**CSC4067 - Advanced Intelligent Information Systems  
Pedestrian Detection**

**David Jack - 40057705  
Timothy O'Neill - 40062164  
Nathan Sullivan – 40059116**

**1. Introduction and Background**

The goal for this project is to implement a pedestrian detector in MatLab using various machine learning algorithms. The general process for this is to take a set of images which we know either contain or do not contain people, extract features from them using an algorithm like HOG or PCA and create a model from those features which other testing images can then be compared against to predict whether or not the image contains a person or not.

There are two main parts to our pedestrian detection; the feature extractor and the classifier. The role of the feature extractor is to reduce the amount of resources required to describe a dataset and allow the classifier to work on a more generalized set of data by extracting relevant information from the data.

The classifier is the part of the system which predicts whether or not a testing image matches one group or the other. In our case that is predicting whether or not an image contains a person. After being given a set of training data to learn from, the classifier will have a split between people and non-people and then when a testing image is given to it, it will extract the features from the testing image and see where it lies within the model.

An issue to keep in mind during feature extraction and classification is that of overfitting[1]. Overfitting can occur when the model is trained to too specific a feature and then is incapable of making correct predictions afterwards. An example of what could cause this would be a system being trained to recognize faces being trained with 1000 images of a single person’s face as being faces and a wide array of other objects as not faces. If the testing face is not the person the system was trained with, it would be unlikely to recognize it because it was trained too much with the training face.

Before implementing the pedestrian detection for video, we first tested multiple combinations of feature extraction and classification to see which one would be most suited to our needs based on a combination of prediction accuracy and computation time. The accuracy of each combination is required because without an accurate prediction model we would have the same amount of success if we took chunks of video and randomly assigned a prediction to it. The reason we need to consider computation time when choosing a feature extractor and classifier is that for a video on disk we may have as much time as we need to process it, but for any application which requires speed or detection in real time such as CCTV we require a method that takes at worst 500ms per prediction for a 2 fps video.

**2. Training**

**2.1 Pre-Processing**

When we are loading images to train our system, we are preprocessing them to improve results and reduce the time taken to process. The testing images are also being processed in the same manner before being tested against each model. We are preprocessing these images in 2 main ways:

* Grayscale Conversion
* Normalisation

Grayscale conversion is simply converting each of the images used from RGB to 255 shades of gray. As part of this process, each image is reduced from having 4 colour channels (red, green, blue, and alpha) to only 1 (luminance) which cuts the amount of memory we require by approximately 75% and improves the speed of comparing images by approximately the same amount. While some information may be lost in this conversion, the effects go mostly unnoticed as most of the image’s information is stored in the luminance plane which we retain in grayscale while we lose the chrominance planes which hold far less information. This can be seen below in figure 1. The top left quadrant shows the full RGB image, the bottom left shows the luminance plane (grayscale) and the two on the right show the chrominance planes which we drop when we convert.



Figure 1 - Luminance and Chrominance

The second method used is normalization which simply consists of converting the grayscale image into a set of double values. This changes the range of intensity values from the grayscale image from 0..255 to 0..1. To normalise our images we used the MatLab built in function im2double[2] which also converts the 0..1 value to fill the entire dynamic range of the image.

Two other preprocessing methods we attempted were gamma correction with the goal of balancing brightness across all of the given images. Unfortunately this had a noticeable negative impact on accuracy for almost every feature extractor and classifier combination, up to a 7% decrease as seen in table 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Extractor/Classifier | No Gamma | Gamma | % Decrease |
| NN Raw | 0.71278 | 0.67778 | 4.91 |
| NN HOG | 0.71889 | 0.66333 | 7.73 |
| NN PCA | 0.77778 | 0.72056 | 7.36 |
| KNN9 Raw | 0.71778 | 0.72778 | -1.39 |
| KNN9 HOG | 0.65722 | 0.63944 | 2.71 |
| KNN9 PCA | 0.73667 | 0.73278 | 0.53 |
| SVM Raw | 0.80722 | 0.74778 | 7.36 |
| SVM HOG | 0.76778 | 0.76000 | 1.01 |
| SVM PCA | 0.76778 | 0.76000 | 1.01 |
| Adaboost Raw | 0.74278 | 0.74278 | 0.00 |
| Adaboost HOG | 0.81889 | 0.79889 | 2.44 |
| Adaboost PCA | 0.71444 | 0.70778 | 0.93 |

Table 1 - Accuracy with and without gamma correction for sample rate 25

**2.2 Feature Extraction**

The following subsections contain brief summaries of the feature extraction methods we tested during the development of our pedestrian detector.

**2.2.1 Raw Pixels**

A “feature extractor” with nothing exciting. This is simply using the images as they are loaded in our classifiers. As each image is a 2 dimensional matrix of floats, it can be used as a set of numbers which can be compared to other sets of numbers. This extractor tends to have poor accuracy and long processing times making it a very poor choice.



Figure 2 - Sample of raw images

**2.2.2 Dimensionality Reduction / PCA**

Dimensionality reduction is the process of transforming a data from a high-dimensional space to a lower dimensional space. The benefits of dimensionality reduction include reduced processing time after the overhead of the initial reduction and reduced memory requirement. It also makes data clearer to view as it can reduce huge matrices to tiny ones very quickly. As an example, our raw images our 96\*160, but after applying PCA they are reduced to only 1\*29.

The method of dimensionality reduction we chose to use is PCA (Principal Component Analysis) because it is a well-known general purpose dimensionality reduction technique which doesn’t require much effort to implement.

PCA works by extracting the main components of a dataset and reducing them to a lower dimension. An example is shown below. Using dimensionality reduction, the 2d oval of triangles can be reduced to a 1 dimensional line and still have most of its relevant data (distance between triangles) left intact.

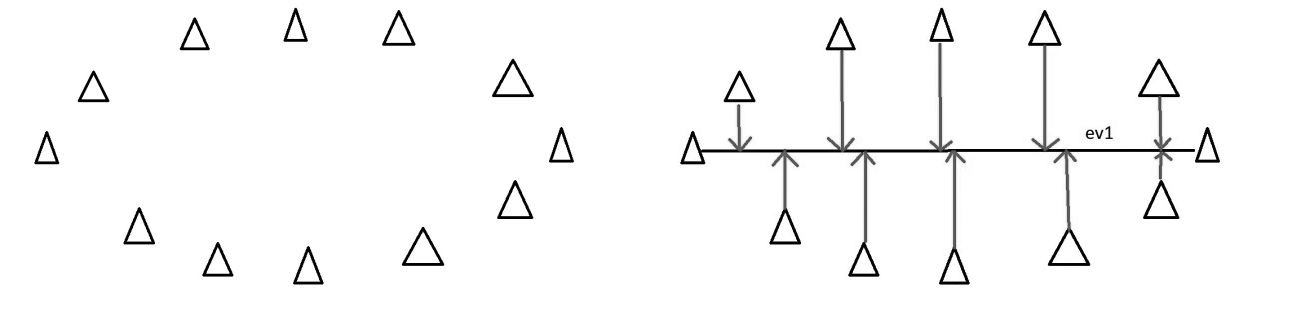


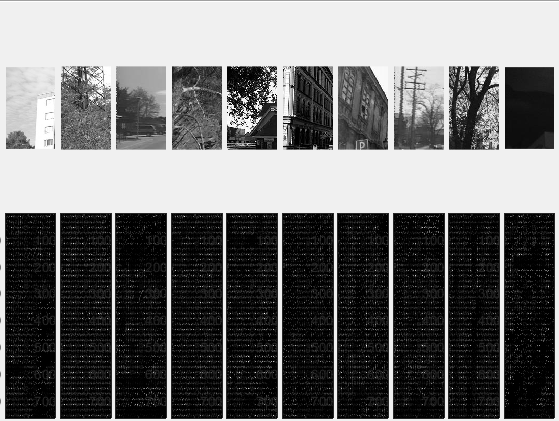
Figure 3 - Simple dimensionality reduction with triangles[3]

One major downside to PCA is the initial overhead of reducing our training images to a lower dimensionality. We are counteracting this slowness by scaling each of our training and testing images to half their size before applying PCA. This also means that any further testing images would also need to be scaled down to half size as well.

PCA’s results tend to be found very quickly, but are not quite as accurate as HOG.

**2.2.3 HOG**

Histogram of oriented gradients, divides an image in a series of cells it then looks at the gradient directions of the pixels in each cell, various stages of normalization are then performed after which the cells are combined back together, an example of this is shown in Fg[].



Hog has several advantages to it, since it operates on local cells, it is invariant to geometric transformations such as rotation and photometric transformations such as blur, because of these properties HOG permits movement of pedestrians to be ignore so long as they remain mostly upright, therefore the HOG descriptor is particularly suited for human detection in images.

**2.3 Classification**

**2.3.1 Nearest Neighbour and K-Nearest Neighbour**

Used as a simple test by giving the label of the closest image we can find. Explanation of why using K neighbours is more reliable than 1 neighbour. Include a diagram.

**2.3.2 SVM [1][2][6]**

Support Vector Machines is a supervised learning mode where we supply the algorithm a set of training examples, each sample is marked as belonging to one of two categories, SVM model is a representation of the examples as points in space, mapped in such a way that the examples of the categories are separated by a clear gap.   
  
Testing images can then be applied to the model and the results verified against some already known values.

**2.3.3 Adaboost? Is that boosting? This is all you David**

**2.4 Parameter Tuning**

We haven't even done this. What params can we tune?

**3. Testing**

**3.1 Data Split**

**3.1.1 Half Split**

We split the dataset into two halves. How and why did we do this? Explain how the split was performed (1 for train, 1 for test, 1 for train, etc)

**3.1.2 Cross Validation**

What is cross-validation in this context and why is it useful? Did we choose to use it over half split?

**3.2 Evaluation**

**3.2.1 Recognition Rate**

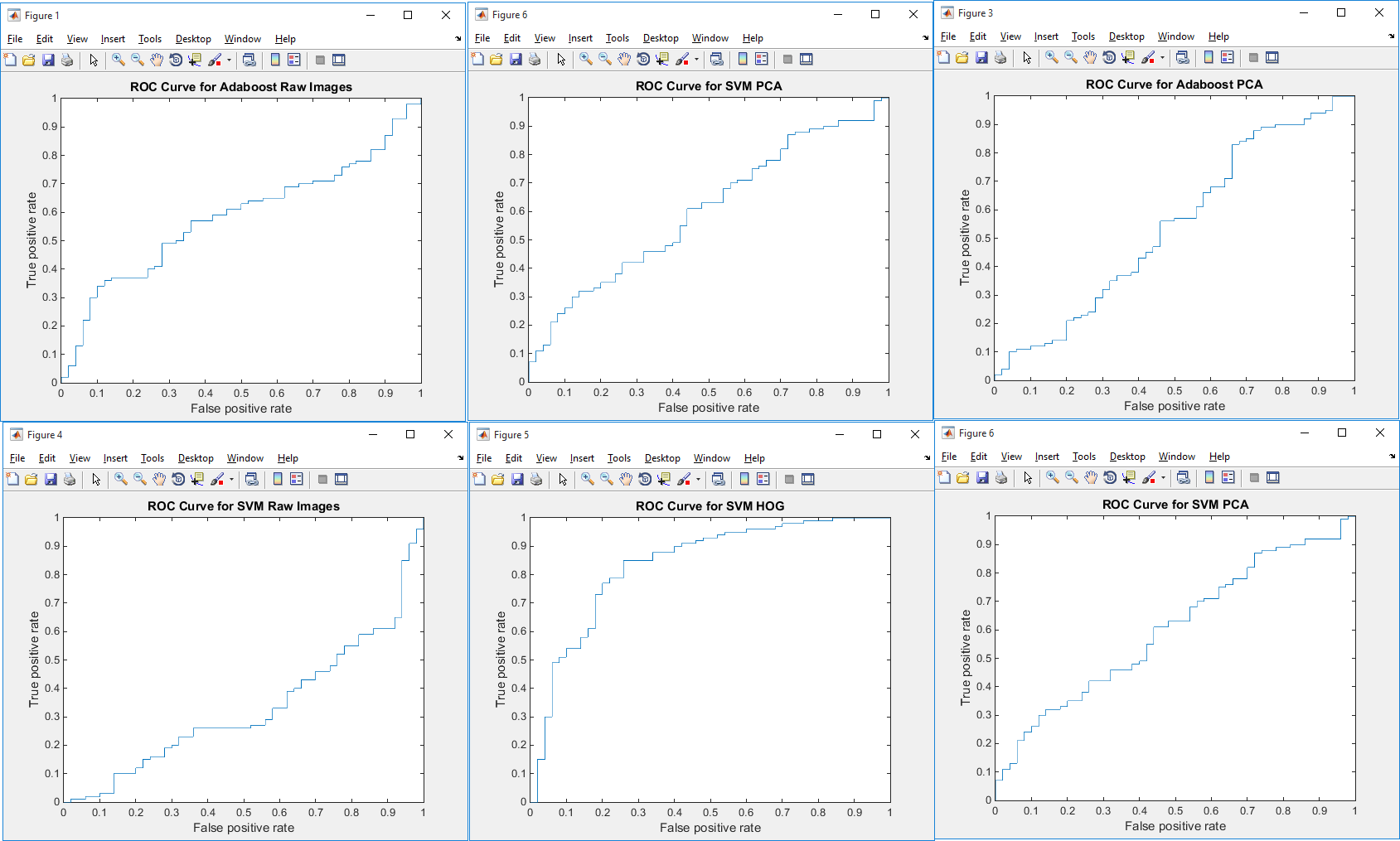
Discuss the accuracy of each feature/model combo. Table of results, especially accuracy for this section.

**3.2.2 Type 1 and 2 Errors**

Table of errors. Throw in a confusion matrix per combination for maximum spook.

**3.2.3 ROC Curves [7]**

Roc curves are created by plotting the true positive rate against the false positive rate, in our case we do this for various classifier combinations these can be seen in Fg[].



Roc curves are very useful in identifying the most optimal combination, using a built in method of matlab it is easy to plot a Roc curve with existing variables.

**3.3 Reflection**

**3.3.1 Method Comparison**

Which is our most accurate model (SVM using HOG for full sample rate, although thats not using full scale PCA)

**3.3.2 Known Failures**

Do we have any way of identifying images which are incorrectly predicted using multiple models? If so, drop it in here because that's an excellent indication of something being wrong.

**4. Detection**

**4.1 Chosen Model**

Relate to section 3.3.1. Line on why we chose it. Even if full scale PCA is faster, SVM HOG might be the better choice due to execution time.

**4.2 Sliding Window [8]**

Sliding window steps through a given image segmenting it up into smaller chunks, this individual pieces of the image then have their features extracted and classified, enabling us to locate multiple objects per image, an example of how sliding window segments up a image is shown in Fg[].



**4.3 Non-Maxima Suppression [4]**

As shown in Fg[], due to the segmenting of the image we will get multiple positive results produced for the same image area, the algorithm then places bounding boxes around what it thinks is a positive result, by using NMS however we can check for intersections of the bounding boxes that are produced by the classifier and discard the bounding box with the lower confidence, this reduces the number of bounding boxes on screen and helps deal with cases of overlap that may occur.

**4.4 Results**

**4.4.1 Comparison to test.dataset**

Show a few screencaps of our detection vs the testing one. We need to get test.dataset drawn on top of the video for this. I can do that ezpz.

**4.4.2 Known Failures**

Which frames or people are we missing? Do visual inspection vs test.dataset to see if we can spot anyone who stands out. All those tiny people in the background probably don't count.

**5. References**

[1]"xkcd: Electoral Precedent", *Xkcd.com*, 2016. [Online]. Available: <https://xkcd.com/1122/>

[2]"Convert image to double precision - MATLAB im2double", *Uk.mathworks.com*, 2016. [Online]. Available: <http://uk.mathworks.com/help/matlab/ref/im2double.html>

[3]"Principal Component Analysis 4 Dummies: Eigenvectors, Eigenvalues and Dimension Reduction", *George Dallas*, 2013. [Online]. Available: <https://georgemdallas.wordpress.com/2013/10/30/principal-component-analysis-4-dummies-eigenvectors-eigenvalues-and-dimension-reduction/>

[1] <http://uk.mathworks.com/help/stats/fitcsvm.html?refresh=true>

[2] <http://uk.mathworks.com/help/stats/compactclassificationdiscriminant.predict.html>

[3] Hog Feature Extraction, CSC4067 Practical 4

[4] Non-Maxima Suppression, CSC4067 Practical 4

[5] <https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients>

[6] <https://en.wikipedia.org/wiki/Support_vector_machine>

[7] <http://uk.mathworks.com/help/stats/perfcurve.html>

[8] <http://thebrainiac1.blogspot.co.uk/2012/07/v-behaviorurldefaultvmlo.html>