Computer Vision Coursework 2017

Face Recognition and Augmented Reality

Miguel Esteras-Bejar, *City, University of London*

1. ReadMe **2**
How to run the function 2
2. Face Recognition **3**
2.1 Face Detection 4
2.2 Feature Extraction with Auto-Encoders 4
2.3 Feature extraction with Histogram Oriented Gradient (HOG) 5
2.4 Bag of SURF features 6
2.5 Training a Softmax Classifier 6
2.6 Train a Random Forest classifier 8
2.7 Train a N-Nearest Neighbours Classifier (KNN) 9
3. Tracking and Augmented Reality **10**
4. Discussion **11**
5. References **11**

1. ReadMe

The motivation of this project is to put into practice the skills and knowledge acquired in the Computer Vision Module. The first task of this project will compare the performance of 2 different methods for the extraction of features from images (HOG and SURF), and the classification performance of three models (Random Forest, K-nearest neighbours, and Softmax neural network) for each of the feature extraction methods, resulting in a total of 6 combinations. The second task uses augmented reality in a toy video showing the hand of Dr Gregory G. Slabaugh.

How to run the function

P = RecogniseFace(image,featureType, classifierName) returns a matrix of size Nx3, where N is the number of faces detected in the image. P(:,1) contains their predicted labels. P(:,2) contains the x coordinate of the center point of the face. P(:,3) contains the y coordinate of the center point of the face. Valid arguments for featureType are; 'HOG' and 'SURF'. Valid arguments for classifierName are; 'KNN', 'RF' and 'SoftMax'.

To run this function <RecogniseFace> (and the one used in the second task, <highFive>), all folders and subfolder contained in the submission package needed to be added to the matlab paths. They contain auxiliary functions and trained models.

2. Face Recognition

Creating a face detection and identification function (<RecogniseFace>) involved the construction of a database containing the faces of students, TA and lectures part of the Computer Vision Module. Faces are detected and extracted from pictures and video frames provided by the university. Features are then extracted from these face images, and used to train a classification model. The models are built by supervised training, using the unique number identifying each individual as label. The classification accuracy of each classifier is validated once with a set of face images from the database (15-20%) that were not used for training (1-fold validation).

When a new picture is given to the function, all faces in the picture will be detected. Then, the image features in the area of the face are extracted, so that these features can be used as input into one of the trained classifiers. The model output is the predicted face label (fig. 1).

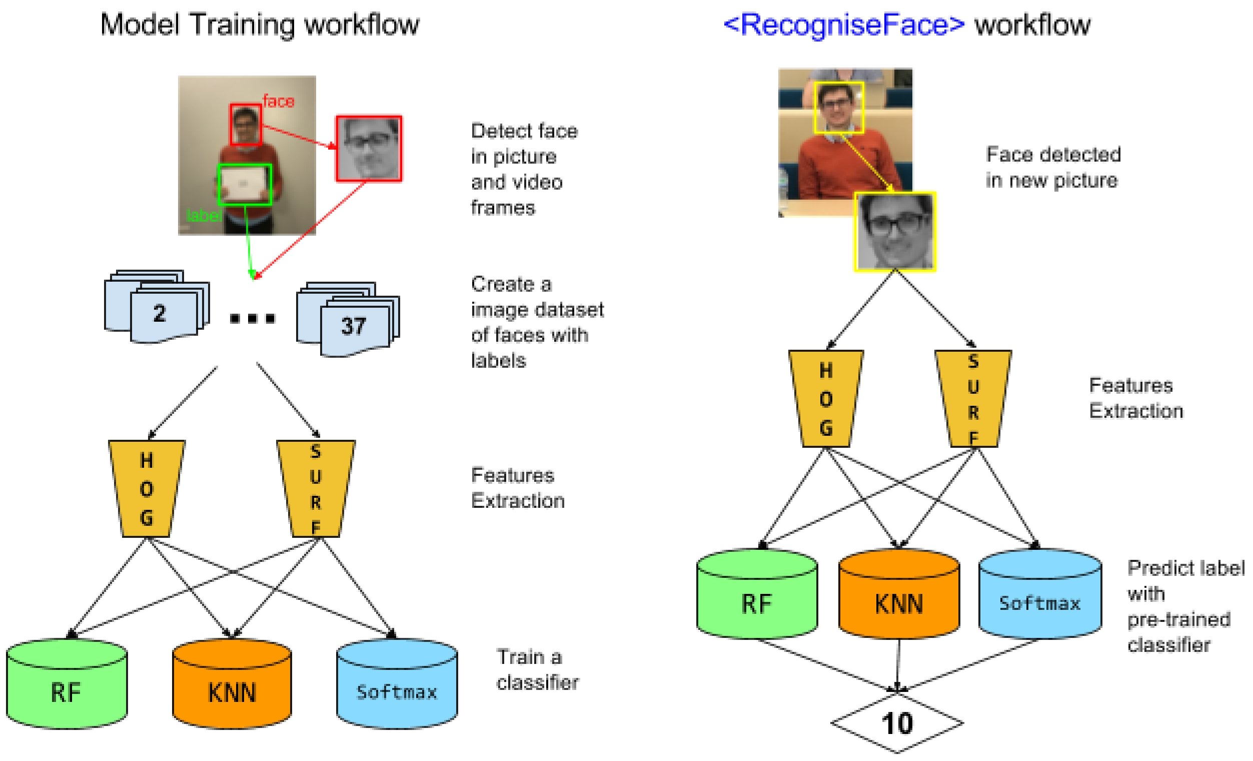


Figure 1. On the left, the training process workflow. Faces are extracted and pre-processed to build a database. SURF and HOG feature extraction is used to create a input vector for the classifier (Random Forest, K-nearest neighbours and Softmax Layer). On the right, the function workflow. When a new image is given, the function detect the faces, extract their features and, based on them, predict a class label (number).

2.1 Face Detection

The dataset contains a total of 29829 images nearly equally divided between the 37 classes (37 different people). Some of the images were pictures captured, but most of them were obtained by extracting video frames (extractVideoFrames.m, included with supplementary documents). Some images had to be rotated and resize to smaller scale to ease computation and facilitate the detection of faces. A DCT normalization algorithm was applied to remove illumination effects that could decrease face detection accuracy (<DCT_normalization> function is part of the INface toolbox by Vitomir Struc, included with supplementary documents). The normalize image is then scanned by a “CascadeObjectDetector” that recognized upper bodies, from the Matlab computer vision toolbox. The minimum size of the object area is set to 150X150 pixels, and the merge threshold to 5, to reduce false positives. A new “CascadeObjectDetector” that recognizes faces is then applied to the upper body area, detecting areas of minimum size 60x60 pixels and with a merged threshold of 5. If a face is detected, the face area is cropped from the image and its size normalized to 40x40 pixels (to ease further processing). If no upper body or face are detected, the pixel intensity of the DCT normalized image will increase by a factor of 1.2 (brightening the image) and the same “CascadeObjectDetector” will be used.

This script (cropFaces.m, included with supplementary documents) returns a near 100% accuracy in detecting and extracting faces from single person pictures, and about 90% accuracy for group pictures. This face detection technique is used to create a dataset of images of faces for training and to detect face in new images inside the <RecogniseFace> function.

The face labels in the training set were initially extracted from the images using Optical Character Recognition (OCR). Before applying this technique image preprocessing was used to detect and set to 0 all pixels values outside the white board containing the label. Despite this, the performance of the OCR function was never better than 70%. Hence, manual allocation of images into folders named after the labels was used.

2.2 Feature Extraction with Auto-Encoders

Extracting features with an Auto-Encoder network was initially tried and then discarded due to its poor performance. Hence it is not part of the final function.

Autoencoders are unsupervised learning algorithms, that take input data x_i without labels, and compress (encode) it to z_i. The data z_i is then decompressed (decode) back to $\bar{x}_i$, which approximates the input data x_i. During the training of the autoencoder, the objective is to minimize the sum of squares differences between x_i and $\bar{x}_i$, which can be achieved by

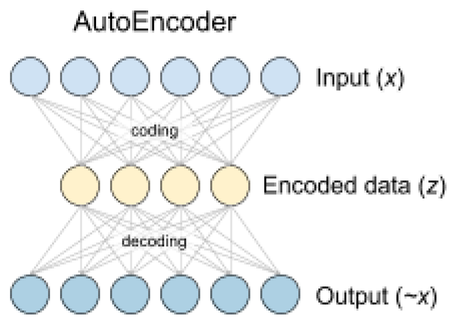


Figure 2. A autoencoder network for unsupervised learning. The network learns how to encode (compress) the input data and then decode it into an output that approximates the input data.

stochastic gradient descent (fig.2). Given an input image, the coding process will transform the pixel values of the image into a smaller size vector containing the features needed to decode the vector back into an approximation of the original image. In practice, the Auto-Encoder trained during this project failed to learned filters to extract descriptive features for all the classes.

2.3 Feature extraction with Histogram Oriented Gradient (HOG)

HOG is a common feature extraction method that transforms image data into a fixed descriptor vector. The descriptor vector is created with local values of gradient orientation. The image is divided into a grid of 4x4 pixels windows (the size of the window depend on the application and image feature density). For every window, HOG calculates the gradient vector at each pixel (16 vectors in the 4x4 window) and uses them to create a 9-bins histogram in the range 0 to 180 degrees. The value added by each gradient vector to the histogram is proportional to the magnitude of the gradient, and the value is proportionally splitted between the two closest bins (fig.3). The use of histogram bins instead of gradient vector managed to reduce the size of the descriptor, from 16 gradient vectors per window to a string of 9 magnitud values. These histograms are then normalized and concatenated to form a one-dimensional vector (image descriptor). The normalization step makes the descriptor invariant to some variations in the pixel values due to illumination.

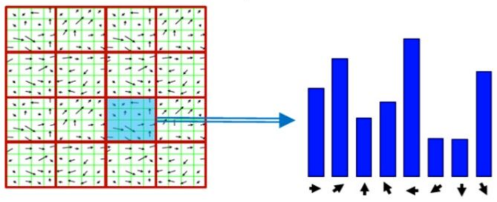


Figure 3. Histogram created during HOG feature extraction, containing the values of the gradient vectors of every pixel in the window. Extract from lecture notes.

For this project, two different set of predictor vectors are generated from the training set; one calculated from 80x80 pixel images with a 8x8 pixel window, and another calculated from 40x40 pixel images with a 4x4 pixel window (fig.4). These descriptor vector, together with target labels (unique number allocated to each person) are used to trained the face classifiers.

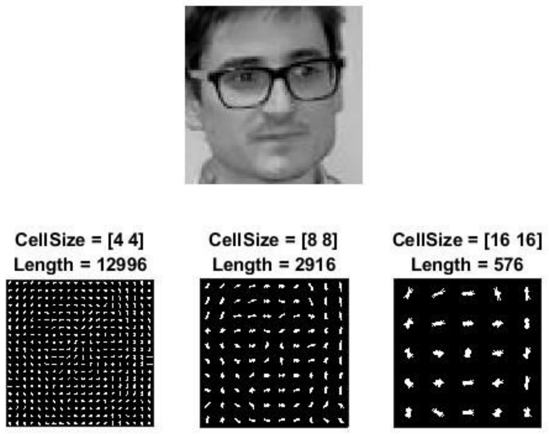


Figure 4. HOG feature extraction on an image from the training data. Descriptors resulting of different window size are shown.

2.4 Bag of SURF features

Bag of features is a technique translated from natural language processing (bag of words). There, documents can be encoded not by using every word they contain (as many of them do not give much information), but as a frequency of a set of keywords. In the same way that keyword can be predetermined to create a vocabulary, bag of features creates a fixed size vocabulary of features for all images.

In this case, the vocabulary (size 100) is formed with Speed Up Robust Features (SURF) features, a scale and rotation invariant local feature detector (Herbert Bay et.al., 2006). The vocabulary is built extracting features over all images and categories, and reducing their representation to 100 using K-means clustering. Training images can be then encode into a vector of size 100 (or histogram) containing the feature frequency (fig.5). This process eliminates the spatial information of the features.

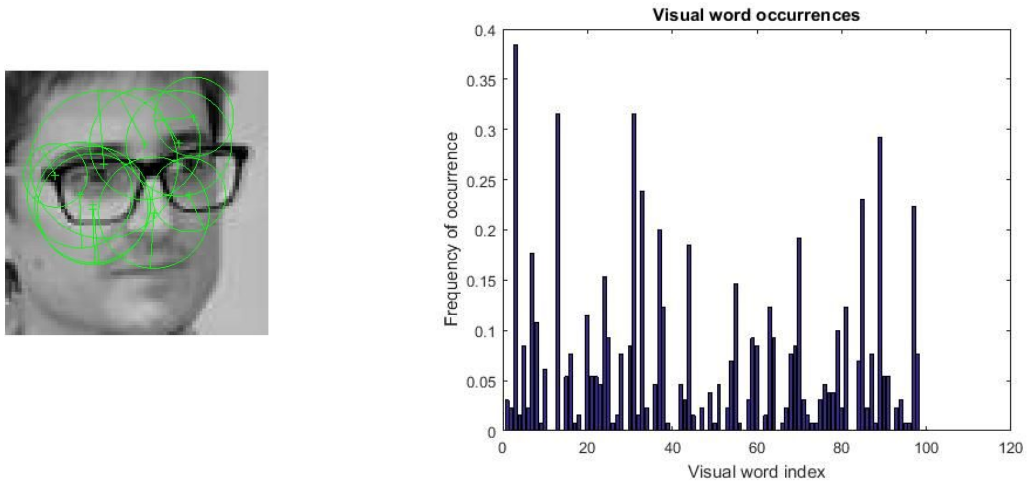


Figure 5. Bag of features histogram and SURF feature extraction. On the left, local features are extracted with SURF. On the right, these features are then mapped into a 100 bins histogram. Each word can be encoded into a vector of size 100 that contains the values of the frequency of the binned features for that word.

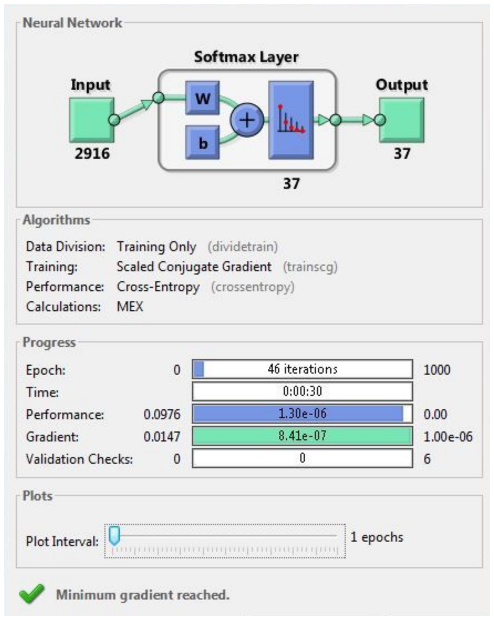
2.5 Training a Softmax Classifier

Softmax is neural transfer function often used in the final layer of a network to calculate probability scores for classification problems. The Softmax layer can also be used as a stand alone classifier which inputs vector (of image features) and linearly transform into output class scores, which are interpreted as the unnormalized log probabilities of the input belonging to each class label. This classifier model can undertake supervised training and adapt to the training data by backpropagation learning.

The main advantages of this techniques include a small computational time, and the calculation of probabilities for all classes. Hence the true class probability might be used for training and prediction even if not being the highest score (eg. rank-3 accuracy = true label is within the top 3 scores). On the other hand, a stand alone softmax classifier might perform poorly if the data is not linearly separable.

The softmax classier training with HOG features extracted from the images 40x40 pixels stopped after 46 epochs (33 seconds) due to the early stopping rule of minimum gradient reduction. At this point, the cost function (cross-entropy) was near 0. This model predicted both the training data labels and the validation data labels (images not used for training) with 99.98% accuracy.

Training a Softmax classification neural network with SURF features as input resulted in model with a similar performance accuracy (100% in training set, 99.73% in validation set).



Algorithms
Data Division: Training Only (dividetrain)
Training: Scaled Conjugate Gradient (trainscg)
Performance: Cross-Entropy (crossentropy)
Calculations: MEX

Progress

Epoch:	0	46 iterations	1000
Time:		0:00:30	
Performance:	0.0976	1.30e-06	0.00
Gradient:	0.0147	8.41e-07	1.00e-06
Validation Checks:	0	0	6

Plots
Plot Interval: 1 epochs

Minimum gradient reached.

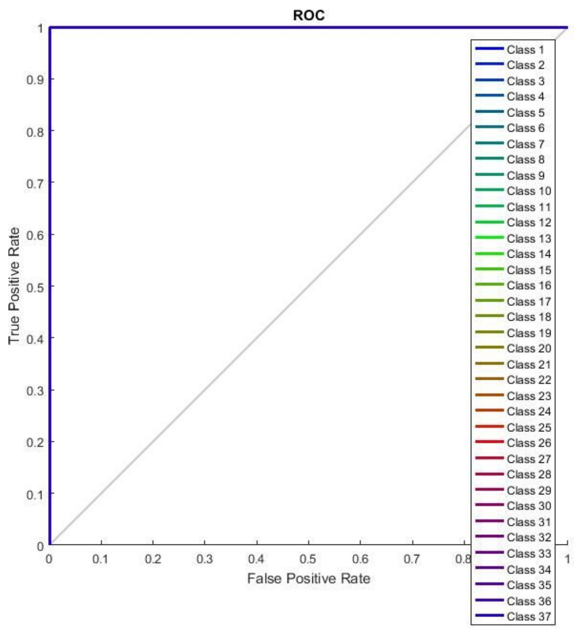
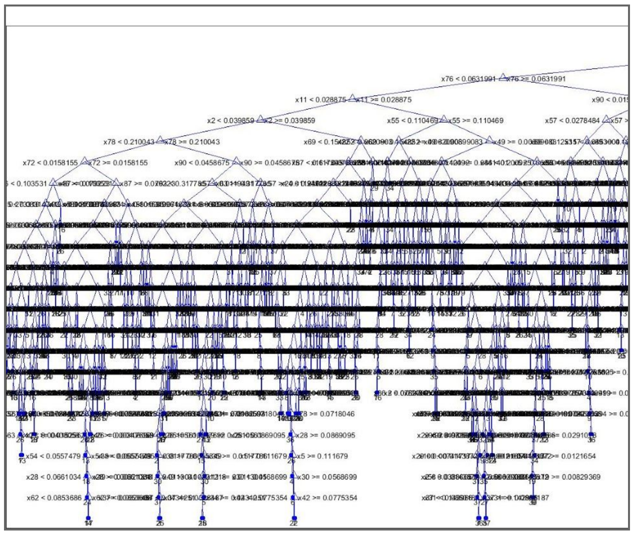


Figure 6. Softmax classification neural network performance. On the left, shows the early stopping parameters that triggered the end of training. On the right, The ROC curve showing a near perfect performance on the classification of the validation set.

2.6 Train a Random Forest classifier

This model is trained with the matlab implementation of Random Forest (Breiman, 2001), called <TreeBagger>. These algorithm trains a ensemble of decision trees each of them grown using a random subset of the training instances (bootstrapping with replacement). In addition, the feature used for split at each node is also sampled from a random set of features, adding variability between trees and reducing the risk of overfitting. Contrary to the Softmax classifier, Random Forest does not make any assumption on the distribution of the data although its computational cost is considerably greater. When a new input (encoded image) is given to the model, it will be send thought all grown trees and estimate a score probability for all classes (similar as in Softmax classifier).

Initially, a forest containing 1000 trees is grown. However, no improvement in performance is observed after 200 trees, hence the final model are limited to this size. As for the other classifiers, the training accuracy was 100% in both cases. The validation accuracy was 99.68% and 99.53% for HOG and SURF model respectively.



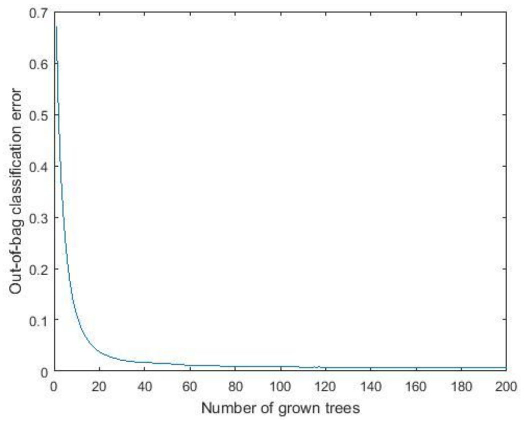
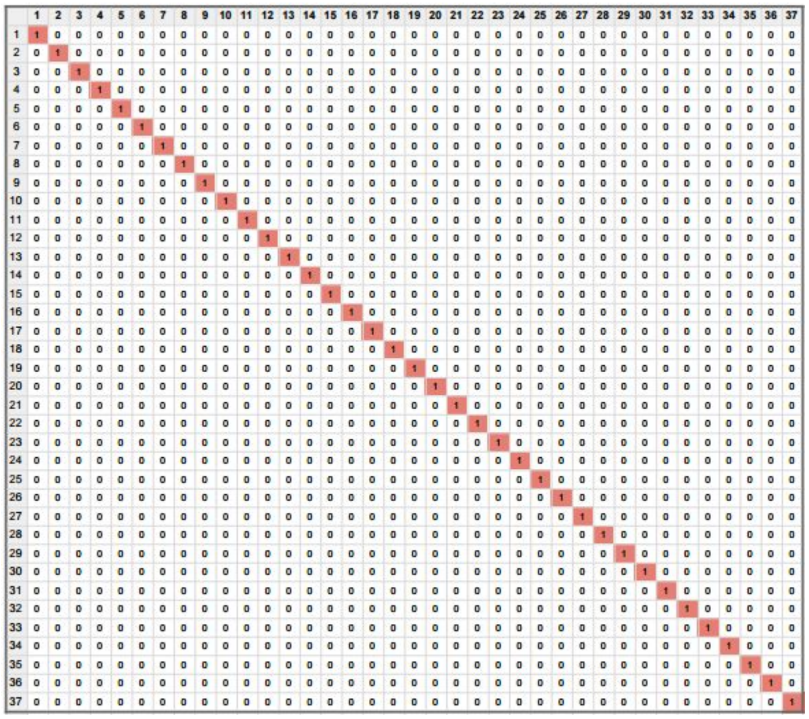


Figure 7. Random Forest training. On the right, a section of a tree. Trees are not prune as the overfitting that this might create is eliminated with the averaging of all trees in the forest. On the right, the classification performance (out of bag classification error) decreasing with the size of the forest.

2.7 Train a N-Nearest Neighbours Classifier (KNN)

The N-Nearest Neighbours Classifier is a conceptually very simple unsupervised algorithm that classifies new instances based on a similarity measure. Contrary to the SVM, the KNN classifier needs to retain in memory all training instances. The predicted class of a new example is calculated by majority vote of the N-neighbours. Neighbour instances are defined by the measure of a distance function (eg. euclidean, manhattan, minkowski,...). Training hyperparameters, namely the value of N and the distance function are calculated by automatic hyperparameter optimization part of the <fitcknn> function in matlab. This method returns the value N and the distance metric that minimize the loss function (by 5-fold cross-validation).

Best results were observed using the minkowski distance function in both the HOG encoded image space and SURF encoded image space. Due to computational constraints the KNN model trained on HOG encoded data only used half of the training data (~12.000 instances). While the model trained on SURF encoded data used the full training set (in this case SURF vectors are considerably smaller than HOG vectors, hence the computation cost). The classification accuracy of both models is 100% in training set, and 99.97% and 99.83% in the validation set for HOG and SURF trained respectively.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 8. Confusion matrix resulting on the validation of the KNN model for SURF encoded instances. The values in the matrix are given in percentage form, and showing a near perfect classification (99.83%).

3. Tracking and Augmented Reality

highFive(videofile) will render a red cow into Dr Gregory G. Slabaugh hand.

Initially, the <highFive> function defines a 3D coordinates system with origin in the center of the hand (x mark). Under this system, 6 3D points are defined, one at the origin [0,0,0], and one at every finger tip of the hand (eg. [42.8 -10 0] for the thumb). The distance between points is measure in mm, and all points are assumed to initially be in the same plane (z=0).

These points also define the 2D coordinates in the image were to place the tracking points that will follow the hand as it moves in the video. All these calculations are done during the first frame of the video.

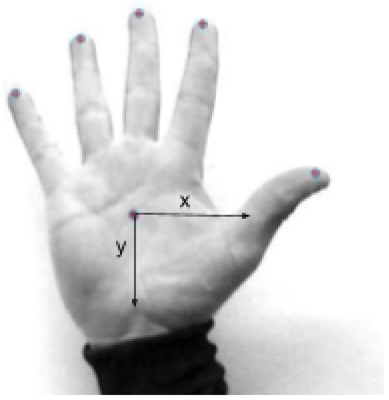


Figure 9. Coordinate system used to describe the position of the 3D points at the fingertips. The 2D projection of these points is used for tracking of the hand during the video.

The 2D points are tracked in every video frame using vision.PointTracker() in matlab. For every frame, the new location of these points is used to calculate the projection error (using function <Pro_error> function, included in the folder). The projection matrix is then calculated and stored for every frame.

Finally, a 3D object (red cow) is projected into each of the 2D frames of the video, using the projection matrix so its location changes accordingly to the position in the 3D coordinate system of the 2D points tracked during the video. As a result, the cow stays in the hand and moves with it for the duration of the video.

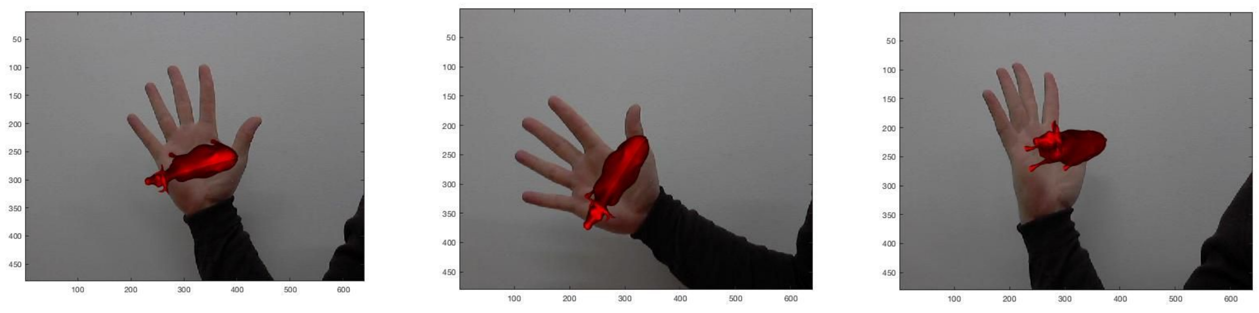


Figure 9. A red cow is projected to each 2D video frame tracking the movement of the hand accordingly to the projection matrix.

4. Discussion

Considering the high accuracy of the model given good quality data, the most difficult step remains to perform an accurate face detection. This was particularly challenging considering illumination variance and face rotations.

The output of the classification model might be used for other purposes other than classification. The score probabilities returned by Random Forest and Softmax classifier could be used to investigate the similarities between classes and between classes of other nature (eg. dog faces, books,...). Similarly, KNN distance metric can be used to calculated similarities, or to represent instances in a lower dimensional space for visualization. In addition, these same model, or a combination of them, could be used for the detection of emotions in faces.

In order to built a more robust algorithm, training set should include a wider range of illumination variance and geometrical transformations on the faces, including different background. The models trained in this project are expected to decrease performance on this circumstance. If this is the case, a pre-trained CNN (eg. facebook face recognition) could be transfer to the function to perform the feature extraction instead of HOG or SURF.

The 3D object projection requires a lot of computation hence could not be done in real time (projectionmatrix was pre-calculated for every frame). Improvement in the algorithm speed and a more robust point tracking system could result in a near real time accurate projection. In addition, these same principle could be used to project a moving 3D object, considering a 3D object whose vector structure (not only projection) changes at every frame.

5. References

Breiman, L. Machine Learning (2001) 45: 5. doi:10.1023/A:1010933404324

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, 2006. SURF: Speeded Up Robust Features , European Computer Vision Conference (ECCV).

INface toolbox by Vitomir Struc:
https://uk.mathworks.com/matlabcentral/fileexchange/26523-the-inface-toolbox-v2-0-for-illumination-invariant-face-recognition