# CSC3067 Group Project

Ross Brown &

Rory Flynn - 40259039

GitHub Repo: https://github.com/almatrass/video-analytics

Before we started on the application, a learning method and a feature descriptor had to be chosen. Since we weren’t sure which was most appropriate, we decided to implement and test several options. In this section we cover several of the methods tried before we decided on our final model.

## Pre-processing

Pre-processing is an essential first step before we start using any feature descriptors or implementing any learning methods. The specific pre-processing techniques that are used can have a significant impact on the quality and accuracy of the feature descriptor, which will be used after this initial step.

One pre-processing technique for image processing is noise reduction. Images are often full of various types of noise, such as salt-and-pepper noise. This noise can make it difficult for image processing algorithms to accurately detect or analyse the features of interest in the image. By reducing the noise in the image, it is possible to improve the performance of the feature descriptor.

Another important technique is contrast enhancement, which is a technique used to improve the visual quality of an image by increasing the difference between the light and dark areas of the image. This is done by stretching out the image, so that the full range of intensity values is used. To do this, we could use histogram equalisation, which attempts to evenly distribute the intensity values across the entire range of the histogram.

Finally, we can also use brightness enhancement, which improves the quality of an image by adjusting the overall brightness. This is done by applying a linear transfer function to the intensity values in the image, in order to stretch or compress the range used by the image.

## Feature descriptors

### HOG

HOG is well known in the video analytics world as one of the best methods for identifying objects in images. This is because HOG can accurately detect the shape of an object (in this case someone’s face), even if the person is not looking directly into the camera. HOG is also very good at dealing with changes in lighting and scale, which is perfect for face detection on our images.

For HOG testing, a histogram is created of pixel orientations, compared with something like full image descriptor, where the image pixel array gets passed directly to the learning method. The histogram can then be used to attempt to detect objects in images, such as faces. Below is a screenshot of an example HOG generated from one of the test images:

Graphical user interface, application

Description automatically generated

A collage of a person's face

Description automatically generated

As can be seen the HOG extractor has done a good job at picking out the faces and lines separating the faces.

### Full image

Passing the full image to the learning method is another option we could use. Although this was attempted, this was not as effective as other methods. This descriptor when used with a classifier we found often struggled to distinguish the foreground and background of an image and is very sensitive to variance. This made it very hard for the learning method to differentiate between the face and the background wall of the images.

Below is an output from an early detector attempt using SVM with full image. This model when paired with various pre-processing techniques (Contrast, brightness) and the SVM classifier produced similar, promising accuracies of around ~80% after testing, with a precision of 75% and recall of 88%. Although these statistics seem fine, we can tell from the precision that the model is prone to a significant number of false positives which we can see in the boxed image below.

A group of people wearing helmets

Description automatically generated with low confidence

A better approach would be to use method such as HOG to detect the faces before passing it to the learning method. This would allow the learning method to focus on the faces themselves, and not have to worry about the background. This would be more effective and would be able to more accurately identify the faces.

### Gabor filtering

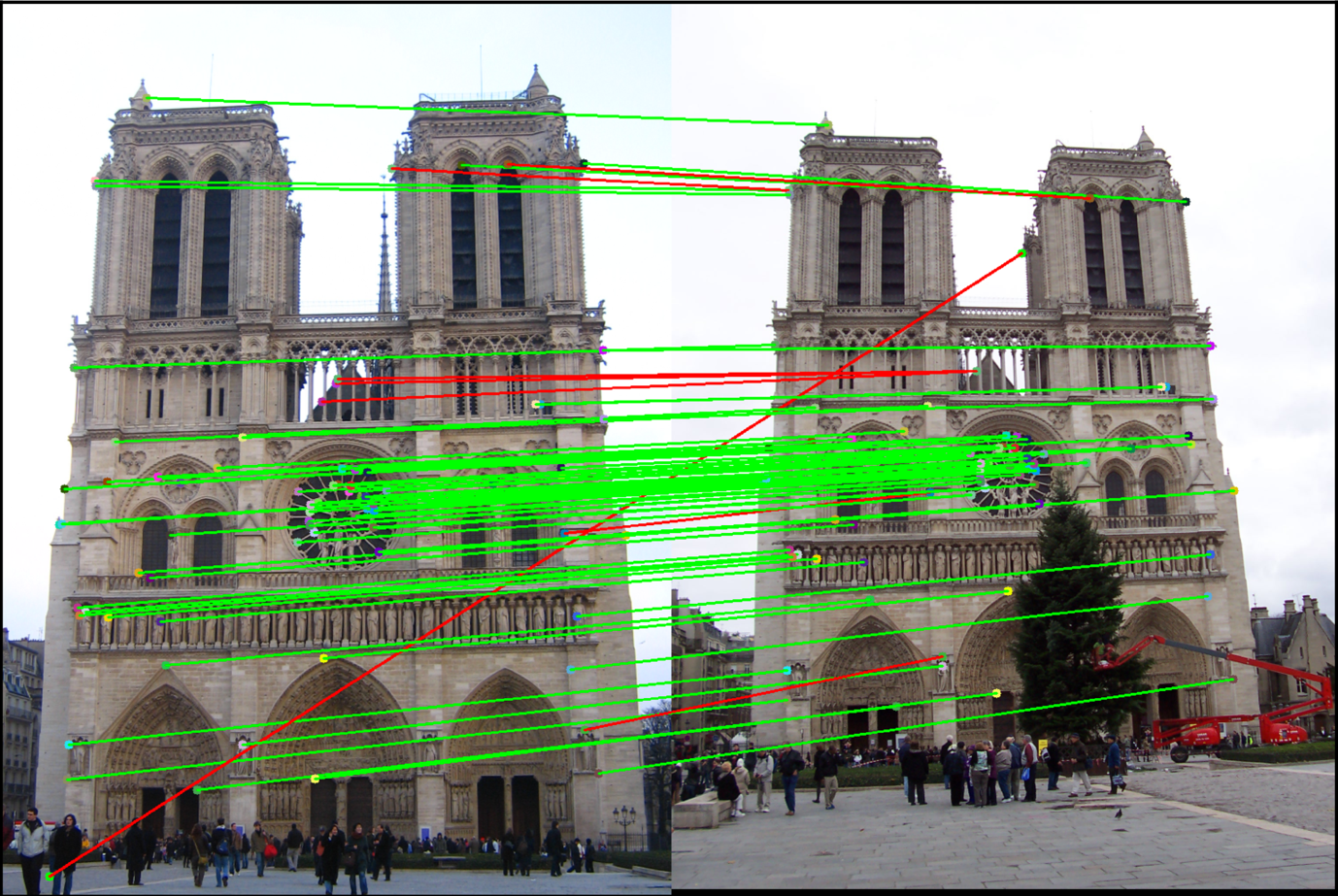
Using Gabor filtering is another good way of implementing a feature descriptor for extracting information from an image before used by the classifier, it works in a similar way to HOG.

Gabor filters are mainly used for edge detection, which should help us in detecting the edges of faces. They do this by measuring the frequency and orientation of the image and are very good at extracting specific objects from images. They’re also very widely used in facial recognition systems, so this method would be perfect for our use case.

Gabor was the chosen feature extractor for our final model.

### MSER, SIFT, SURF and BRISK

We also briefly used MSER, SIFT, SURF and BRISK feature descriptors, which attempt to highlight key areas of images. Using these methods, they can highlight points in the image which are areas of interest or can attempt to match parts of the image. For example, with SIFT, it works by matching up features in images. This is a good example visually demonstrating what the process looks like:



Ultimately, while these methods showed promise some of them appeared incompatible with our dataset, namely BRISK and Surf with the others not yielding any improvement over the likes of Hog and Gabor. If we were to make improvements to the system in future, these descriptors could be reconsidered.

## Classifiers

### SVM

SVM is a supervised learning algorithm that we can use to accurately detect faces in the test images. SVM is ideal for face detection because it can learn complex patterns and features in the images. This is important for detecting faces in the images, especially since they have different lighting, different poses, and other factors that can affect the accuracy.

SVM can learn complex patterns and features in images, which can help the app better differentiate between face and non-face objects and improve accuracy. This is obviously important for face detection, where the app needs to be able to accurately identify faces in a wide range of conditions and scenarios.

Overall, using SVM in a face detection app can provide several benefits, including improved accuracy, better performance, and the ability to handle large and complex datasets should we want to expand the application in the future. This can help ensure that the app provides reliable and accurate face detection and can be more beneficial than using nearest-neighbour algorithms.

### KNN

KNN is a lazy learning algorithm that we can use to accurately detect faces in the test images. Lazy learning means it does not make any assumptions about the underlying data and only performs computation when it is needed. This makes KNN ideal for face detection, where the data distribution can vary widely and the app needs to be able to adapt to changing conditions, such as lighting etc.

Compared to other methods, such as SVM, KNN has several advantages for use in our application. KNN is a lot faster and more efficient, which can help speed up the detection process.

KNN is also highly adaptable, which means it can handle a wide range of changing conditions. This is important for face detection, where the application needs to be able to accurately detect faces in a variety of scenarios and conditions.

Basically, using KNN in our application can provide several benefits, including improved performance, flexibility and is still very accurate. This can help ensure that the application provides accurate face detection and can be as beneficial as using SVM.

## Testing

To get definitive figures for each of the feature descriptors and learning methods, we had to run tests on each so we could compare accuracies using different pre-processing techniques and different K values/kernels and make a more informed decision on what to use. Initially tests were performed using randomised sets with a 70:30 split in favour of training. However, all models below received updates to implement K fold cross validation. The results of these tests are below for each of the setups tested.

### SVM with HOG

SVM with HOG produced some interesting results. It was the fastest learning method we tested using every kernel and had good accuracy.

|  |  |  |  |
| --- | --- | --- | --- |
| Pre-processing | Kernel | Accuracy | Time (seconds) |
| None | Linear | 77% | 6 |
| None | Gaussian | 80% | 6 |
| Brightness | Gaussian | 83% | 7 |
| Contrast | Gaussian | 87% | 7 |

|  |  |
| --- | --- |
| True Positives | 1.00 |
| False Negatives | 0.00 |
| False Positives | 0.40 |
| True Negatives | 0.60 |
| Precision | 71% |
| Recall | 100% |
| Sensitivity | 93% |
| Specificity | 78% |

### SVM with SIFT

SVM with Sift did not perform as well as with HOG. The execution time was slower, and the accuracy values were worse for every pre-processing value/kernel, except for linear kernel. For this reason, we decided to disregard using sift.

|  |  |  |  |
| --- | --- | --- | --- |
| Pre-processing | Kernel | Accuracy | Time (seconds) |
| None | Linear | 78% | 11 |
| None | Gaussian | 79% | 9 |
| Brightness | Gaussian | 82% | 10 |
| Contrast | Gaussian | 83% | 11 |

|  |  |
| --- | --- |
| True Positives | 0.83 |
| False Negatives | 0.17 |
| False Positives | 0.28 |
| True Negatives | 0.72 |
| Precision | 79% |
| Recall | 83% |
| Sensitivity | 91% |
| Specificity | 75% |

A picture containing text, group, posing, line

Description automatically generatedA picture containing group, keyboard

Description automatically generatedA picture containing text

Description automatically generated

A precision score of 79% and a recall score of 83% for this model indicates that the model can correctly identify the presence of an object in an image 79% of the time, but it also means that 21% of the time the model will incorrectly identify the presence of an object. Similarly, the recall score of 83% means that the model can identify most of the instances of an object in the image, but 17% of the time it will fail to identify the presence of the object. This combination of scores suggests that the model is likely to produce a significant number of false positives, where it causes the detector to incorrectly identify the presence of an object that is not actually there. This leads to the inaccurate boxes drawn in the images above.

### SVM with MSER

SVM with MSER was an improvement over SIFT, with better accuracy values. The accuracy values performed better than HOG in all the tests except gaussian kernel with contrast enhancement, however it was slower to run the tests.

|  |  |  |  |
| --- | --- | --- | --- |
| Pre-processing | Kernel | Accuracy | Time (seconds) |
| None | Linear | 83% | 12 |
| None | Gaussian | 83% | 12 |
| Brightness | Gaussian | 87% | 13 |
| Contrast | Gaussian | 86% | 11 |

|  |  |
| --- | --- |
| True Positives | 0.86 |
| False Negatives | 0.14 |
| False Positives | 0.18 |
| True Negatives | 0.82 |
| Precision | 83% |
| Recall | 86% |
| Sensitivity | 93% |
| Specificity | 80% |

A picture containing text, posing, group, team

Description automatically generated

A picture containing text

Description automatically generatedA picture containing indoor

Description automatically generated

Mser has a measurable improvement over Sift with a precision score of 83% and a recall score of 86% the model can accurately identify relevant images 83% of the time, but it also indicates that the model is still producing a significant number of false positives. This is because a high recall score means that the model is successfully identifying a large proportion of relevant images, but the lower precision score suggests that it is also incorrectly labelling a significant number of non-face images as face. As a result, the model is producing many false positives, which can lead to inaccurate results we can see above.

## KNN With HOG

KNN with HOG allowed us to get a faster result when compared with SVM, however the drawback to this is it is not as accurate. We tested using different K-values, which produced different degrees of accuracy. Ultimately we decided KNN would not be as appropriate as SVM, since the accuracy is beneficial in our case and for our simple application the time doesn’t really matter.

|  |  |  |  |
| --- | --- | --- | --- |
| Pre-processing | K-Value | Accuracy | Time (seconds) |
| None | 1 | 72% | 4 |
| None | 3 | 75% | 3 |
| Brightness | 5 | 78% | 5 |
| Contrast | 7 | 82% | 6 |

|  |  |
| --- | --- |
| True Positives | 1.00 |
| False Negatives | 0.00 |
| False Positives | 0.50 |
| True Negatives | 0.50 |
| Precision | 69% |
| Recall | 100% |
| Sensitivity | 90% |
| Specificity | 75% |

A picture containing text, group, posing, line

Description automatically generated

## Final selection – SVM with Gabor

The implementation of our classification system was done using support vector machine (SVM) with Gabor feature extraction. The system is designed to classify images into either face or non-face categories. To improve the performance of the system, a range of pre-processing techniques will be applied to the images prior to classification. Additionally, K fold cross validation will be used to evaluate the performance of the system.

### Pre-processing

One of the most used pre-processing techniques is image smoothing. This technique is used to reduce the level of noise present in the image. Noise can be caused by various factors such as low-light conditions or image capturing devices. Image smoothing can be performed using various methods such as median filtering, Gaussian filtering, or bilateral filtering.

Another pre-processing technique is image sharpening. This technique is used to increase the contrast and edges of the image, making it more defined and clearer. Image sharpening can be performed using various methods such as unsharp masking or high-pass filtering.

Image normalization is another pre-processing technique that is used to make the image have a consistent intensity range across all pixels. This is important because different image capturing devices may produce images with different intensity ranges, which can affect the accuracy of further processing or analysis. Image normalization can be performed using various methods such as histogram equalization or contrast stretching.

All these techniques were used in various combinations when attempting to optimise the classification model, we can see their affect in the tables in the section above. They all have their range of advantages and disadvantages which were considered during this optimisation process but in practice we found that increasing the contrast by itself yielded the best results.

Below shows the standard image, with the same image after pre-processing.

A collage of a person's face

Description automatically generated

A group of people's faces

Description automatically generated with low confidence

### Feature extraction

Gabor feature extraction is a method used in image processing and computer vision to extract features from images. This method is based on the Gabor filter, which is a mathematical function that is used to extract spatial and spectral information from an image.

Gabor feature extraction involves applying the Gabor filter to an image to extract features such as texture, shape, and orientation. The filter is applied in multiple orientations and scales, allowing for the extraction of a wide range of features.

One advantage of using Gabor feature extraction is that it is rotation invariant, meaning that the extracted features will remain consistent even if the image is rotated. This makes it useful for applications such as object recognition, where the orientation of the object may vary.

Another advantage is that Gabor feature extraction is computationally efficient. The Gabor filter is a simple mathematical function and applying it to an image is relatively fast. This makes it suitable for use in real-time applications, such as surveillance and security systems.

There are also some limitations to using Gabor feature extraction. For example, the extracted features may not be as robust as those extracted using other methods, such as SIFT (Scale-Invariant Feature Transform) however, for the purposes of this system the accuracy increase from using Gabor over another technique such as SIFT, or HOG was a worthwhile trade off.

Additionally, the Gabor filter is not translation invariant, meaning that the extracted features may be affected by the position of the object in the image. Below is a visualised example of the gabor filter.

Chart, surface chart

Description automatically generated

### Classification

Once the Gabor filters have been applied to the images, the resulting features will be used to train a support vector machine (SVM) for classification. The SVM is a type of machine learning algorithm that is particularly well suited to binary classification tasks such as this one. It works by finding a line (or hyperplane) that maximally separates the two classes of data, in this case face and non-face.

The SVM will be trained using a range of parameters, including the kernel type, regularization constant, and kernel width. These parameters will be optimized using a grid search to find the combination that produces the best performance.

Another advantage of SVM for image classification is its ability to handle complex, non-linear data. In many cases, the objects in an image can have complex shapes and patterns, which can make it difficult for other machine learning algorithms to accurately classify them. SVM, however, can learn complex patterns and classify images accurately, even when the data is non-linear.

In terms of performance, SVM is known to have a high accuracy rate compared to other classification algorithms. It is also robust to overfitting, which makes it a good choice for this system. Although, a limitation of SVM is that it can be computationally intensive, especially when dealing with large datasets. It is also sensitive to the choice of kernel and hyperparameters, which require careful tuning to achieve optimal performance.

## Evaluation using K fold cross validation

To evaluate the performance of the classification system, K fold cross validation was used. This is a method of dividing the data into K folds and training the model on K-1 folds while testing it on the remaining fold. This process is repeated K times, using a different fold as the test set each time. The final performance of the model is the average performance across all K folds.

## Results

### Training Data

|  |  |  |  |
| --- | --- | --- | --- |
| Pre-processing | Kernel | Accuracy | Time (seconds) |
| None | Linear | 84% | 12 |
| None | Gaussian | 86% | 13 |
| Brightness | Gaussian | 87% | 13 |
| Contrast | Gaussian | 93% | 15 |

### Classifier Performance

|  |  |
| --- | --- |
| True Positives | 0.98 |
| False Negatives | 0.02 |
| False Positives | 0.12 |
| True Negatives | 0.88 |
| Precision | 89% |
| Recall | 98% |
| Sensitivity | 85% |
| Specificity | 79% |

### ROC Curve

Chart

Description automatically generated

Based on the performance we can see the image recognition model has a very high precision and recall. This means that it can correctly identify a large proportion of the images it sees, and it is also able to correctly reject a large proportion of the non-target images.

The model has a relatively low sensitivity, which indicates that it sometimes fails to identify target images. This may be due to a variety of factors, such as the difficulty of the images it is being asked to recognize or limitations in the model itself.

The model also has a relatively low specificity, which indicates that it sometimes incorrectly identifies non-target images as being part of the target class. This is also known as a "false positive" error.

Overall, these values suggest that the image recognition model is quite accurate, but there is room for improvement in its ability to correctly identify all target images and reject non-target images.

# Detection

The Gabor feature extraction method was used in combination with a SVM classifier for the face detection system. This combination was chosen for its high accuracy, which minimizes the number of false positives and negatives. The sliding window approach requires analysing many samples per frame, so even a small difference in accuracy can greatly impact the performance of the system.

The implementation of the detector has several adjustable settings. The first is the number and size of windows used during detection. To implement a multi-scale sliding window, the detector scans each frame multiple times, using a different window size each time. This allows the detector to obtain samples of faces at different scales relative to the training images, even at different distances from the camera. The settings used for the detector were 4 sample sizes with dimensions of [27,18], [18,27], [36,24] and [24,36] . Once the samples were taken, they were all resized to [27, 18] to match the training images.

Another adjustable setting is the step increment, which determines the number of pixels between each sample taken by the sliding window. A larger step increment allows for fewer samples to be taken without significantly impacting accuracy, reducing the time needed for detection. A step increment of 10 was used for the detector.

Every sample taken by the sliding window is passed into the SVM classifier, which returns a prediction and the confidence of the prediction. If a face is detected, the details of the sample location and the confidence of the prediction are stored. After a frame has been scanned by each sample size, non-maxima suppression is applied to reduce the detections to only those with the highest confidence where boxes overlap. This allows for more accurate predictions of each face.

The method used for non-maxima suppression was simple: if the overlap between two boxes is greater than a given threshold, the box with the lower confidence score is removed. For the detector, a threshold of 25% overlap was used.

The detector's performance for some images can be seen below along with the number of faces detected.

A picture containing indoor

Description automatically generatedA picture containing text

Description automatically generated

A picture containing group, line

Description automatically generated

A picture containing text, group, posing, line

Description automatically generated

|  |  |  |
| --- | --- | --- |
| Image Number | Faces Detected | Actual Number of Faces |
| 1 | 9 | 7 |
| 2 | 15 | 15 |
| 3 | 20 | 8 |
| 4 | 76 | 57 |

As can be seen from these results, the detector generally detects more potential positions in the frame where a face could be than the test dataset provides. To see why, we will look at the second image as an example.

A picture containing group, line

Description automatically generatedAs we can see the system has classified various parts of people’s bodies as a face, such as necks, arms etc.

This image shows that the detector generally produces more boxes than the test dataset. Firstly, there are some false positives produced by the detector, for example in some of subject’s necks are categorised as a face. It is possible that this may simply be an error on the part of the classifier, but it may also have been caused by the training data. For example, see these images taken from the training data set:

As these are training images used as positive examples of faces, it is possible that the partial view of the subject’s neck is confusing the classifier. Although, the more likely cause is the size of the dataset used to train the model.

The other reason for the higher number of boxes produced by the detector can be seen by looking at the woman on the right side of the image. The detector has placed two separate boxes to cover the face. This is likely caused by the sizes chosen for the scaling window. The largest sample size is not quite able to completely cover the face, so two smaller windows are used instead. Adding additional larger sample sizes may help to identify faces that are close to the camera but will cause a significant increase in time taken by the detector to process each frame and may cause other boxes to be removed by the non-maxima suppression if the overlap is too great.

## Final Comments

In conclusion, the implementation of a classification system using SVM with Gabor feature extraction is a promising approach to the task of classifying images into face and non-face categories. The use of pre-processing techniques, Gabor filters, and SVM provides a robust and effective method for extracting features from the images and making accurate classifications.