Face Recognition and OCR Report

The objective of this project was to develop two functions.

The first function P = RecogniseFace(I, featureType, classifierName) returns a matrix P representing the people present in an RGB image I. P is a matrix of size Nx3, where N is the number of people detected in the number of image. The three columns represent

1. id, a unique number associated with each person that matches the number in database provided

2. x, the x location of the person detected in the image (central face region)

3. y, the y location of the person detected in the image (central face region)

The second function, detectNum (filename) accepts an image or video file of a person holding a number in their hand and detectNum returns the number seen in that image/video.

Matlab was chosen for its ease of use.

Face Recognition

To make it easier to train faces, I used the face detector function to recognise each face in every image. Once the face was recognised, the image was cropped for just that region and resized.

Since there was an unequal number of images I adjusted the images using a min set Count and a partition function to balance the image set. I partitioned the data sets into a training, validation and testing set.

To increase performance, I used Matlab's bag of features to extract SURF features. Then I ran a trainImageCategoryClassifier which is an SVM. The confusion matrix showed a high precision accuracy of 99% on the training set and 95% on the validation set.

Challenges

I encountered some challenges. For example, I wasn't able to save the classifier. The save function allows the classifier to be saved and loaded as a structural array. However, the predict function only works with a classifier. I spent a lot of time on the OCR so unfortunately I ran out of time to train all the different models. In future, I would make a plan budgeting ample time to train models and debug Also, I would investigate more how to save classifiers.

Additionally, Matlab often would overload the CPU or crash causing many delays, even though I was using the state of the art MacbookPro. I found the data type conversions unnecessarily complex. In future, I hope to use python and open cv, using a GPU.

OCR

The images contain pictures of people holding up white pieces of paper with a number on it. The first approach attempted was to label the white rectangular background using the image labeller app. Maybe there would be a way to recognise these white pieces of paper within the image and that way the OCR could be restricted to this white area. I tried to train an Object Detector from Ground Truth Data. However, the labels that were created didn't get good results as labels. An alternative method would be to train a neural network to recognise these white areas.

First I used the several Matlab preprocessing steps. These included, detecting candidate text regions using MSER, removing non-text regions based on basic geometric properties, removing non-text regions based on stroke width variation, merging text regions by using the bboxOverlapRatio function to compute the pair-wise overlap ratios for all the expanded bounding boxes, then use graph to find all the connected regions. I drew much from this Matlab Tutorial:

(https://uk.mathworks.com/help/vision/examples/automatically-detect-and-recognize-text-in-natural-images.html)

Challenges

There were some images that didn't classify correctly: Some images were rotated

I coded a code block that rotated the image then compared the confidences with each other. Whichever 'word' confidences were bigger, it returned that image. If the image was rotated, This would return a rotated image with the correct classification.

And some classifications were picking up other numbers or just recognising one number

I solved a lot of these misclassifications by increasing the bounded boxes expansion rate from 0.02 to 0.04.

For very few examples it doesn't classify correctly as the writing on the fire hydrant looks like numbers. Sometimes there is a bright white light on the side of the image and the ocr has a higher confidence there, thereby misclassifying the numbers.

A possible solution would be to only allow black writing in the ocr.

Also, some of the rotated images had a higher confidence of classifying numbers as 00 that were sideways. For example, 68 was classified as 00. Instead the image needed to be rotated and the number classified correctly. To solve this, I hard coded that the final number could never be 00. This helped in many cases to get the right classification.

A lot of work occurred, understanding how to use compatible commands in matlab. For example, understanding the difference between read and imread. Additionally, it was necessary to convert different data types often, which was fiddly.

References: https://uk.mathworks.com