

Face Recognition, Emotion Recognition and Augmented Reality Implementation in MATLAB

by Mithu James, City, University of London

FACE RECOGNITION

Face recognition system is the application of computer vision to identify a person by the facial properties. This may be done by extracting facial features, storing them in a database and comparing a future image against the stored features to identify the individual. Though this is usually applied in security systems, administrative portals etc. this concept is adopted in the coursework to develop a scaled down version of face recognition system.

1) Requirement

The objective was to develop a MATLAB function that identified individuals within a single or group photo by their labels along with center co-ordinates of faces. The function was to take image, computer vision feature type and machine learning classifier of choice as inputs. It was expected to process them and return individual identification parameters as an array. The function was also required to have the capability to process atleast 2 types of features and 3 types of classifiers. The function was to take the form:

$P = \text{RecogniseFace}(I, \text{featureType}, \text{classifierName})$

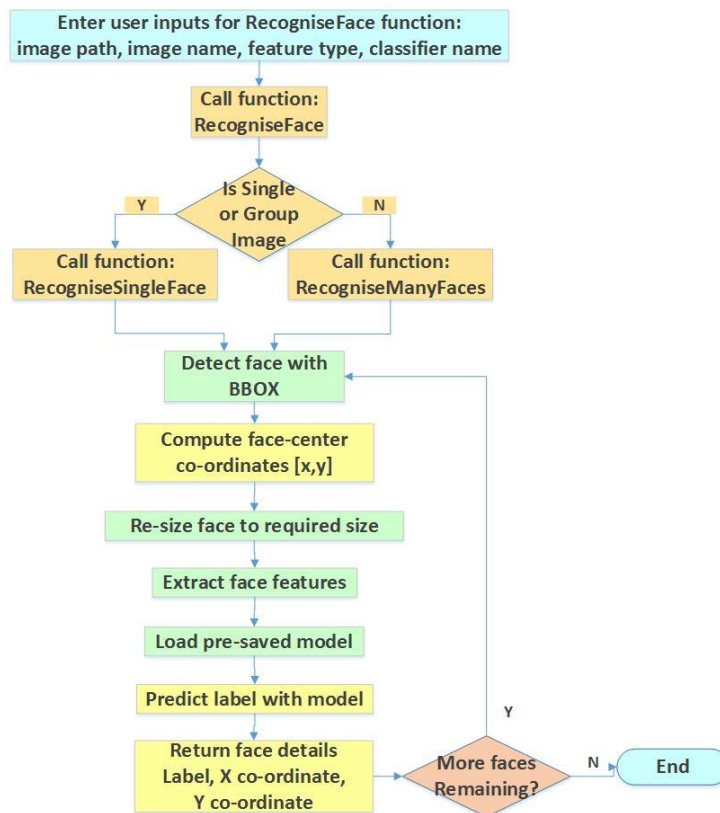


Fig1: Implementation of RecogniseFace function

2) Deliverables

The above requirements were fully implemented and a brief summary is provided below.

- a) Function name: RecogniseFace
- b) Input arguments: Image, featureType, classifierName (type: string)
- c) Output: faceDetails (type: array)
- d) Supported image types: Single, group photos
- e) Supported feature types: Histogram of oriented gradients (HOG), Scaled Up Robust Features (SURF) and Local Binary Patterns (LBP)
- f) Supported classifiers: Support vector machine (SVM), Random forest (RF), Feedforward neural network (FNN) (Naïve Bayes wasn't done due to problems with low class variance).
- g) Best performing model: HOG feature extraction followed by SVM classification named as: HogSvm_i300x300_c4x4 indicating image size of 300x300 and cell size of 4x4.

3) Implementation

A brief overview of the approach to face detection, recognition and emotion recognition is depicted below.

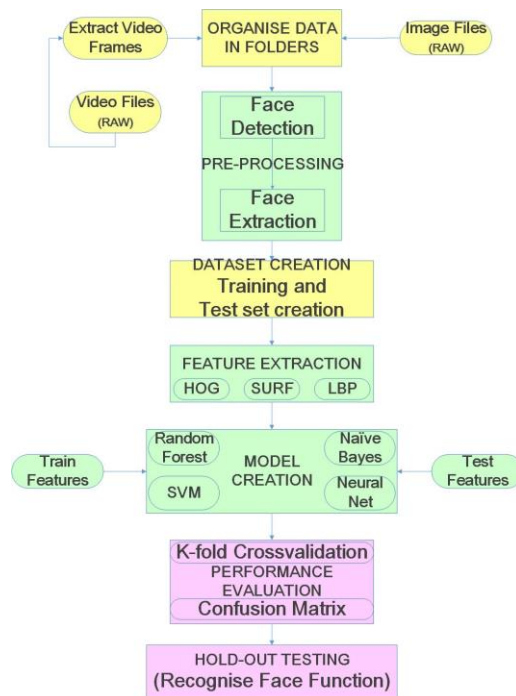


Fig 2: Implementation approach

Detailed enumeration of above implementation is given below.

a) Sorting and labeling of resources

- i. All the images and video files of every individual were organized and grouped into folder by the number on the placard. This number was used as the Label for every individual.



Fig 3: Dataset structure

- ii. Though automatic grouping and labelling of the files were attempted with **OCR recognition**, it worked only with a handful of images. Therefore, manual sorting and labelling was adopted. OCR recognition was attempted in many ways. One of them was blob analysis and the other was regionprops in getting the bounding box around the text. Though filtering by a threshold area size or confidence levels were tried, it worked only for a handful of cases and was not reliable. A sample label read by OCR is embedded. **14**
 - iii. **Output of this stage:** Original image and video files sorted into folders by their labels
- b) **Creation of image collection** – Resources sorted and labeled above were first processed to create an image collection.
- i. **Video files** within each labelled folder were processed by extracting the frames within them. For identification purpose, the frames were saved by concatenating video file name and frame count. For e.g. frame 23 from video IMG_3098 was named and saved as IMG_3098[023].
 - ii. **Image files** didn't require any further processing or labelling.
 - iii. **Output of this stage:** Images and video frames sorted into folders by their labels
- c) **Face extraction**– As face detection is the first step to face recognition, faces were extracted from above folders and saved to a new similar folder structure.
- i. **Face detection:** Viola-Jones face detection algorithm was used to segment face like area in every video and photo image. It used *Haar-like* features to search images for faces. (Note: In this dataset flawed detection was observed with items like switch board being identified as face due to its Haar-like features. Few people faces were also not identified; it is imagined that those faces had weak Haar-like features leading to early filtering in the cascading process.)
 - ii. **Face cropping:** Upon detecting faces in the images, they were cropped and saved to a new folder structure. This helped in removing those non-faces that were still detected as faces by the algorithm. Face cropping drastically reduced training time as the much smaller image could be more easily processed.
 - iii. **Extended bounding box:** Additionally, a more tailored approach to face detection was done by extending the bounding box by a small percentage of the detected area. This was to benefit from additional features that were specific to “this dataset” as peoples’ features were strikingly different by hair colour, hair length, eyecolour and skin tone. The additional area to be extracted was decided through trial and error to extract just what was needed. For e.g. $yTop = BBOX(N,2) - 200$ could be used for individual images, but it had to be modified to $yTop = 0.2 * BBOX(N,2)$ to extract faces from

group photos due to differently sized faces. For the specific problem in hand, this method seemed to be initially providing better results than original bounding box. Though the results were very good on individual images, it had slightly lower performance that group photo and hence, this approach was side-tracked.

iv. **Output of this stage:** Cropped faces organized into folders by their respective labels.

d) **Dataset creation** – The newly created ‘face’ dataset was further refined to choose a reduced set of images for training and testing purposes.

i. **Selection of video frames:** A subset of video frames that best contributed to feature extraction was chosen by selecting every ‘Nth’ video frame. Programmatically, this was done by selecting video frame that was a multiple of a constant (say 7 in most cases). This selection benefitted in four ways: (i) avoiding very similar adjacent video frames to learn features from slightly different frames (ii) improving training time with reduced data (iii) smoothening class imbalances by adjusting the ‘N’ value and (iv) enhancing flexibility to augment the dataset as and when needed.

ii. **Formation of training and test sets:** Training and test sets were formed by shuffling and randomly selecting the images.

a. Initially images from video files were mainly used for model creation. The training and test sets were created with a 95-5 split of video frames. The intention was to train on video frame and validate generalization on a separate hold-out set i.e. original image files that were captured differently. However, this idea had to be compromised due to class imbalances that arose from fewer video files for certain labels.

b. Besides the *individual image files and video frames*, training data was also harnessed from the *group image and video files*. This approach was adopted when the model predictions were not good on the group photos. Though it dropped the test accuracy during model creation, it was chosen as the way to go forward due to better generalization.

iii. **Output of this stage:** Training and test sets for model creation.

Testing of main RecogniseFace function was done with (i) one single photo for every label (ii) a few variety group photos and (iii) non-person image (e.g.tree) as seen below.

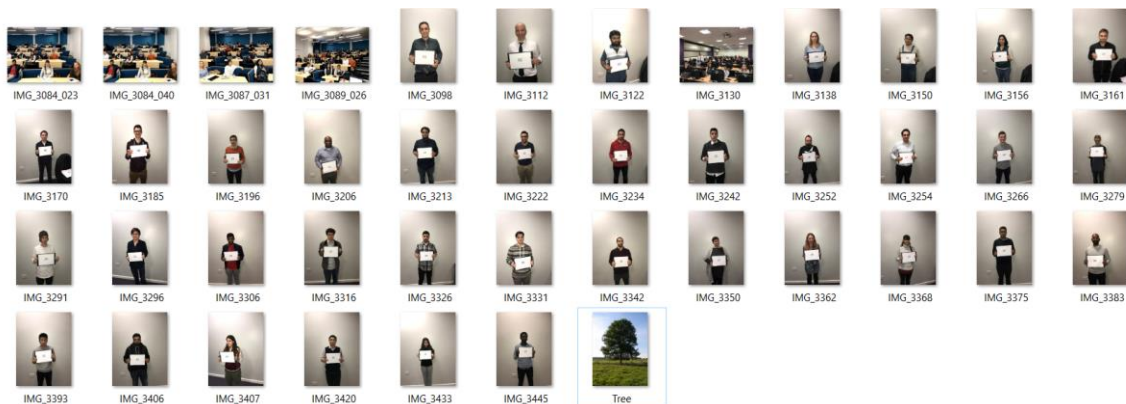


Fig 4: Hold-out test set

e) **Feature extraction** – is a kind of dimensionality reduction to represent interest points of an image as a compact feature vector. In order to select good features that were repeatable, distinctive and accurately identify 2D position, three types of feature extraction

were adopted. LBP was chosen alongside popular methods HOG and SURF, as existing literature showed promising results.

i. **Histogram of Oriented Gradients (HOG): (Best performing model was HOG+SVM)**

- a. **Technique** – It counts the occurrences of gradient orientation in localized portions. of the entire image. The gradient histogram measures orientation and strengths of image gradient within an image region. Originally developed for people recognition, it can easily handle many dimensions of the order of a few thousands. It calculates the gradient on an image region by convolving with $[-1 \ 0 \ 1]$ filters and transposing it. Upon calculating gradient magnitude and direction, it computes histograms on a specified cell size and normalize them within overlapping blocks of cells; the histograms are eventually concatenated. The invariance to bias is provided by the histogram normalization. HOG was chosen due to its effectiveness in people detection and particularly due to proven research in handling partial occlusion (which is needed while testing on group images). As proved back in the days, it worked best with SVM.
- b. **Challenge** – in using the HOG feature was that the number of features extracted was dependent on the size of the image. As the resources for the task had been captured in different cameras in different forms (i.e. images and videos), the pre-processed images differed in resolution and size. Therefore, re-scaling of images to one size was essential. More sacrifice of features was required mainly to scale down single image faces to the size of faces in group photo. As some of the faces in group photo were as small as 30x30, they were also scaled up. The optimal resize and local cell size for gradient computation were decided through several rounds of training. Best performance was observed with size of [300 x 300].
- c. **Effectiveness** – HOG performed well with faces that were *oriented* in different directions or varied angles. Though resizing led to loss of information, its performance was appreciable due to *scale invariance*. It was also effective when faces were *occluded*. For e.g. in below image, faces are occluded by computer monitors but they are rightly identified (red circle, green star and red star indicate occlusion, correct classification and incorrect classification respectively).



Fig 5: Occluded face recognition

ii. **Bag of Features (BoF) or Scaled Up Robust Features (SURF):**

- a. **Technique** – Bag of features extracts SURF features from all images in the training set and reduces the number of features to a histogram of visual vocabulary by quantizing the feature space with K-Means clustering. Histogram formed the basis for training the classifier with encoded label item being used for actual image classification.

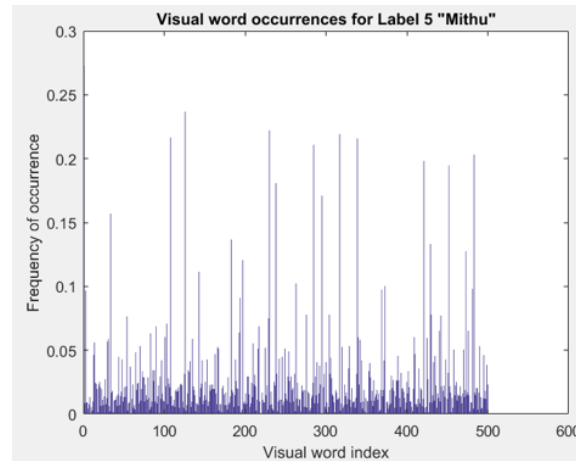


Fig 6: Visual word occurrences

- b. **Challenge** – It was more convenient to use BoF than HOG as resizing of images was not required. Training was also quicker as it was possible to restrict the number of features by selecting only the strongest features. But, optimal performance necessitated choosing the right number of bag of features. Model creation was done with 500.
- c. **Effectiveness** – Though BoF could be conveniently implemented in MATLAB with SVM, its accuracy was much lesser than HOG.
- iii. **Local Binary Patterns (LBP):**
 - i. **Technique** – Though not popular in face recognition as much as HOG and SURF due to its texture identification methodology, considerable research in LBP based face recognition encouraged to experiment with this feature. LBP describes local texture by comparing the intensity of each pixel with its neighborhood which can be of grid size 8 x8, 16 x 16 etc. Face can be considered to be made up of micro textures and LBP can be used to describe global and local facial texture by dividing the face into blocks and computing histogram for each of them. Global texture helps in differentiating non-face parts and local provides detailed information for face recognition. Results can be combined to a descriptor vector and fed to a classifier to decide face likeness of the input image.
 - ii. **Challenge** – Though quick to implement, the challenge was in deciding an appropriate fixed neighbourhood cell size. Drawbacks were that it failed to encode details differing in scale and gray scale conversion and resizing caused it to lose a great deal of details.
 - iii. **Effectiveness** – Results on single images were good as it was robust to pose and illumination changes. However performance on group image was not appreciable with fixed cell size. It is strongly felt in the backdrop of existing research that a mix of different cell sizes can improve model performance.
- f) **Model building** – is the process of using the extracted features to train a classifier that allowed to pass the actual image that needed to be classified. Three types of classifiers were used:
 - i. **Support Vector Machine (SVM):**
 - a. **Technique** – is a powerful supervised learning approach that avoids overfitting with its strong regularization properties. Its quality of generalization and easiness to train datasets with several attributes makes it a popular choice for image classification. The

margin maximization property is used by linear SVMs to fit hyper-planes between bounding classes. Non-linear SVMs use non-linear kernel to transform input data to high dimensional space for problem solving.

- b. **Challenge** – It was the most convenient to implement with MATLAB as extracted features could be directly fed in to train the classifier. As it seamlessly dealt with the high dimensions of images, training was fine. However, the major downside was the training time. It took more than an hour to train a model and more than 5 hours for 5-fold cross validation.
- c. **Effectiveness** – Models trained with SVM were the most reliable in terms of predictive power. Its accuracy was much better than other models and hence chosen for the final model.

ii. Random Forest (RF):

- a. **Technique** – is an ensemble of decision trees grown using bootstrap sampling. Random subset of features are used for node splitting.
- b. **Challenge** – Implementation was relatively straight forward but interpretation of the results were very challenging due to the large dimensions. Therefore hyperparameter tuning was not easy and well structured; it was mostly based on trial and error.
- c. **Effectiveness** – Quite contrary to expectation, predictions were not appreciable with RF even though OOBError was seen to be reducing. Though some of the hyperparameters like no. of trees, sampling were altered, results were still not good. As other classifiers gave better results, not much work was done with random forest.

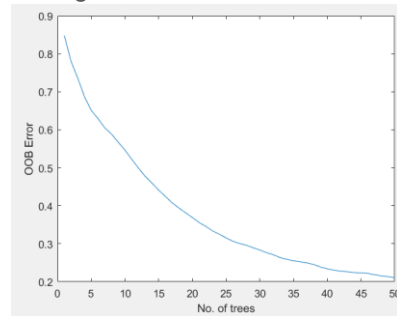


Fig 7: Random Forest OOB error

iii. Feed Forward Neural Network (FNN):

- a. **Technique** – It is a method of abstracting relationships between the input feature vector and output class label without any linearity assumptions. The power of neural networks lie in its ability to make non-linear approximations; it is done with activation functions that are non-linear. They can also model complex problems by increasing the depth of hidden layers for better transformations. For classification problems, it can use a softmax function at output to predict class probabilities.
- b. **Challenge** – Data feed was a bit convoluted as features had to be fed in as a vector array. Except for the 3 to 4 additional statements to pre-process features and labels, implementation was straight forward. However, the problem of vanishing gradient was observed during training. It was taken care through
- c. **Effectiveness** – Though a few varieties of feedforward neural nets in MATLAB were tried, patternnet which uses scaled conjugate gradient algorithm gave best results. It still used backpropagation for weight updates. Patternnet took lesser time than SVM for training and its performance was better on group images.

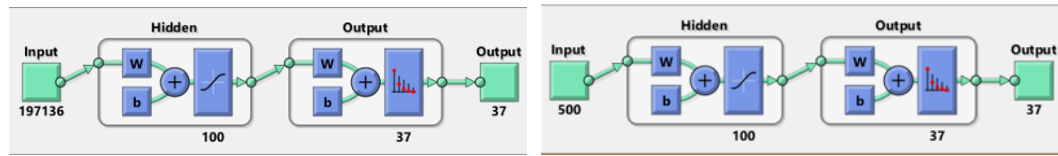


Fig 9: Patternnet (feedforward net) for HOG and SURF features respectively

Though training with Naïve Bayes was also attempted, it did not progress due to low or zero class variance. This needed for intense training to weed out unwanted items and was too onerous.

Hence, Naïve Bayes training was skipped.

```
PredictedHogNBTestLabels = predict(classifierHogNB, testHogFeatures);
[confMatHogNBTest orderHogNBTest] = confusionmat(testHogLabels, PredictedHogNBTestLabels);
disp('HOG-NB CLASSIFIER TRAINING AND EVALUATION ARE COMPLETE');
Error using ClassificationNaiveBayes/fitNonMNDists (line 232)
A normal distribution cannot be fit for the combination of class 1 and predictor x61965. The
data has zero variance.
```

```
Error in ClassificationNaiveBayes (line 104)
    this.DistributionParameters = fitNonMNDists(this);
```

Fig 10: Naïve Bayes not building model due to low class variance

g) Performance Evaluation – is the process of evaluating a classifier performance by passing a set of unseen images i.e. the test set. This is done by matching test images against a pre-trained database of images and assigning the label with highest matching score.

During model building, three types of testing are done one after another:

- i. **Testing of trained classifier with test set made up of faces** – This was done by running the 15% test set through the trained classifier to evaluate the performance. The test set with pre-saved faces was processed to extract the relevant feature (say, HOG, LBP or BOF) before passing it through the trained classifier.
- ii. **Testing of trained classifier with full individual or group images** (i.e. with RecogniseFace function) – This was done by passing whole image (not just the face) of an individual or group through the classifier to predict the class label. The test image was then processed to detect and extract the face, extract relevant features and pass it to classifier for label prediction. The output included class label and 2D coordinates of the center of face.
- iii. **Testing of trained classifier with group photos** – As all model evaluations worked only with pre-labeled individual images but the model was also required to predict group faces reliably, manual intervention was done to test the classifier on group photo.





The baselined model was evaluated in two ways:

- iv. **Confusion matrix** – True positives, True negatives, Accuracy and Specificity were determined for model comparison
- v. **K-fold Cross validation** – was used to determine training error by averaging the error from each fold. As it estimates error by averaging over k different partitions it reduces variance and gives better indication of model generalization.

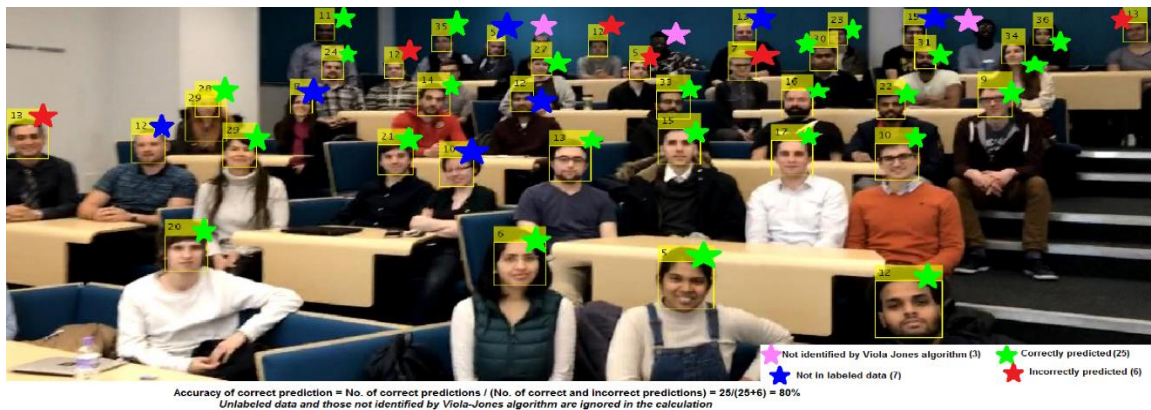
h) Packaging – is the process of collating and combining all scripts, functions and models to form one source code package. Training and test data were excluded as the relevant features from training data were already part of the model and therefore did not need it for future predictions. A single interface was also provided to use the package through the wrapper RecogniseFace function. As mentioned in the beginning page, it can take single or group image for class label predictions.

4) Analysis of final models

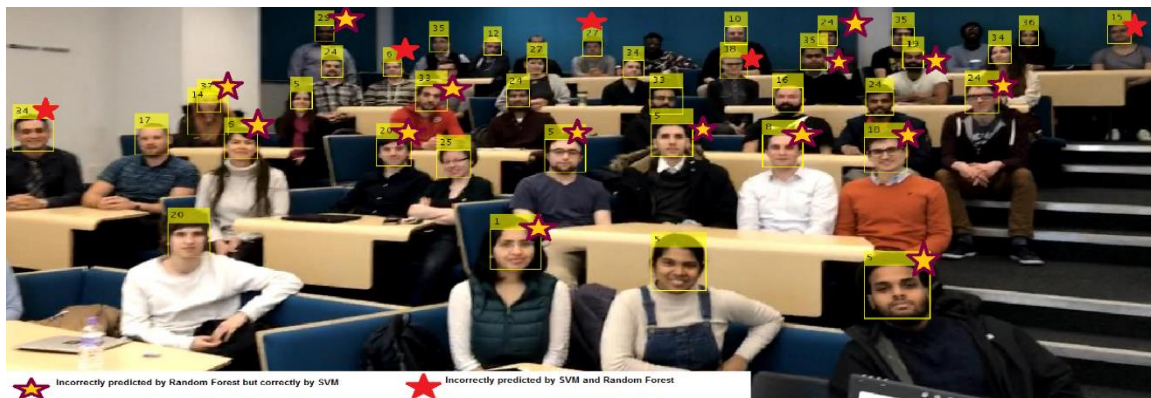
Amongst all features extracted and classifiers used for model creation, HOG features, SVM classifier and neural network classifier provided the best performance. As HOG-SVM was marginally better than HOG-NN, HOG-SVM is chosen as the final model. Anyhow, the two models are compared below.

Legend:  Not identified by Viola Jones algorithm (3)  Correctly predicted (25)  Not in labeled data (7)  Incorrectly predicted (6)

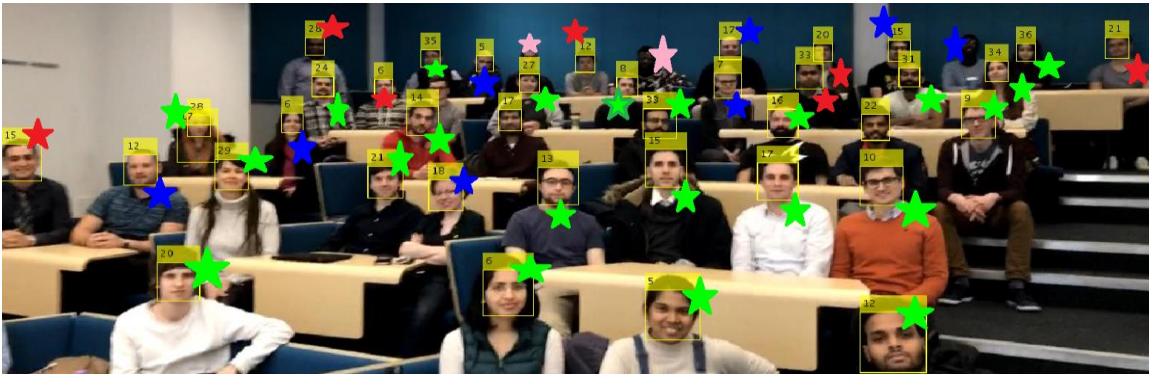
Accuracy of prediction i.e. correct prediction / (correct + incorrect predictions) for SVM in group photo is 80% as seen below.



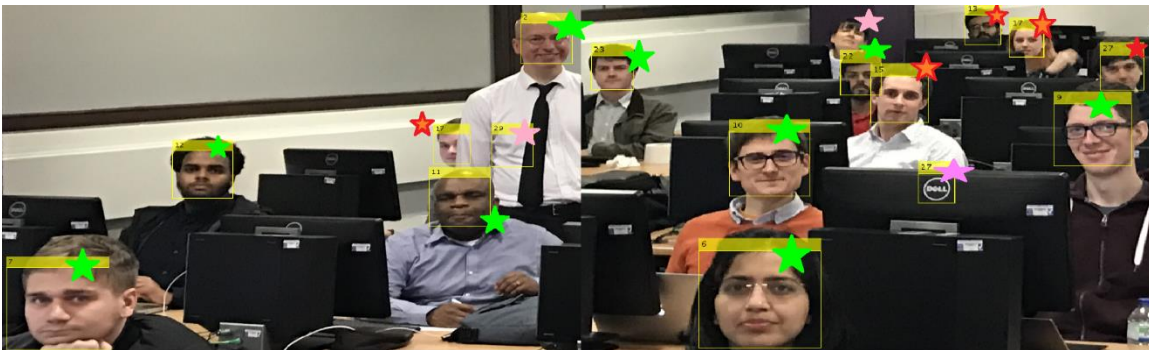
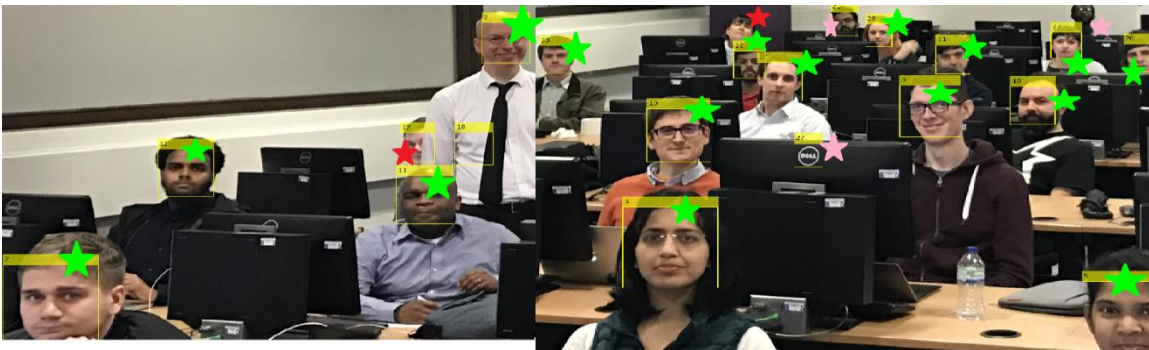
The same image was predicted by random forest with much less accuracy (highlighted stars below are incorrect predictions).



But, the prediction was better with feedforward neural network with an accuracy of 76%. One of the faces incorrectly predicted by SVM was correctly predicted by neural net.



However, in a more challenging picture with occlusions, neural network surprisingly gave better results with accuracy of 83% while SVM only had 77% accuracy. This may be explained by the lower cross validation error i.e. better generalization observed for neural network.



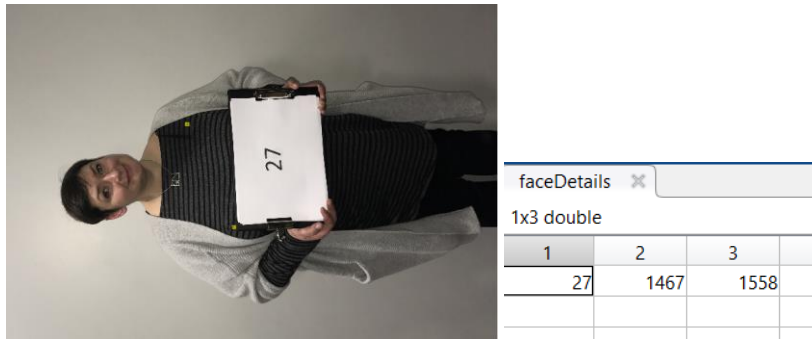
Note: Calculations exclude unlabeled images and those not identified by Viola-Jones algorithm

Due to lengthy training time and limitations of time, **5-fold cross validation** was mainly done for the final two model. It was observed that HOG-NN (0.0293) had lower error than HOG-SVM (0.03564). Confusion matrix for the test set for various models is given below.

Confusion Matrix Accuracy			
Feature/Classifier	SVM	FFNN	RF
HOG	0.987	0.981	0.899
SURF	0.977	0.943	0.91
LBP	0.99	na	0.927

LBP – Feeforward could not be modeled due to memory constraints on intel I7 quad-core with 16GB RAM.

As seen above accuracy was good for single images, but rotation was a problem. However, the code deals with it neatly on faces rightly identified by Viola-Jones algorithm. For e.g. rotated image below was rightly identified as one face and RecogniseFace function return the label along with face centre coordinates.



```
Single photo detected
HOG features extracted
SVM prediction done
Label of single photo is: 27, x-coordinate = 1467, y-coordinate = 1558>>
```

When tested with non-face image (in this case, a tree), below result was returned by RecogniseFace function.

```
confMatHogSvmTest      37x37 double
confMatHogSvmTrain     37x37 double
faceDetails            []      No faces detected
featureType            'HOG'   Not an image of person(s)
imageName              'Tree.jpg' Label of photo is: , x-coordinate = , y-coordinate = >>
```

As seen below, an array of labels and [x,y] coordinates were returned by RecogniseFace function on group photo.

Editor - FaceRecognition.m								
faceDetails								
39x3 double								
	1	2	3	4	5	6	7	8
1	11	423	430					
2	23	1031	437					
3	15	1123	435					
4	36	1275	437					
5	35	562	446					
6	5	627	453					
7	12	748	453					
8	12	506	485					
9	5	798	484					
10	13	1385	432					
11	13	922	434					
12	34	1240	462					
13	31	1135	470					
14	27	601	470					

```
Command Window
Completed face:36
Completed face:37
Completed face:38
Completed face:39
Face recognition on group photo done! 39 faces identified>>
```


The final model was observed to be handling some occlusions, recognizing slightly disoriented faces and small far-away faces in group photos. The model can be reasonably relied on for prediction due to its good test set accuracy and generalization results. However, illumination and blurring were observed to affect accuracy. As the model relies on Viola-Jones algorithm, it can only recognize faces identified by the algorithm's bounding box. Similarly, non-faces and unlabeled faces detected by the algorithm also get label prediction. Lastly, the two final models were observed to correctly identify almost all faces between them even though prediction wasn't completely accurate for each model. Therefore, it might be a good idea to develop an ensemble of models for prediction so that prediction is more frequently accurate. In doing this, memory will be very critical and therefore GPU programming must be embraced in future work.

Finally, as images from photos had some problems with rotation, this was taken care through a custom function that corrected the orientation. For e.g. if bounding box wasn't identified in a test image due to rotation, then the function rotated it by 90, 180 and 270 degrees until orientation was right to identify a bounding box. However, if Viola – Jones algorithm were to identify bounding box in the incorrect rotated position then the function and the entire prediction mechanism cannot be relied to make accurate predictions.

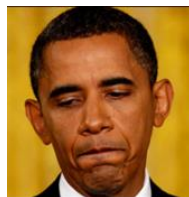
EMOTION RECOGNITION

Emotion recognition was performed in the same way as face recognition. The feature and classifier for best performing face recognition model was used to train a new classifier to identify four emotions labeled as: 01- Happy, 02 – Sad, 03 – Angry and 04 – Surprised. Test data consisted of faces that expressed one of these emotions and were organized into four corresponding folders. Training was done on 90% of the data and testing on 10% of unseen data. Performance evaluation was done with confusion matrix. A separate hold-out test set of two popular photos – smiling Mona Lisa and sad Obama were also correctly predicted.

There is only two deliverables: (i) Main script to execute: *EmotionRecognition.mat* and (ii) Trained model *HogSvm_Emotion*.

confMatTest					confMatTrain				
4x4 double					4x4 double				
1	2	3	4		1	2	3	4	
409	3	0	0		35	10	1	0	
6	294	0	0		4	28	1	0	
0	2	93	0		2	5	4	0	
0	0	0	66		2	2	0	3	

accuracyTest 0.7526
accuracyTrain 0.9863



Emotion prediction done with HOG-SVM

Image name is: monalisa.jpg and the Predicted label is: 1 >> EmotionRecognition

Emotion prediction done with HOG-SVM

Image name is: sadObama.jpg and the Predicted label is: 2 >>

AUGMENTED REALITY

Augmented reality(AR) is the enhancement of real world view by superimposing computer generated video, 3D object etc. AR 3D viewers let users to place 3D models in their environment with or without the use of trackers. (Tracker is an image to allow model attachments in AR).It has wide-ranged application due to its ability to render real world experience to end users.

1) Requirement

The objective was to develop a MATLAB function that accepted a video file and augmented it on to a hand in outstretched pose by tracking the hand movements. The function was to take video input and display the video by augmenting it on to the moving hand. The function was required to have the form: highFive(filename)

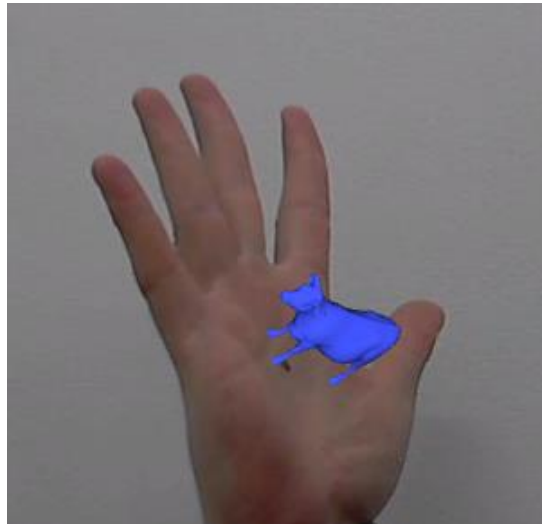
2) Deliverables

The above requirements were fully implemented and a brief summary is provided below.

- a) Function name: highFive
- b) Input arguments: videoFileName (type: string)
- c) Output: augmented display
- d) Supported input file types: 3D object, video, image
- e) Final function: Optimisation with non-linear least squares error minimization and 3D projection are available at:

3) Implementation

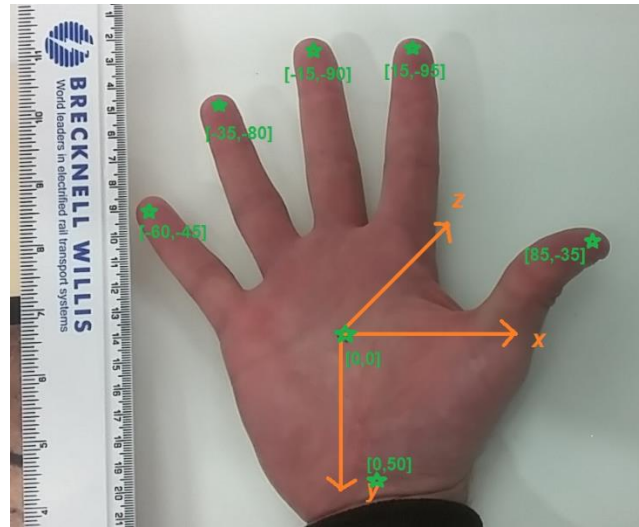
Two objects involved in the task are the moving hand and the 3D image of a cow. As the cow was required to be projected onto the hand, the points of the hand were identified manually from the finger edges. Though these coordinates were fixed, movement of the hand in the video caused them to change their relative position. As the cow was to be projected to the hand, the available coordinates of the hand were tracked with KLT tracker algorithm during the hand movement. This enabled the cow to be projected to the moving references thereby giving impression of cow moving along with the moving hand. All these were accomplished through model-pose estimation for this augmented reality task.



Firstly, a custom hand coordinate system was defined using the hand image with the ruler. The dark point at the centre was arbitrarily made the origin and all other coordinates $[x, y]$ were defined accordingly in millimeter. In these coordinates, the hand and cow had unique position and heading. Secondly, the 3D location and orientation of the cow in the custom coordinate system, also called as Extrinsic parameters, was represented by 3D translation vector (T) and 3x3 rotation matrix (R). Mathematically it is computed as Extrinsic parameter matrix = $[R|T]$. It is an invertible rigid body

transformation with 6 parameters – 3 for position (\square_x , \square_y , \square_z) and 3 for orientation (t_x , t_y , t_z). Then, the camera Intrinsic parameters (K) along with Extrinsic parameters were used to compute the cow's 2D projection points onto the 2D hand. This is referred to as Projection Matrix (M) and is computed as $M = K * [R|T]$. Every time the projection points were computed, the error was minimized through non-linear least squares optimization.

A sample of the custom defined hand coordinate is shown below.



Steps:

- i. Environment was initialized by loading (extrinsic and intrinsic) camera parameters and 3D image of the cow.
- ii. Custom 3D points coordinate system for the hand was defined.
- iii. 2D points for the hand from the first frame of the moving hand video was obtained
- iv. KLT tracker algorithm was invoked by initializing it with 2D points from first frame of the video.
- v. The moving hand video was intercepted frame by frame with the below steps:
 - a. Undistorted the current frame as model-pose estimation assumes distortion free lens.
 - b. Verified if the initialized points were present in the first frame.
 - c. Displayed first frame of the hand.
 - d. Computed 2D points to project the cow from camera intrinsic parameters, custom 3D coordinate system for the hand, 2D points of the hand and arbitrary initialization point for projection. This was optimized through non-linear least squares optimization to estimate six pose parameters - (\square_x , \square_y , \square_z) and (t_x , t_y , t_z) from 2D points of the hand, 3D coordinates of hand and their projection.
 - e. Computed extrinsic parameters using optimized 2D projection points above
 - f. Computed the Projection Matrix using extrinsic and intrinsic parameters.
 - g. Projected the cow to the 2D hand coordinates with the projection matrix, translation parameters and a few other parameters.
 - h. Reset the KLT tracker to have the latest hand coordinates that were detected in the current frame
- vi. Steps from v [a-h] were repeated until all frames in the video were read

4) Analysis of final solution

Challenges: This task was challenging for every part of it like developing custom coordinate system, calibrating camera, computing 2D projection matrix for every frame, calculating reprojection error through non-linear least squares optimization etc. Eventually, when it started to work some aberrations were observed in the movement and they were corrected as below.

1. Initially when the cow was projected, the cow was left in its original projection point as the hand moved. Soon it was realized that the projection matrix wasn't getting the points
2. Initially the cow was loaded in every iteration and this slowed down the view. Performance drastically improved upon moving it out of the loop and loading it only once.
3. Upon projecting the cow for the first time, it was very bouncy. This was due to the initialization of the starting point for ReprojectionError to a fixed value all the time. Upon changing the initialization to the last optimized projection point, the bouncing was reduced drastically.
4. Further smoothening of the movement of cow along with the hand movement was done by averaging the projection matrix to avoid wide variations in the values in every iteration.

This piece of software is likely to work without much issues. However, it can be modified to obtain the hand coordinates automatically through segmentation. Though Chan-Vese segmentation was attempted, this route wasn't successful. If this works, then any other 3D image can be projected to any other moving object. These thoughts are deferred for future trials.

REFERENCES

1. http://bytefish.de/blog/local_binary_patterns/
2. Wikipedia.org
3. <http://stackexchange.com/>
4. [Fundamentals of Computer Vision book by Mubarak Shah](#)
5. [Computer Vision: Algorithms and Applications book by Richard Szeliski](#)
6. [Photogrammetry tutorial by Cyrill Stachniss](#)
7. [Research project in LBP applied to Face Detection and Recognition by Laura Sanchez Lopez](#)

RESOURCES SUBMITTED

1. This document briefing the coursework implementation (on Moodle)
2. Code for Augmented reality – 7 files (on Moodle)
3. Code and model for Emotion recognition – 3 files (on Moodle)
4. 6 face recognition models and 1 bag of features for SURF (on my google drive due to space constraints in Moodle). The link is:
<https://drive.google.com/drive/folders/0B5eoGE3DvICWd3RwMW50T1NSMUE>.

