background pixels are static.A threshold "Threshold" is put on this difference image to improve the subtraction.

$$|P[F(t)] - P[F(t+1)]| > Threshold \tag{2}$$

This means that the difference image's pixels' intensities are 'thresholded' or filtered on the basis of value of Threshold. The accuracy of this approach is dependent on speed of movement in the scene. Faster movements may require higher thresholds.

We used the Mean and Median of the subtracted image as elements of the feature vector. Feature vectors were obtained for each frame and used to train Perceptron/SVM using the manually labeled data. The perceptron/SVM calculated the separating conditions for the feature vector thereby giving us a learned threshold for separating background and foreground frames.

Problems Faced:

- **Static object can't be recognised** The feature vector of pixels having static object is zero so these frames are recognised as background frames.Also very slowly moving object are not classified upto appreciable accuracy.Frames get detected once they start moving.

Improvement in accuracy:

- Frame differencing with overall average frame gives feature vector which gives better accuracy for our video data.
- For videos of longer length we can take hourly image average and then taking frame differencing with this average.

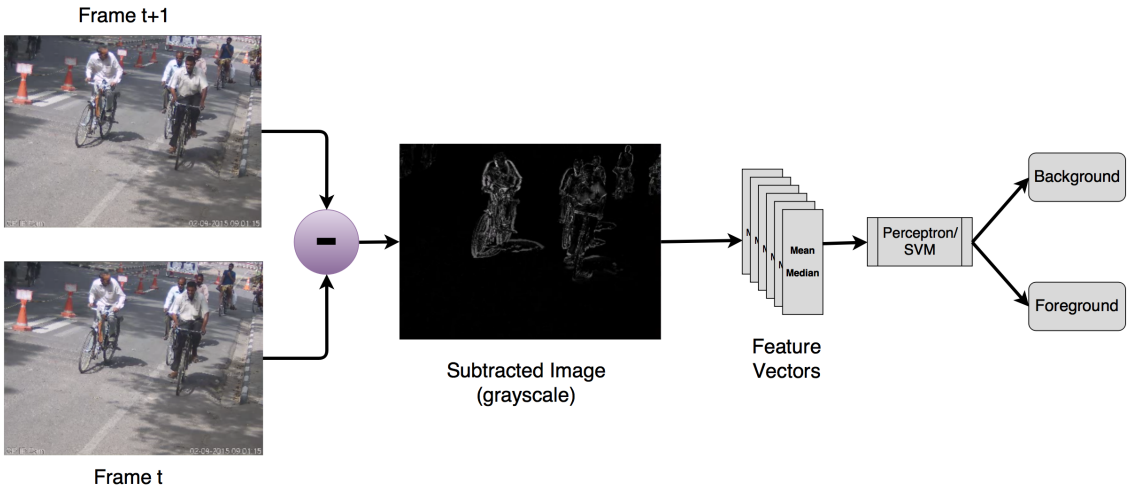

Figure 1: Frame Differencing

4 CNN Based Feature Extraction

In this method the choice of background is the actual background having no moving/still object in them. We do so because this method takes no consideration to the previous frames in any sense. Each frame of image is fed to the CNN for feature extraction. We do not incorporate the details and information about the previous frame or future frame. So to evaluate our method we used the data from video clips having no vehicles standing in them. Since we had already labeled them as background frame in the previous method.

We use pre-trained caffe model of BVLC GoogLeNet[Insert Reference] for our purpose [3]. We treat this model as a black box that gives us features to train another model for classification. The CNN has 22 layers in it as shown in Fig 2. The actual caffe model is trained to classify the images into the 1000 ILSVRC 2014 classes (the dataset on which it is trained). But for our purpose we use the 'pool5/7x7_s1' layer of the caffemodel to extract our features. This layer is of length 1024. We use these values as our features. We pass each frame through the CNN and create the feature vector

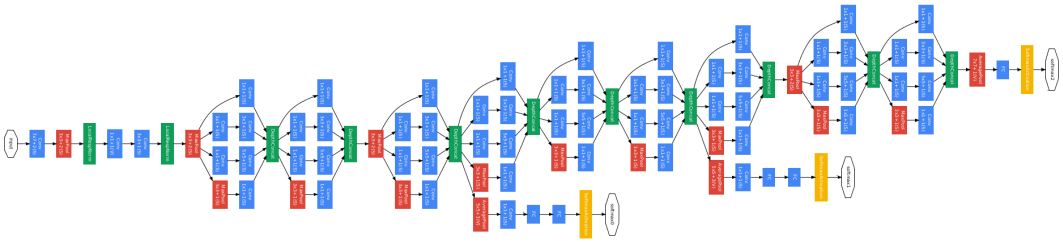

Figure 2: GoogLeNet Layer Structure

of length 1024 for each frame. Now we train a LinearSVM over these feature vectors received and obtain a classifier for our need. We get a very fast feature extraction with this method and also

the classification is very fast. We can do classification of frames at very high speed. Figure 3 summarizes our method.

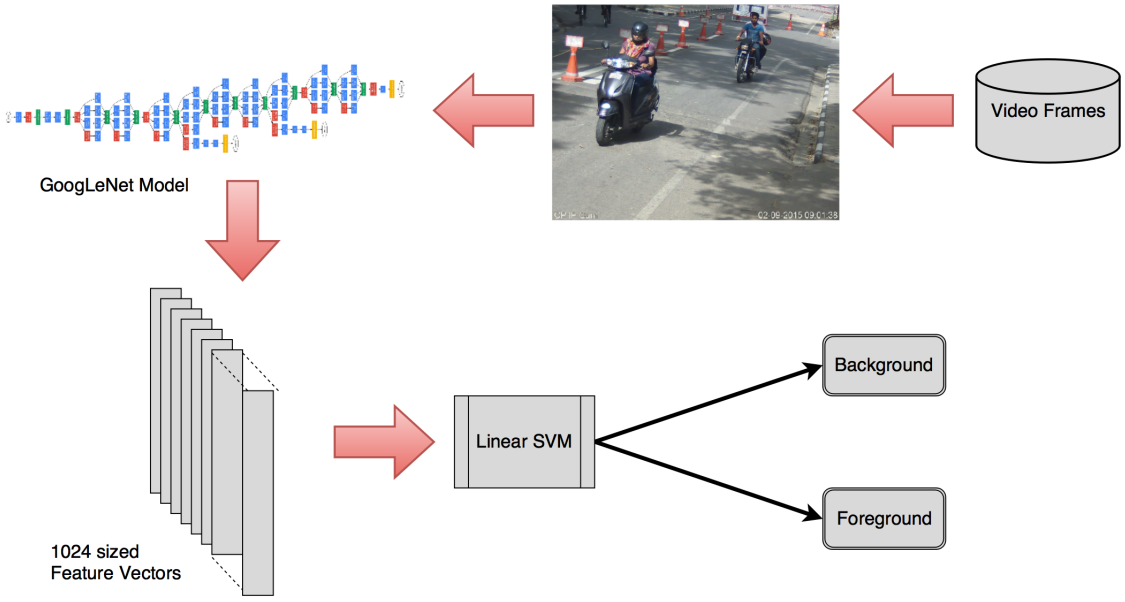

Figure 3: Frame Classification at Test time

Problems Faced:

- **Slow local feature Extraction** The extraction of 1024 sized feature vector on local machine was extremely slow with extraction speed of 0.16 frames per second. Hence to solve this problem we rented a ubuntu 14.04 instance on Amazon Web Service with NVIDIA GPU support. This increased the speed of our feature extraction and made it to around 11 frames per second. Which is still extremely slow and is not practical for real time classification of images.
- **Slow remote AWS feature Extraction** To solve the speed issue we thought of manipulating the frame itself and resizing it. On resizing the frames we found that most of the motion and object features were captured even in the resized image. We resized the image to 1/5th of its original size. On this resized image, the speed changed to 220 frames per second which is extremely fast given that most of our videos had only 25fps video speed. This is a very huge increase as shown in 4.

AWS Machine Specification

- OS - Ubuntu 14.04
- Memory - 15GB
- GPU - 1x NVIDIA GRID (Kepler G104) + 8 x hardware hyperthreads from Intel Xeon E5-2670

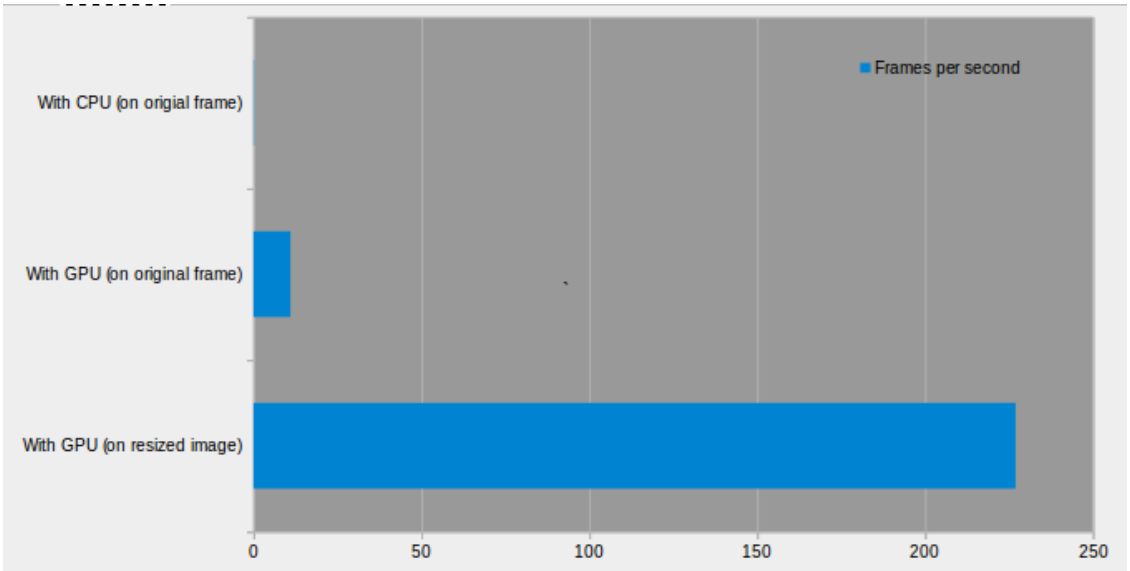

Figure 4: Feature extraction speed on various platforms

5 Code-Book Creation

CodeBook is a method for creation of a model for background. Using the given frames we try to estimate if the current pixel value at a given position has occured till now in the video [2]. There is an underlying assumption that pixels corresponding to background region will occur periodically in video. The method tries to use this to model the background. The algorithms evaluates the color and brightness distortion for the pixels of the image to create the codewords to be added to the code-book. A few nice things about this method are that it can cope up with illumination changes. Also the training doesn't require labeled data. This is an algorithm for fast background modeling and background subtraction.

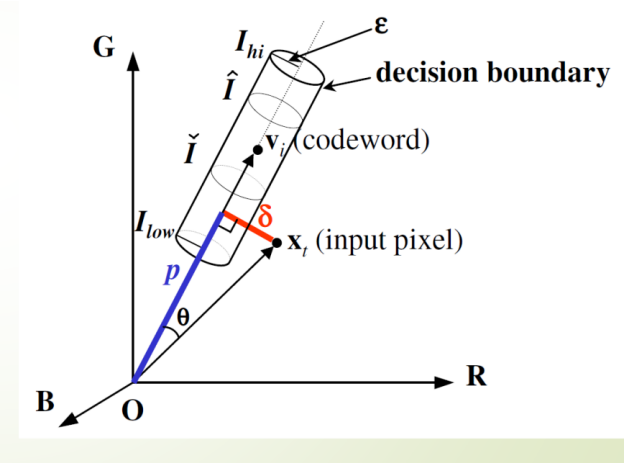

Figure 5: CodeBook depiction

For a N frame long video a code book is created of length L codewords for each pixel. Each pixel has a different code-book size based on its sample variation. Each codewords $c_i, i = 1..L$, consists of an RGB vector $v_i = (\bar{R}_i, \bar{G}_i, \bar{B}_i)$ and a 6-tuple $aux_i = \langle \check{I}_i, \hat{I}_i, f_i, \lambda_i, p_i, q_i \rangle$.

5.1 Algorithm for Codebook construction

1. $L \leftarrow 0, C \leftarrow \emptyset$
2. *for* $t = 1$ to N *do*
   (a) $x_t = (R, G, B), I \leftarrow R + G + B$
   (b) Find the codeword c_m in $C = c_i | 1 \leq i \leq L$ matching to x_t based on two conditions
      i. $colordist(x_t, v_m) \leq \epsilon 1$
      ii. $brightness(I, \langle \check{I}, \hat{I} \rangle) = true$
   (c) If $C \neq \emptyset$ or there is no match, then $L \leftarrow L + 1$. Create a new c_L by setting
      i. $v_L \leftarrow (R, G, B)$
      ii. $aux_L \leftarrow \langle I, I, 1, t - 1, t, t \rangle$
   (d) Otherwise update the matched codeword c_m, consisting of $v_m = (R_m, G_m, B_m)$ and $aux_m = \langle \check{I}, \hat{I}, f_m, \lambda_m, p_m, q_m \rangle$, by setting
      i. $v_m \leftarrow (\frac{f_m R_m + R}{f_m + 1}, \frac{f_m G_m + G}{f_m + 1}, \frac{f_m B_m + B}{f_m + 1})$
      ii. $aux_m \leftarrow \langle min\{I, \check{I_m}\}, max\{I, \hat{I_m}\}, f_m + 1, max\{\lambda_m, t - q_m\}, p_m, t \rangle$.
   end for
3. For each codeword $c_i, i = 1..L$, wrap around λ_i by setting $\lambda_i \leftarrow max\{\lambda_i, (N - q_i + p_i - 1)\}$.

Temporal filtering
$M = \{c_m | c_m \, \epsilon \, C \, \Lambda \, \lambda_m \leq T_M\}$

5.2 Algorithm for Background Subtraction

1. $x = (R, G, B), I \leftarrow R + G + B$
2. For all codewords in M after temporal filtering, find the codeword c_m matching to x based on two conditions:
   (a) $colordist(x, v_m) \leq \epsilon 2$
   (b) $brightness(I, \langle \check{I}, \hat{I} \rangle) = true$

$$BGS(x) = \begin{cases} foreground & \text{if there is no match} \\ background & \text{otherwise} \end{cases}$$

5.3 Problems faced

Code written in python for codebook was very slow. Even the optimizations of reducing the size using random points instead of all the points didn't speed the process much. It seemed promising. But the training to create the code-book was extremely memory intensive and we were not able to do that.

6 Results

We got extremely good results with our methods. Although we were not able to go above 90% we got all our working methods to score above 75%. Frame differencing worked much better than expected and gave very precise classification of frames. An attached video along with this report shows the result. The CNN based method for extraction also performed extremely well. This only means that the features extracted from the 'pool5/7x7_s1' layer captured the features of the image properly. We faced lot of problems with caffe installation on AWS but it was worth the accuracy and speed that we obtained. We were not able to prepare the model for code-words.

The following are the precision obtained for various algorithms:

Method Used	**Performance**
CNN based feature Extraction + SVM Classification	85%
Frame Differencing (with previous frame)	76%
Frame Differencing (with video avg frame)	83%

References

[1] D Stalin Alex and AMITABH WAHI. Bsfd: Background subtraction frame difference algorithm for moving object detection and extraction. *Journal of Theoretical & Applied Information Technology*, 60(3), 2014.

[2] Kyungnam Kim, Thanarat H Chalidabhongse, David Harwood, and Larry Davis. Real-time foreground–background segmentation using codebook model. *Real-time imaging*, 11(3):172–185, 2005.

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.