Final Project Report : Surveillance (Face Recognition)

CSCI 5722 Computer Vision, Fall 2016

Vibhor Mishra & Shubham Mudgal

Initial Project Objective:

The project was aimed to create a robust face recognition technology which would efficiently and effectively detect faces and recognize people in the images and then extend it to videos. The initial plan was to make a mobile application using available technology and implement the system. The application would take an image from either the image file already present in the mobile phone or click a new image using the phone camera and automatically detect and recognise the faces present in the image. It would detect multiple faces(if multiple faces were present) in the image and on the fly display the recognized person's name.

Initial Project Goal:

Since there are a lot of complications in this specific area of object recognition(i.e. Face recognition), the accuracy was aimed to be at around more than 85%. Although, the performance of the system for face detection part was assumed to be around 90% in the images and 90% in the videos (given the images and videos are of sufficiently good quality meaning that the faces present in the data are clearly visible).

Algorithm:

There were five main sections in which the project was divided.

1) Collect multiple images of a set of people to form training dataset which will be labeled and used for classification.
2) Detect the faces in those images
3) Extract features
4) Train classifier using the feature set
5) Test the system using test data

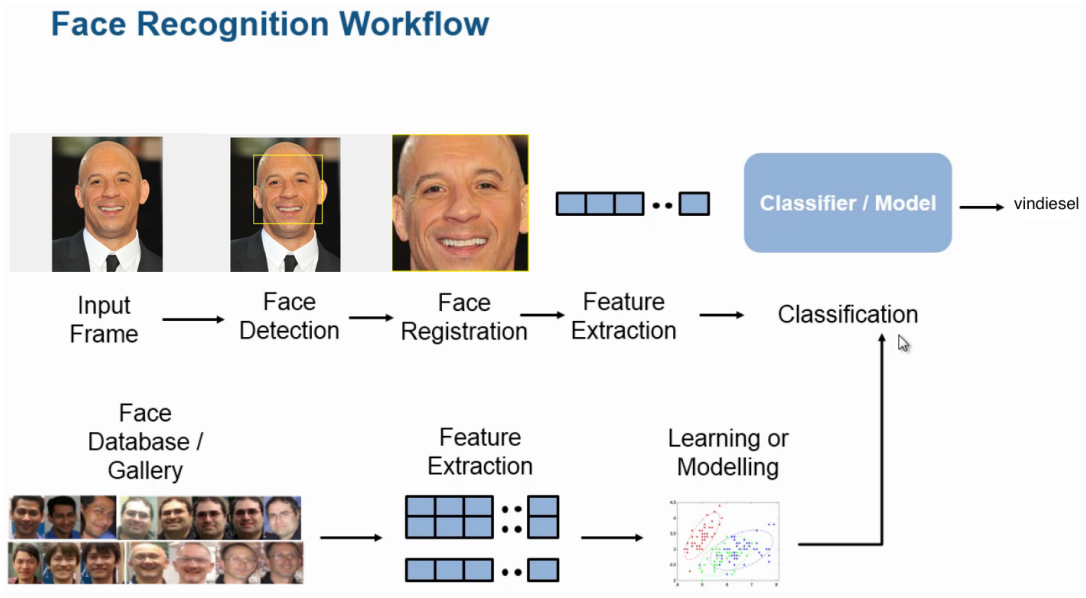


Figure 1: Face Recognition Workflow (source: Mathworks)

1) Data Collection:

One of the most time consuming aspect of the project was to collect data of a specific people to get all the different features of their facial qualities. Initially images were collected from google search of known celebrities which were readily available. This imageset, however, was not exhaustive enough to fully train the classifier. The rudimentary operations performed were done by collection 25 images of 5 celebrities (having variations in those images in terms of hues, saturation and illumination). Later on in the project, a program was built to extract images of the person from a video where facial images would be cropped and stored at a location at the rate of 10 frames per second. This approach saved a lot of time, even though a lot of filtering had to be done to the extracted image before they were fit to be used for training purpose.

2) Face Detection:

The next step in the process was to detect faces in the images collected in step 1. Several tutorials provided by Mathworks were used to understand how the face detection step actually worked in Matlab. The images for training data may contain people with a lot noise(i.e. their whole body or other objects near them). This noise or unwanted data was not required and had to be trimmed out of the images before extracting necessary features. The inbuilt function provided in the 'Computer Vision toolbar' called '*CascadeObjectDetector'* was used for this purpose. This method would detect any facial quality like eyes, ears, nose or the whole face (as required) present in an image. A box was applied to the part of the image to enclose or show the face detected. There were a few challenges here as well, where there were sometimes false positives or false negatives. These issues had to be dealt with before proceeding to further steps in the

process as it was an important decision factor responsible for the effectiveness of the system in the early stage. Threshold had to be applied accordingly if too many or too few faces were detected in an image. Initial threshold was set to a default value for each image and then it was increased or decreased according to the response of the function in detecting faces.

Face detection step comes in later stages as well, when the test data is fed to the system. The image provided may have lots of faces along with lots of noise. The system was responsible to detect only the facial part of every person and classify it to the respective category. Although, here the facial part need not be saved anywhere since only the features extracted from the facial region were required for the system to predict.

3) Feature Extraction:

One of the most important questions in this project (or any project related to object recognition) was which feature extraction methodology is to be used which would solve this problem in the most efficient way. The initial approach done by using the *'bagOfWords'* algorithm. Even though the features extracted solved the problem of object recognition and rightfully recognized the objects like car, table, etc.(tested in a prototype), it would not yield good enough results when applied to face recognition. One of the reasons for poor performance here could be lack of enough training data or use of a poor classifier. When tested on 25 images of 5 different people, the system (using *bagOfWords* as feature extraction approach) proved to have around 64% accuracy. Hence, correctly classifying only 16 faces out of 25. This was the result when 20 faces were provided for each 5 people. Hence the training set consisted of 80 images in total.

A paper called *'Histograms of Oriented Gradients for Human Detection'* by Navneet Dalal and Bill Triggs, showed that the efficiency of HOG based feature extractor combined by SVM classifier improved the results for person recognition (and face recognition). After reading and understanding the paper thoroughly, HOG was used to the method to extract features from a face. Histogram of oriented gradients essentially stores the local object appearance and shape within an image as distribution of intensity gradients or edge directions. The use of this feature extractor algorithm when combined with SVM gave a result of 92% accuracy when tested with the same 25 static images as above.

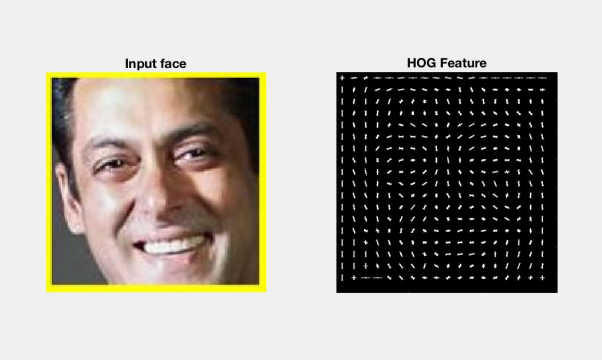


Figure 2: Extracted HOG features of an input image of a face

Of course this level of performance cannot be generalised since both the training data and testing data were limited and also the number of people classified were only 5. However, when tested this approach with face recognition in videos and live streaming (from webcam), the system turned out to be efficient 88% of the time.

4) Training the Classifier:

The next big step after feature extraction was train the classifier based on the features gathered. Again there were a lot of classifiers out there to choose from, but Error-Correcting Output Codes(ECOC) proved to be well suited for this problem of multi-class classification. This classifier trains the system taking in the features provided and it is also found out that efficiency of classifying correctly is enhanced based on the amount of training data provided. There are two stages to this technique: training the classifier and predicting the test data. Prediction will be discussed in the next step. The classifier creates clusters of feature points.

5) Testing the data (Prediction):

The last step of the process is to test it. Input images or test images are provided to the classifier for it to predict the correct class(or person to which the given face belongs to). This step actually involves a prerequisite of most of the above steps. First we collect the test data(which is to be recognised). Then face is detected from the image provided in the test data. Next, features are extracted from the face detected using the same feature extraction algorithm used in step 3(in this case HOG). These features are then provided to the classifier to predict the class of the test image.

The prediction step is a little tricky since the classifier gives in the probability of which class the given image could belong to. The class with the highest probability is chosen to be the rightful class of the input image.

Recognition of Multiple Faces in one frame:

Our initial goal was to do face recognition on an image provided only one person existed in the test image. But we further enhanced the functionality and scope of the project to detecting and recognising multiple faces at once. This was a fairly easy approach as the input test data was a static image. However, upon further expansion of the scope, we moved to detecting and recognising multiple images in a video feed. This was particularly a fair amount of work as this required to take multiple frames of a video, detect faces in it, recognise those faces and then

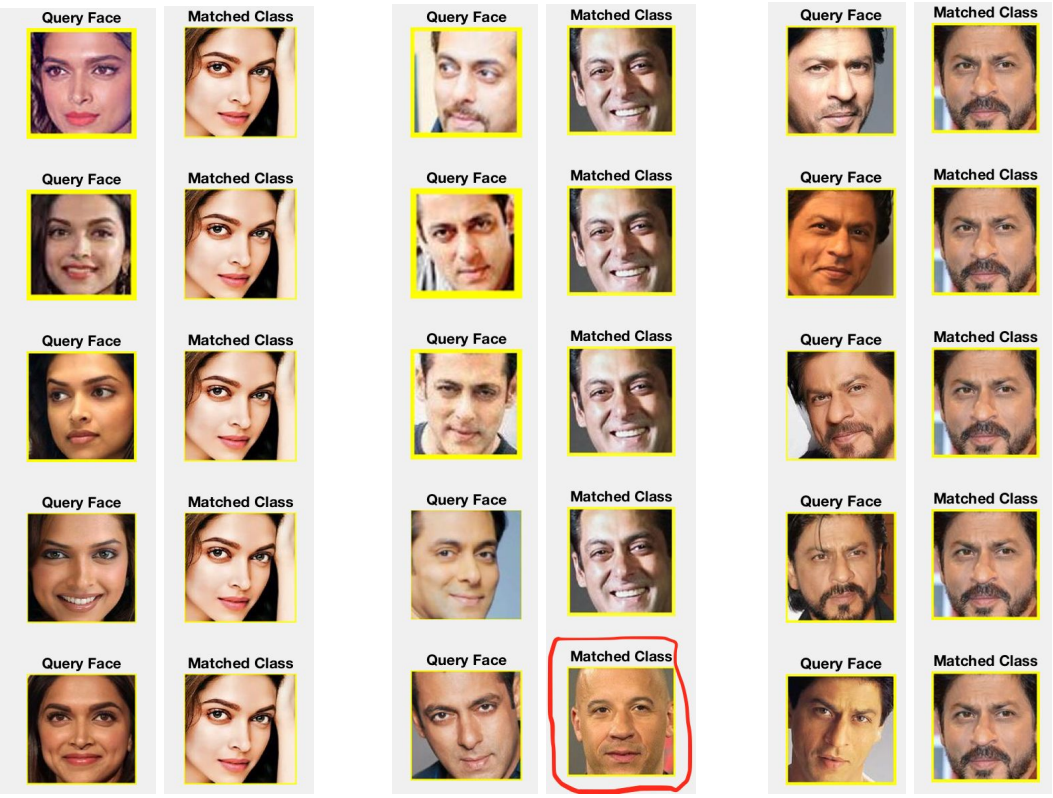
return result, and all that in real time while the video is playing. Now the real challenge was to make the system faster and correct at the same time.
A class was used to implement this which would take in a frame from the video, and store the points in the image where faces are detected. This face is then cropped and the features are extracted from it. These features are used to predict the class of this face. Now, since it would take a lot of time to do this for every frame, we implemented a tracking system which would track multiple detected faces by implementing KLT algorithm in the following video. *'PointTracker'* object provided in the Computer Vision toolbox was used to track the corner points gathered for a detected face in the frame. Here are the following steps used to track and classify faces in a video:

- For each face, corner points are found using *'detectMinEigenFeatures*' which are used to uniquely identify faces. These points and the bounding box IDs are stored in the Object tracker class object to keep track of each face or box in the video frames.
- To check if the detected faces in the new frame are new or from the last frame, the area of overlapping box region of already existing and new boxes (faces) is compared to a threshold value.
- If the threshold is not satisfied:
    - it's a new face. Compute corner points and store new box information in the class
    - Extract features and run classification for this new face and store the classified label.
- If the threshold is satisfied:
    - It's an already existing one. Update the points and box information.
    - Don't perform classification for that particular box enclosing face and use the previous prediction.

In many cases, the face movement was very abrupt and the boxes were not matched and deleted from the object. A new box was then assigned to the newly detected old face and process was continued. This approach was used to detect and keep track of multiple faces detected in a video feed.
One problem with this approach was that once a face is detected and recognised to be of class A, it would then track that face unless the person moves out of the video. If the initial prediction was however not correct, it would propagate the wrong prediction for as long as the person is in the video. Therefore to prevent this flaw, detection and recognition on the current boxes(faces enclosed in boxes) was done every 10 frames. This way the faces detected initially would not propagate to the whole video feed.

Results:

As seen below, for the dataset with 25 images of each class, we get the accuracy of 92% with HOG feature extractor and using SVM classifier.



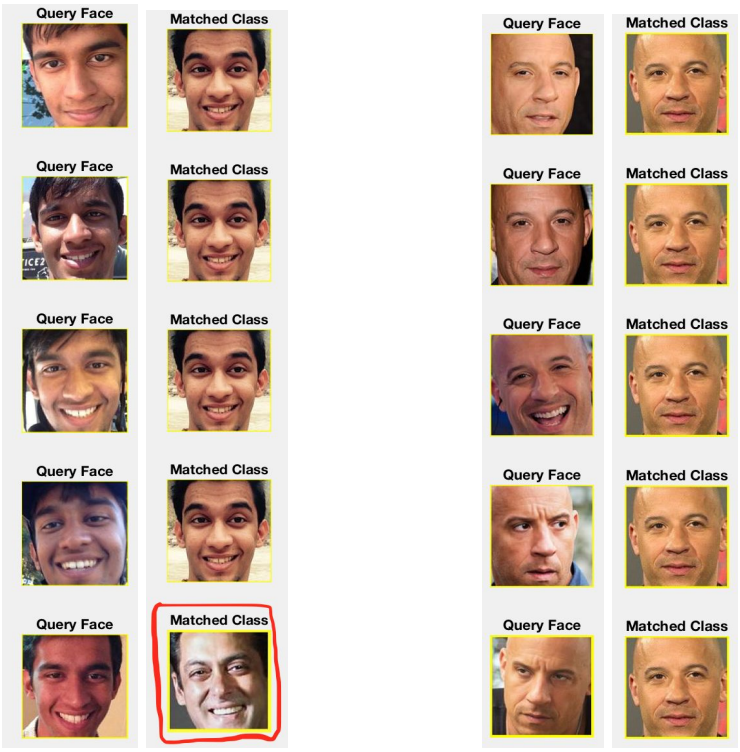


Figure 3: Results of testing 25 images against the SVM classifier trained on 80 images using HOG features (92% accuracy)

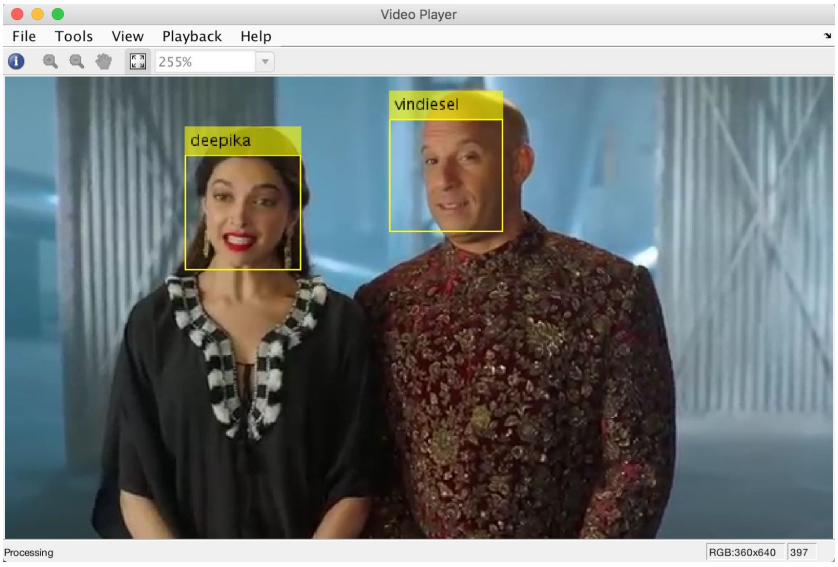


Figure 4: Face recognition showing accurate results when played on a video (for most part)

On increasing the dataset to huge number for each classes, the accuracy we get in videos is around 88%. With videos, the accuracy is computed by calculating the [time for which the classifier predicted labeled trained class correctly] dividing by [total video time] *100:

Accuracy (%) = (time for which the classifier predicted labeled trained class correctly)/ .
(total video time) * 100

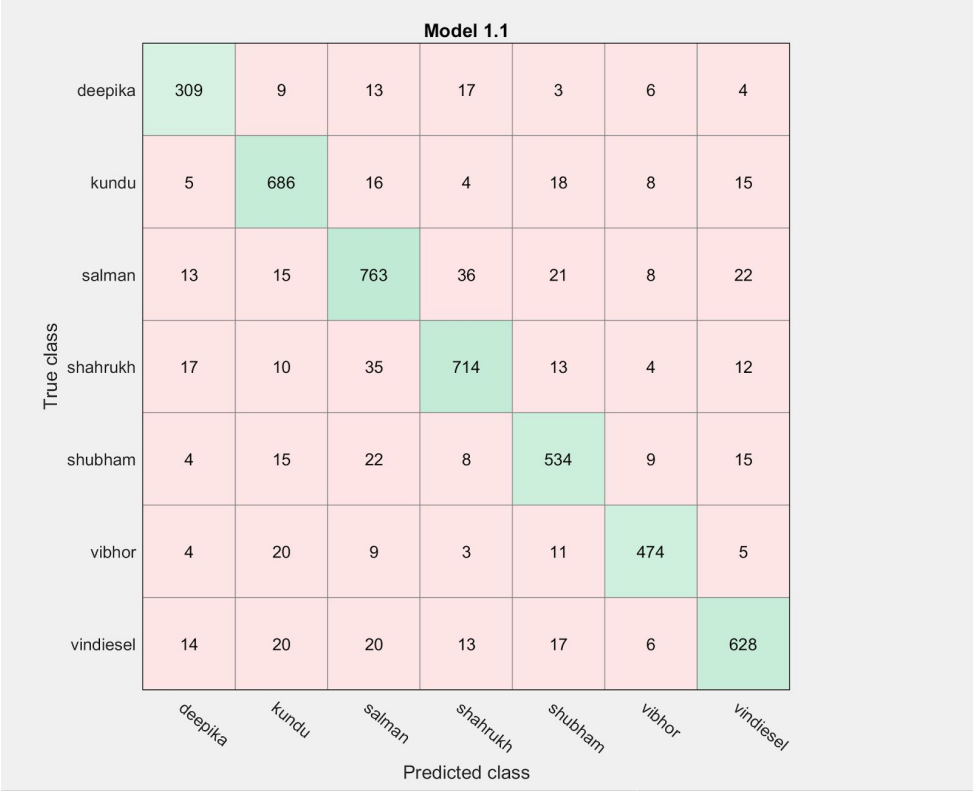


Figure 5: Confusion Matrix

Computer Vision Methods Used:

1) For the Face Detection part, a method called '*CascadeObjectDetector'* provided in Computer Vision toolbox was used. A few tweaks like setting up the threshold dynamically for different images was done for dealing with false detection of faces and true detection of non-faces in the given image.
2) Kanade-Lucas-Tomasi(KLT) algorithm used to track multiple objects in a video feed.
3) videoPlayer, a Computer Vision method used to show video feed in a player.
4) snapshot, a Vision camera method used to acquire the current frame as a single image.
5) PointTracker, a method used to track points of the detected face.
6) insertObjectAnnotation, used to annotate truecolor or grayscale image or video stream.
7) extractHOGFeatures, method used to get histogram of oriented gradient features of a given input image.
8) Fitcecoc, Error-correcting output codes classifier used to cluster the images into multi-classes based on the features, using SVM as default classifier.

Limitations:

1) Abruptly changing scenes: This system will not perform very diligently on a video that contains abruptly changing objects, locations and scenes because the objects are tracked until they move out of the frame. In abruptly changing scenes, there would have to be more detection and less tracking and hence the system is more suitable for surveillance scenario.
2) Training the classifier for a new face: Another limitation of the current system is that suppose the classifier is trained for a thousand image, and if a new face(person) or class is to be added to the classifier, then the whole classifier has to be trained again with the features of all 1001 people. This approach is not scalable for large scale system as it would take a lot of time to again train the classifier for every new person coming in the system.

Future Extensions:

On the fly training and classification of unknown people who aren't currently present in the database. The above project currently detects only the known faces present in the classifier. When an unknown face is detected in a video, the system shows it simply as an 'unknown' tag. In future extension- we would give the user an option to enter a name beside the unknown face and the get images from the video itself of that 'unknown' person. While the system would detect all the known faces correctly, at the same time it would get several images of the unknown face, extract features and again train the classifier with the newly found data. It will then correctly detect and recognise the (once unknown) face in the video feed.

Contributions to the Project:

Vibhor Mishra:

- Research about the methods used to extract features for face recognition
- Research about how to track objects in a video feed
- Implementing multiple face detection in images and videos
- Analysing which classifier best suits for face recognition
- Making the project interactive
- Documenting the project

Shubham Mudgal:

- Running prototype of a simple program for object recognition to get started with the project.
- Collecting data for training and testing purposes

- Implementing and weighing various feature extraction methods and training the classifier
- Implementing object tracking
- Documenting the project

Things learnt during the project:

- Various methods used for feature extraction and classification. Their merits and demerits based on the requirement of the project.
- Object tracking in a video feed.
- Explored Computer Vision Matlab toolbox and classification learner app.

Advice for Future Computer Vision Students:

- An interesting and fairly new (and developing) field of research. Will have a lot learn and explore with.
- Do not take other heavy courses along with this course as this course (being very interesting) would (or could) end up taking a lot more time than expected.

References:

- Face Recognition Tutorial by Mathworks.
- Object Recognition using Machine Learning and Deep Learning Tutorial by Mathworks.
- Paper on '*Histograms of Oriented Gradients for Human Detection'* by Navneet Dalal and Bill Triggs.
- Code reference:
- https://www.mathworks.com/matlabcentral/fileexchange/47105-detect-and-track-multiple-faces