CS771A Project Report

Group Number 28

Group Members:

1. Utkarsh Agarwal(13754)
2. Kumar Manvendra(13363)
3. Keshaw Singh(13347)
4. Angad Chandhok(13111)

Objective

- Detect and classify objects in the traffic surveillance video into pedestrians and vehicles
- Further classify the vehicles into:
    - Bicycle
    - Motorcycle
    - Car
    - Rickshaw
    - Autorickshaw

Overview

The total work done in our project can be mainly divided into the following four tasks:

1. **Feature extraction:** We extract different features such as HOG and SIFT for the labeled training data and detected objects from the video.
2. **Training the classifier:** We train different types of classifier based on the different features extracted from the training data and chose the best classifier-feature combination for further use.
3. **Foreground-Background separation:** We separate the foreground and background of a frame of the video and segment the foreground into objects and thus get the detected objects.
4. **Classification of the detected objects:** We extract the chosen type of feature(from task no. 2) of the detected objects and then classify the objects using the classifier.

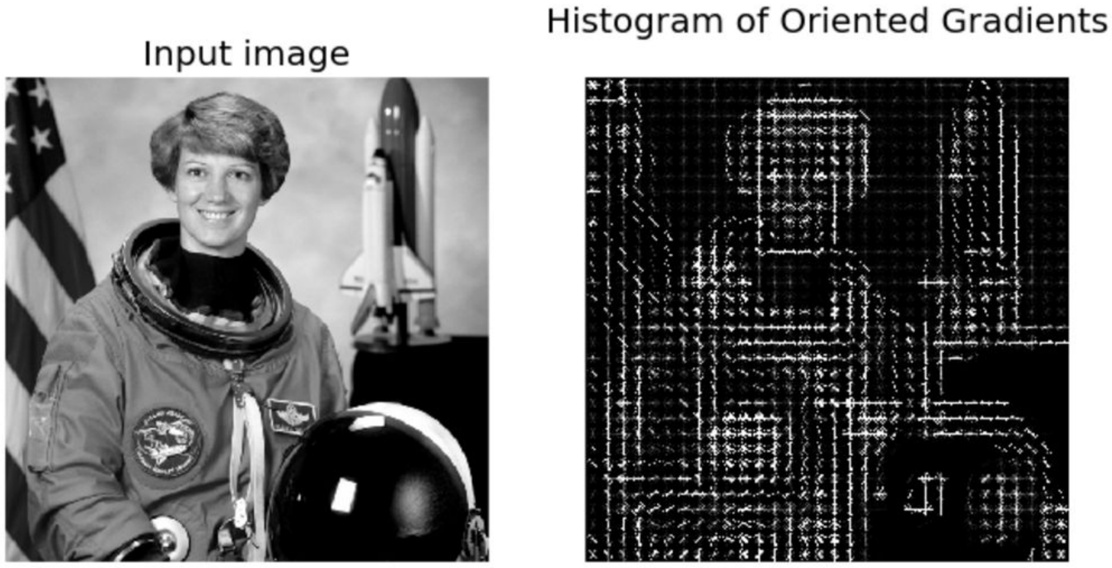
Each of the above mentioned tasks are explained later in this report.

1. Feature extraction

Mapping an image directly to a numpy array for feature extraction is not a good idea. This method is highly inefficient as the feature representation varies highly with changes in geometry and scaling. So same car in two different images with different sizes and different photometric conditions is depicted very differently which has drastic impact on the performance of the classifier. So to account for such and many other factors we use some standard feature descriptors which are well known for object detection.

a. HOG(Histogram of Oriented Gradients):

The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image.
The essential thought behind the HOG descriptor is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled. The descriptor is then the concatenation of these histograms. For improved accuracy, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this value to normalize all cells within the block. This normalization results in better invariance to changes in illumination and shadowing.
The HOG descriptor has a few key advantages over other descriptors. Since it operates on local cells, it is invariant to geometric and photometric transformations, except for object orientation. Such changes would only appear in larger spatial regions.



Since the number of HOG features extracted from an image depends on the number of pixels in it, we resized every training and testing image of objects into 100 X 100 images before extracting HOG features from them.
We experimented with various values of HOG parameters and different classifiers to choose the one which gives best results.

b. SIFT(Scale-Invariant Feature Transform)

Scale-invariant feature transform (or SIFT) is an algorithm in computer vision to detect and describe local features in images. Applications include object recognition, robotic mapping and navigation, image stitching, 3D modeling, gesture recognition, video tracking, individual identification of wildlife and match moving.

For any object in an image, interesting points on the object can be extracted to provide a "feature description" of the object. This description, extracted from a training image, can then be used to identify the object when attempting to locate the object in a test image containing many other objects. To perform reliable recognition, it is important that the features extracted from the training image be detectable even under changes in image scale, noise and illumination. Such points usually lie on high-contrast regions of the image, such as object edges.
SIFT can robustly identify objects even among clutter and under partial occlusion, because the SIFT feature descriptor is invariant to uniform scaling, orientation, and partially invariant to affine distortion and illumination changes.
The number of SIFT features of different images are not always same. So, we quantized the SIFT features by used **K-Means Clustering**. We experimented with various number of clusters to get optimal results.'

2. Training the classifier

The provided label dump data was used to extract labeled objects from the videos and these extracted objects formed our initial training set. The initial training set contained the same objects multiple number of times without much change in size and orientation. Such a training data would have led to overfitting no matter what the classifier was. So to avoid such overfitting we first build the training set by extracting labeled objects every 30th frame and then manually removed whatever redundant data were present for the same objects.
We have experimented with the previously discussed features with variations in their parameters and different classifiers. The results are shown below. All the results are for **5-fold cross validation**.

Experiments with various parameters of HOG and different classifiers

1. Cells per block = (3,3) and Pixels per cell = (8,8)

Classifier	Accuracy	Precision	Recall
Linear SVM	0.852951	0.829102	0.816334
Random Forest (n_estimator = 100)	0.815646	0.862853	0.862853
Random Forest (n_estimator = 200)	0.823252	0.866324	0.866324
Random Forest (n_estimator = 250)	0.821079	0.860848	0.860848
Adaboost (n_estimator = 100)	0.674393	0.689259	0.689259

2. Cells per block = (2,2) and Pixels per cell = (8,8)

Classifier	Accuracy	Precision	Recall
Linear SVM	0.761412	0.684711	0.593173
Random Forest (n_estimator = 200)	0.805340	0.761680	0.739426

3. Cells per block = (1,1) and Pixels per cell = (8,8)

Classifier	Accuracy	Precision	Recall
Linear SVM	0.770887	0.694461	0.689056

4. Cells per block = (3,3) and Pixels per cell = (6,6)

Classifier	Accuracy	Precision	Recall
Linear SVM	0.808354	0.757420	0.722123
Random Forest (n_estimator = 200)	0.742032	0.675545	0.557587

Experiments with different number of cluster while using SIFT features for classification and different classifiers

No. of clusters	Classifiers			
	LinearSVM	Random Forest n_estimator = 100	Random Forest n_estimator = 200	AdaBoost Classifier n_estimator = 100
100	0.625053	0.635853	0.645788	0.573218
200	0.619438	0.655723	0.655723	0.554643
300	0.598272	0.647084	0.651835	0.556371
400	0.621598	0.661771	0.662203	0.557235
500	0.609071	0.656587	0.665226	0.525701
600	0.597840	0.661771	0.659611	0.546004
700	0.623758	0.667818	0.669114	0.531317
800	0.628509	0.664794	0.669114	0.536501

900	0.637149	0.666090	0.679049	0.543844
1000	0.611231	0.665226	0.672138	0.537796

Conclusion: After comparing the performance of SIFT and HOG we settled on HOG with cells per block = 3,3 and pixels per cell = 8,8 and LinearSVM as the classifier.

3. Foreground-Background separation

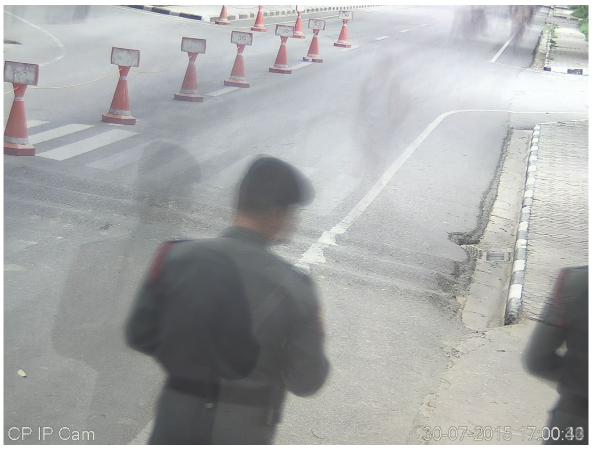
Foreground objects present in a frame can be thought as regions in the frame which are different from the corresponding regions of the background frame

We first used the first frame of the video as the background frame. This will give wrong detected objects for cases when an object is already present in the first frame.

Then we used the accumulated weighted frame of the whole video as the background frame. The accumulated weighted frame is calculated using the following relation:

$$c = (1 - \alpha)c + \alpha f$$

Where c is the accumulated weighted frame , f is the current frame and α regulates the update speed(how fast the accumulator forgets the earlier frames). In other words, keeping α low will ensure that the accumulated frame does not change drastically due to the presence of a new object in a few frame. In videos where an object is present in most of the frames, the accumulated weighted frame will have such objects in them. So we ruled out this idea.



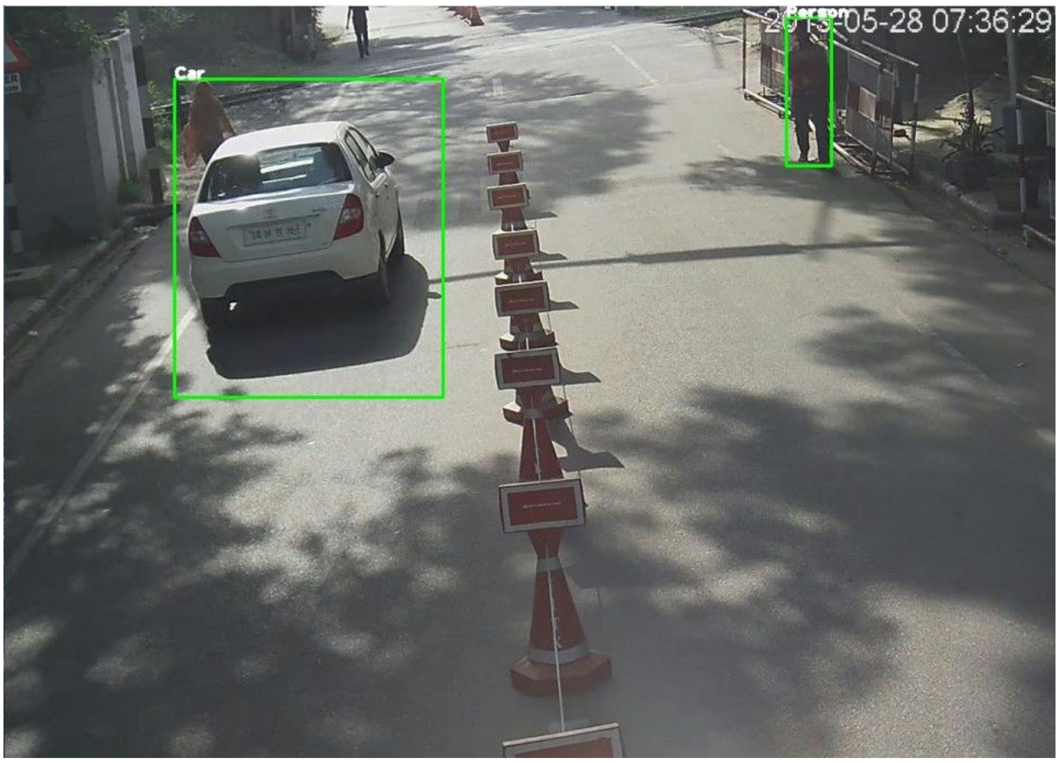
Even with $\alpha = 0.0001$ the accumulated weighted frame turned out to be the above frame due to the presence of the guard at almost the same position throughout the video.

Finally we decided to use the **BackgroundSubtractorMOG2** function of OpenCV for foreground-background separation.It is a Gaussian Mixture-based Background-Foreground Segmentation Algorithm. One important feature of this algorithm is that it selects the appropriate number of gaussian distribution for each pixel. It provides better adaptability to varying scenes due illumination changes etc. In addition to this, we used the following image-processing on each frame for better object detection:

- **GaussianBlurring** as it is highly effective in removing gaussian noise
- **Morphology Transformations** to separate very close foreground objects
- **Threshold on the size of objects** to further remove noise

4. Classification of the detected objects

The objects detected after the foreground-background separation are resized into 100 X 100 images and HOG features are calculated for them. Then they are classified using the LinearSVM classifier and given labels accordingly.



Still of our LinearSVM classifier at work

We also tried an alternate approach which uses sliding-window technique and HOG to detect and classify the objects simultaneously.

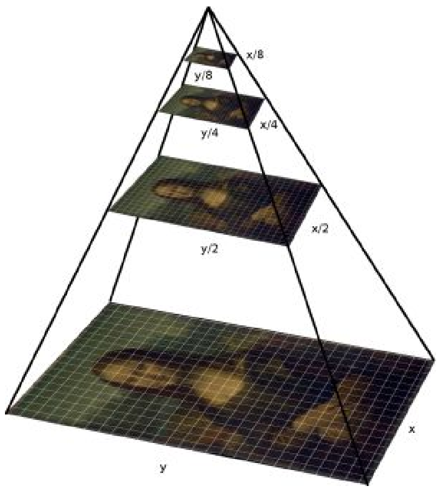
Sliding-window and HOG

In this, only the foreground-background separation and classification part differs from the previous method.

Basic Algorithm: We slide a window of size 100 X 100 across the frame of a video. HOG features are calculated for each window and each window gets a label according to the LinearSVM classifier.

There are three major problems with the basic algorithm.

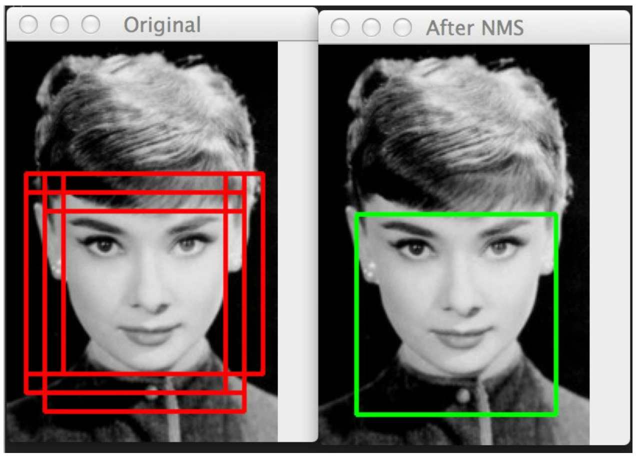
Problem 1: An object may a size greater than the window size.
Solution: Image Pyramids.



An example of an image pyramid. At each layer of the pyramid the image is downsized and (optionally) smoothed.

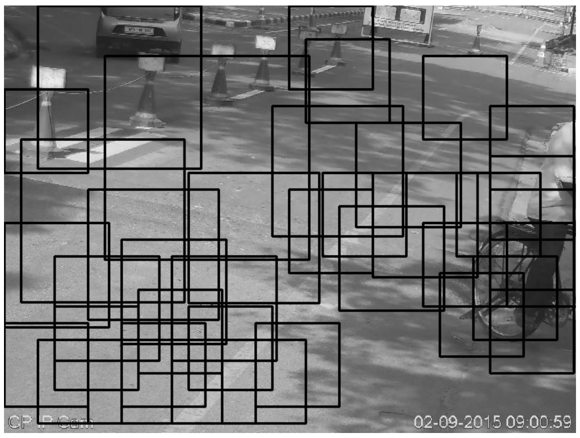
An “image pyramid” is a multi-scale representation of an image.
Utilizing an image pyramid allows us to find objects in images at different scales of an image.
And when combined with a sliding window we can find objects in images in various locations.
At the bottom of the pyramid we have the original image at its original size (in terms of width and height). And at each subsequent layer, the image is resized (subsampled) and optionally smoothed (usually via Gaussian blurring).
The image is progressively subsampled until some stopping criterion is met, which is normally a minimum size has been reached and no further subsampling needs to take place.
The sliding window is run across all the images present in the image pyramid of a frame.

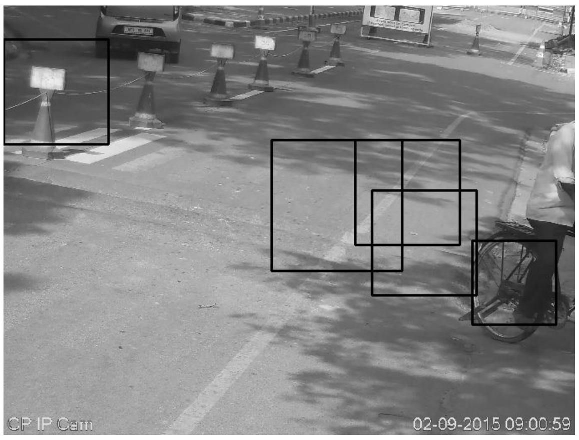
Problem 2: The same object may get classified in many overlapping windows.
Solution: Non-maximum suppression
Non-maximum suppression (NMS) is a key post-processing step in many computer vision applications. In the context of object detection, it is used to transform a smooth response map that triggers many imprecise object window hypotheses in, ideally, a single bounding-box for each detected object. It essentially merges superfluous boxes to get a single box which is a better outline.



(Left) Detecting multiple overlapping bounding boxes around the face we want to detect. *(Right)* Applying non-maximum suppression to remove the redundant bounding boxes.

Problem 3: A window may get a label even though it does not contain any objects.
Solution: Hard-negative mining.
For each image and each possible scale of each image in your training set, apply the sliding window technique and slide your window across the image. At each window compute your HOG descriptors and apply your classifier. If your classifier (incorrectly) classifies a given window as an object (and it will, there will absolutely be false-positives), save the window in the training set with label as 'background'. This approach is called hard-negative mining.





(Left) Before Hard-negative mining, many background windows get classified as Bicycle. *(Right)* After applying Hard-negative mining, misclassified background windows are reduced drastically.

Future improvements

- Sliding window can also be used to process the output of OpenCV to get a better focus on the objects in the rectangle. Currently the rectangles drawn are very big so there is a lot of extra background being input into the classifier. With sliding window we can eventually bring the object under the window and filter the background to get better results.
- We are not stabilising the predicted labels of objects. We pipeline every frame into the classifier which gives us varying labels. We can improve this by taking into account the history of labels and showing the most confident prediction throughout the video.

References

- https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients
- https://en.wikipedia.org/wiki/Scale-invariant_feature_transform
- http://docs.opencv.org/3.1.0/
- http://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/
- http://www.pyimagesearch.com/2015/11/09/pedestrian-detection-opencv/
- http://www.pyimagesearch.com/2015/03/16/image-pyramids-with-python-and-opencv/
- http://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/
- https://github.com/bikz05/object-detector
- https://lirias.kuleuven.be/bitstream/123456789/506283/1/3924_postprint.pdf