Content Based Image Retrieval & Categorisation

B.Sc. Information Technology (Artificial Intelligence)

247696M

Edward Fleri Soler

Table of Contents

Introduction

Content based image retrieval and categorisation is an automated approach to image analysis and classification. This involves extracting information from digital images, for the purpose of comparison and identification of the subject(s) captured in an image. This has a wide variety of applications in modern industries, ranging from detection systems; implemented in security systems, smart phones and autonomous cars, to image searching and classification; retrieving images of similar subject matter or meaning – recently introduced by Google.

In this project, we shall design, develop and test an image classification system, with the purpose of classifying sample images of different scenes. This system shall adopt a wide variety of techniques commonly used in computer vision and we shall understand and discuss the functionality and importance of these approaches to image processing.

This documentation shall commence by firstly considering background research, to ensure a thorough understanding of the topics tackled in this assignment. We shall then outline the proposed system, and discuss the implementation of the system itself. A detailed evaluation of the systems components, together with the expected outcome shall be made, before observing the actual results achieved. Finally, a discussion will take place, evaluating the success of the system together with possible pitfalls and future improvements, before bringing the project to a conclusion.

Background Research

The majority of image categorisation systems follow the same three-part approach: low-level feature extraction, high-level feature representation and classification. At each stage, different techniques for information extraction and handling may be applied, each with their own set of specifications and benefits. In this section, we shall outline the foundations of each of the three steps, providing the background knowledge necessary for a comprehensive understanding of the remainder of the project.

Keypoints

In computer vision, low-level feature extraction is the generation of information describing an image by means of its basic features. These features would not make much sense, and are not very useful to us in their raw form. However, when used in conjunction with the appropriate algorithms, these features can be used to describe and distinguish between similar and dissimilar images.

Low-level features are also known as keypoints. A keypoint is a cluster of pixels which provides information useful for the identification of an image. One keypoint alone provides little confidence for the distinguishing of images; however, a combination of these feature points are capable of describing the salient distinguishing features in an image. By extracting the keypoints of different images, we may compare these images, computing a similarity rating. This has a wide variety of applications in industry, from object identification and optical character recognition (OCR), to scene classification, which is what we shall be tackling in this project.

An ideal keypoint is one which is present when capturing images of an object from different angles, perspectives and conditions. This has brought about the intensive research and development of multiple keypoint detection algorithms, each basing the extraction of information on different image features, such as gradient, pixel intensity and object corners. Given the specifications of this project, the Scale Invariant Feature Transform (SIFT) algorithm is the keypoint detector of choice.

SIFT

The SIFT algorithm was chosen for the detection and description of image features due to its success and common use in literature. The major benefits of the SIFT algorithm over other keypoint detection techniques is its robustness to orientation and scaling. This means that, by taking a large number of keypoints from a large number of training images, a new test image may be compared to the training images even when minimal change in orientation and scale are present. The SIFT algorithm operates in a number of stages dedicated to identifying salient pixel groups in the image, incorporating scale in the keypoint selection process, defining the prominent orientation of each keypoint and describing the keypoint in a linear fashion.

Firstly, a Difference-of-Gaussian (DoG) function is applied. Gaussian filters have the effect of blurring images. The Difference-of-Gaussian function proceeds by applying Gaussian filters of varying intensities to the same image, subtracting the pixel values from each different image. With X Gaussian filters of varying intensity applied to an image, we yield X-1 DoG images, as seen in figure 1 below.

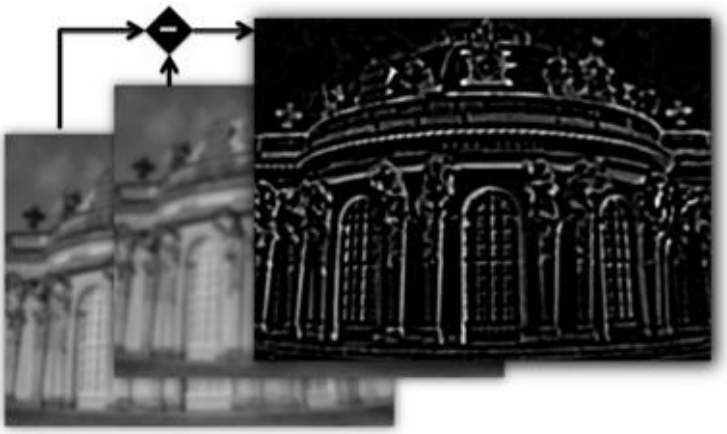

FIGURE 1 DIFFERENCE OF GAUSSIAN (SOURCE: HTTPS://WWW.EECS.TU-BERLIN.DE/)

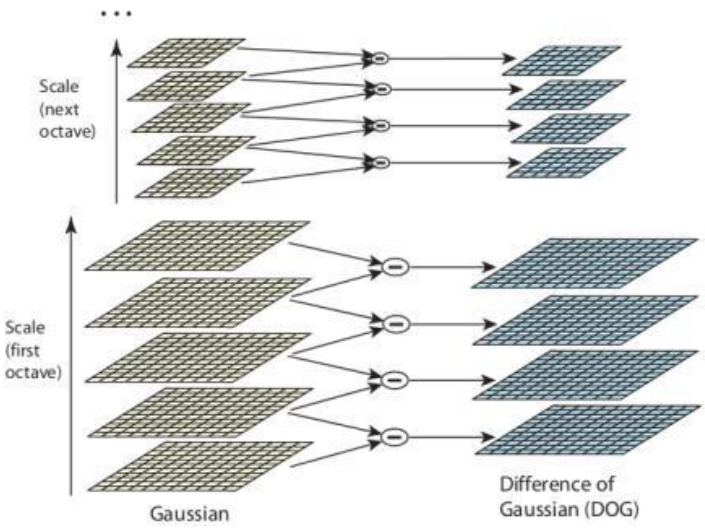

FIGURE 2 SCALE SPACE CONSTRUCTION (SOURCE: OPENCV)

This process is repeated multiple times, rescaling the images at each iteration to form a scale space, as seen in figure 2 above. Each DoG within the scale space is known as an octave, and the number of octaves as well as the variation in Gaussian filter intensity are parameters which may be tailored for different applications of the algorithm. Finally, each DoG pile is analysed for local extrema, defined as pixels with the greatest or smallest pixel intensity when compared to neighbouring pixels in the same DoG image and neighbouring images in the DoG stack. These extrema points mark the locations for keypoints.

The algorithm now begins describing the selected extrema by considering a square grid of 8x8 pixels centred around it. The gradient orientation for each pixel within the square grid is computed. The image gradient is the direction in which a change in colour or intensity occurs within an image. The pixel gradient is therefore computed by considering the intensity of the current pixel, comparing it to neighbouring pixels to deduce an angle and magnitude of change. The square grid is then considered in 16 sections (4x4). For each section, a histogram comprising of bins ranging from 0 to 2π is created, with each bin representing the total magnitude of all pixels falling within that gradient angle, in the respective section. Each histogram is finally normalised to reduce the effects of different lighting conditions. This results in 16 histograms with 8 bins each (π/4 degree intervals) for a final 128-dimension descriptor for each keypoint. This multi-dimensional descriptor is therefore capable of uniquely describing a keypoint within an image.

The SIFT algorithm is acclaimed as being robust to changes in orientation. This is possible by considering a neighbourhood around each keypoint, with the dominant orientation being computed by considering the gradient angle of each pixel. This dominant orientation is then assigned to each keypoint, allowing similar but rotated keypoints to be distinguished and matched.

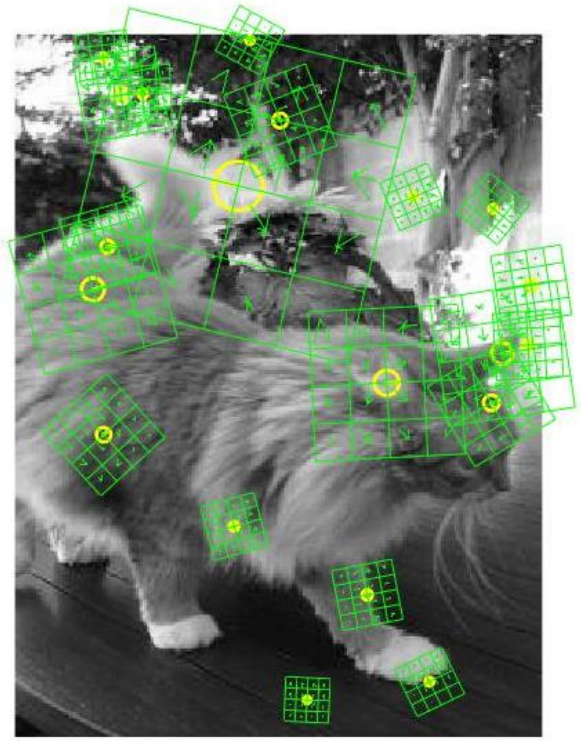

FIGURE 3 GRAPHICAL DEPICTION OF KEYPOINTS TOGETHER WITH THE HISTOGRAM ORIENTATION OF NEIGHBOURING PIXELS

Figure 3 provides a graphical representation of keypoints extracted from a sample image. Each keypoint is represented by a 4x4 grid representing the 16 sections which each 8x8 starting grid is split into. Each

block in the 4x4 section contains vectors depicting the histogram angles and magnitudes, with the size and orientation of each 4x4 grid depicting the scale and overall dominant orientation respectively. Typically, the SIFT algorithm splits input images into windows, extracting a set number of keypoints from each window. The number of keypoints extracted for each image, as well as the space between each keypoint is another parameter of the SIFT algorithm which allows for tailoring to specific needs.

Bag-of-Words

As mentioned above, the image features in their low-level representation as keypoints are hard to handle and are of little use by themselves. A higher-level abstraction of the low-level data is required to make it easier to train and compare images based upon image features. The Bag-of-Words model is a data representation commonly used for the simplification of information extracted from training. The idea behind the Bag-of-Words approach is the ability to describe images through a small set of features, rather than the excessive number of possible keypoints. This generalisation also helps reduce overfitting, a problem which occurs when a training set is learnt too perfectly and does not allow room for slight variation.

When constructing a bag of visual words, we start by extracting the keypoints for each image within the available training set. These keypoints are then mapped out onto a multi-dimensional search space. The K-means clustering algorithm (or any other clustering algorithm) is then applied to group close keypoints into K different clusters. With each iteration, the K-means algorithm improves upon the cluster allocation of each keypoint. When finished, we have K final clusters, each with their own central position within the search space, known as a centroid. Each centroid is now considered to be a word in a bag of K words forming the vocabulary or codebook.

Once the codebook has been generated, keypoints of training images are considered within the same search space as before. A classification algorithm is then used to match each keypoint up with its closest word within the search space. The result, is a histogram of words describing each training image. This histogram contains K bins, equal to the number of words within the codebook, with each bin holding the frequency of word-hits, with each word-hit relating to a keypoint being matched with a specific word. The same process is followed for the training set, producing a descriptor for each image based upon these words. This descriptor is then compared to the training set and undergoes final classification.

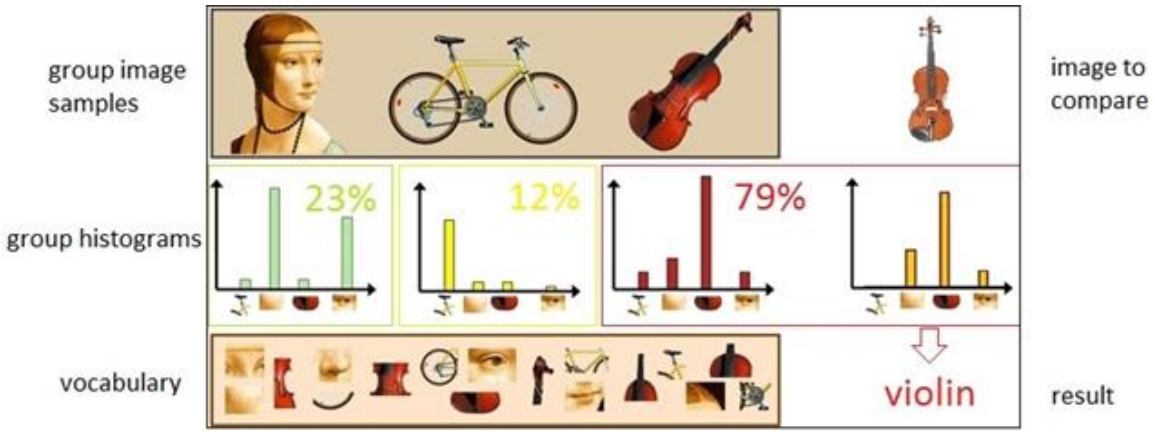

FIGURE 4 EXAMPLE OF BAG OF VISUAL WORDS APPLIED FOR THE CLASSIFICATION OF A TEST IMAGE (SOURCE: VISION & GRAPHICS GROUP)

Classification

The final stage in all image categorisation systems is classification. Classification involves comparing data from an unlabelled test image to data from a database of labelled training images and determining a suitable label. Amongst the various classification algorithms, we shall consider two of the most popular: K-Nearest Neighbour and Support Vector Machines.

K-Nearest Neighbour

K-Nearest Neighbour (KNN) is a classification algorithm based upon Euclidean distance. This algorithm works by mapping out all labelled data onto a search space based upon their feature values. An unseen, unlabelled test input requiring classification is mapped onto this same search space based upon its own feature values. The algorithm then computes the distance from the test input to all other points in the search space. The K points closest to the test point are now considered, with the labels of these points determining the final classification, through majority rule.

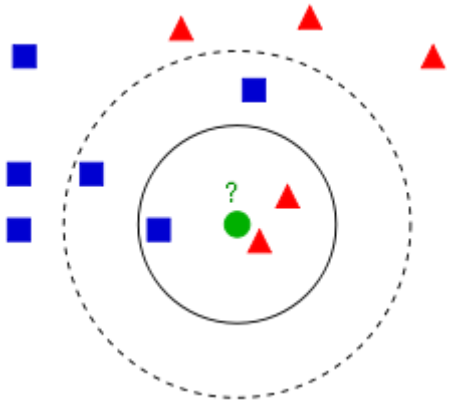

FIGURE 5 K-NEAREST NEIGHBOUR (SOURCE: WIKIPEDIA)

In the case of a tie, the value of K could be incremented or decremented so as to make a conclusive decision. The value of K is a parameter of the algorithm, which could be optimised for any given application through validation tests.

Support Vector Machine

Support Vector Machines (SVMs) are another classification algorithm based upon vector calculus. SVMs are binary classifiers, and can therefore only handle two-class problems. For this reason, multi-class problems are split into two class problems, with one SVM trained for each of the possible classes. The motivation behind SVMs is finding the best possible hyper-plane which divides the two sets of labelled training classes. This ‘perfect’ hyperplane holds the largest distance from any point in the search space on either side. Test values are then mapped onto the search space and assigned the appropriate classification based upon which side of the hyperplane they fall on.

Kernelling is another feature of SVMs which helps define the optimal hyperplane. This involves transforming and mapping all data points onto different dimensions by means of a function. The hope is that this transformation will yield a hyperplane with a larger margin between points on either side, while leaving the data itself untouched. This feature, together with the approach in general makes this classification algorithm one of the most accurate for a wide variety of applications.

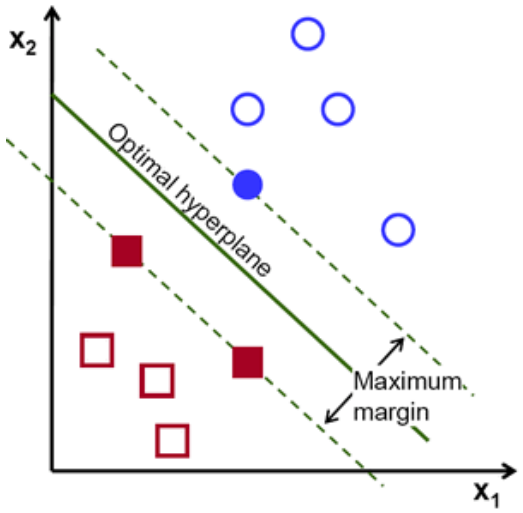

FIGURE 6 DEPICTION OF SUPPORT VECTOR MACHINE

Problem Definition

Aims & Objectives

The aim of this project is to design, build and evaluate an image classification system implementing algorithms and techniques common within the application of content based image retrieval. This system shall focus on the classification of images of different scenes, ranging from indoor environments, to urban and rural landscapes. We shall strive to develop and fine tune the system to achieve a satisfactory accuracy of classification.

Dataset

For the purpose of training, testing and evaluation, a dataset of over 3,000 images has been made available. This data has been readily split into training and testing sets, with each set further split into 14 different categories representing the image scenes. The following is a complete list of scenes:

- Bedroom
- Coast
- Forest
- Highway
- Industrial
- Inside City
- Kitchen
- Living Room
- Mountain
- Office
- Open Country
- Store
- Street
- Suburb
- Tall Building

FIGURE 7 SAMPLE IMAGES OF KITCHEN, INDUSTRIAL, MOUNTAIN & TALL BUILDING SCENES

Method

The programming language of choice for this system was MATLAB, due to its ease of handling multi-dimensional variables, its multiple in-built functions and the wide variety of computer vision libraries available. A large number of discussion forums and support for computer vision applications implemented through MATLAB are also available. MATLAB skeleton code defining the rough layout and work flow of the system was provided, with a list of recommended functions together with their arguments and return variables. The VLFeat open source MATLAB library (available at www.vlfeat.org) was imported and used in this implementation.

The system was designed to follow a flexible structure, allowing the user to determine different feature extraction and classification techniques, so as to explore the characteristics and functionality of each approach.

Data Preparation

Before any analysis of data could be performed, the data itself must be prepared and organised into a convenient manner. The first process was the loading of the images from file and the division of images into different sets. The *get_image_paths* function was defined to extract the file paths of all images and perform cross validation.

Cross validation is the splitting of data to form a new set known as the validation set. A validation set is vital in testing and tuning the parameters of a system. The idea is that a system is trained upon the training set, and initially tested and improved upon the validation set, which acts as a temporary test set. Iterative improvements to the system are made, guided by the results achieved from the testing of the validation set. This ensures that the system is not biased to the final test set which, in a real-world setting, would not be labelled. Once the system has been fine-tuned, the validation set is no longer required, and all samples are returned back to the training set.

A decimal value is therefore passed to the *get_image_paths* function, defining the size of the new validation set as a percentage of the current training set. Random samples from each scene category are then extracted from the training set and moved into the validation set. This function then returns 3 vectors holding the paths to each image of the train, validation and test sets, as well as another 3 vectors holding the labels of the images of each set.

Feature Extraction

The next step in this implementation is the extraction of features from the images. Two different approaches to this step were coded so as to understand their differences and functionality.

Tiny Images

The tiny image approach to feature extraction involves resizing images to small pixel dimensions and using the pixel values as feature descriptors. This procedure is defined by the *get_tiny_images* function. In this function, images from the specified data set are loaded and converted to grayscale. The largest possible, centred square of pixels is cropped out for each image, ensuring an equal aspect ratio is maintained when resizing the images. Each image is then resized to be 16x16 pixels in dimension. The intensity value of each pixel is then taken to act as a descriptor for the image, creating a vector of 256 elements in length to represent each image. This function is called to return the descriptors of all training and validation/testing images, producing two 2-dimensional matrices holding all the descriptors.

Bag-of-SIFT

The alternative to the tiny images approach is the Bag-of-SIFT approach. This process of feature extraction follows the same work flow as the standard Bag-of-Words model. First, a vocabulary or codebook must be defined for this model. A simple test is made to see if a codebook from a previous execution is available to be loaded, so as to save resources and time. If a code book is not present, the *build_vocabulary* function is called so as to build a vocabulary of a specified size upon the training data set.

The *build_vocabulary* function begins by once again loading each image from the training set and converting them to grayscale. The SIFT algorithm, implemented through the VLFeat library function *vl_dsift*, is called to extract a set of keypoints for each image. A parameter of this algorithm is the step size taken between each keypoint extracted. The effects of this parameter and its tuning will be discussed in the next section.

The features of all training images are collected and stored in a matrix, which undergoes K-Means clustering. The value of K passed to the K-Means algorithm is equal to the vocabulary size previously defined. Once again, we will discuss the tuning of this parameter in the next section. The algorithm then returns a set of K points, being the centroids of each cluster. These points are the words forming the codebook.

Once the codebook has been computed or loaded, we return to the main function, where we now call the *get_bag_of_sifts* function. This function extracts the features of each image, and is executed upon both the training set and the validation/test set. In this function, each image of the specified set is once again loaded and converted to grayscale, before keypoints are extracted by means of the SIFT algorithm. We now compare each keypoint to the vocabulary present within the codebook, pairing up each keypoint with the closest 'word' in the 'bag'. As explained at the start of this project, a histogram equal to the size of the bag is then created, and populated with the frequency of each 'word-hit'. This histogram is then normalised, before being returned as the new descriptor for the current image.

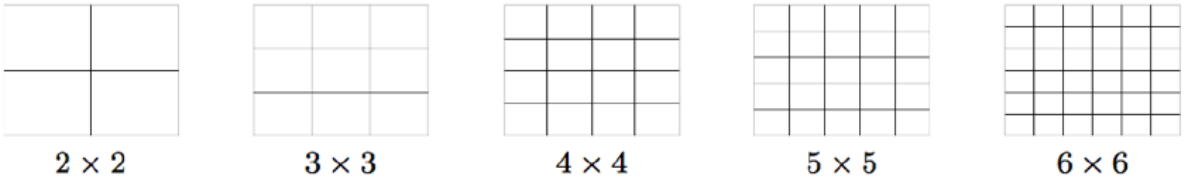

FIGURE 8 EXAMPLES OF SPATIAL TILING LAYOUTS (SOURCE: NOTES BY DR GEORGE AZZOPARDI)

Added functionality has been incorporated into the *get_bag_of_sifts* function by means of spatial tiling. Spatial tiling involves splitting an image into non-overlapping tiles, and repeating the above process within each tile. This produces feature descriptors for each different region of the image, incorporating spatial data into our image descriptors. The result is an image descriptor Y^2 orders bigger when implementing a spatial tile of YxY dimensions. The effects of spatial tiling on the accuracy of classification shall be discussed at a later stage.

Feature Selection

Feature selection is the process of reducing a descriptor to hold the most salient features. The motivation behind this is that some features within the feature set may not hold any fruitful information, but in contrast, prevent the correct classification of test images. Two feature selection algorithms have been incorporated into this system, each designed to select and maintain the most prominent features.

Principal Component Analysis

Principal Component Analysis is a dimensionality reduction method designed to reduce the number of features used to describe images, while retaining a high level of accuracy. This algorithm maps data to a lower-dimension space, maximising the variance. It proceeds by computing the mean value of each pixel in all images of the dataset. This mean image is then subtracted from each original image, to compute the difference image (similar to the DoG procedure). The covariance set is then computed through matrix multiplication of the difference images. This allows us to extract the eigenvectors (each one relating to an image feature), sorting them in order of their corresponding eigenvalue. The K largest eigenvectors required to retain the specified accuracy rates are kept while the others are discarded. The K set of image features to be retained are then extracted once again, forming a new image descriptor of smaller size which still maintains a satisfactory accuracy in its description.

Feature Selection through Genetic Algorithms

Another feature selection algorithm included in the system design is based upon genetic algorithms. This algorithm mimics the process of natural selection by encoding features as chromosomes which undergo the same phenomena as living organisms. At the start, a new set of randomly generated chromosomes is created, with the population having a size equal to the dimensions of each image descriptor. Each chromosome is encoded as a string of 1s and 0s, with a 1 relating to the inclusion of a feature, and a 0 relating to the disregard of a feature. The algorithm iterates through a number of life cycles called epochs. A fitness rating of each chromosome is then computed by means of a fitness function. The fitness function considers all training images to be represented only by the features marked as 1 in the chromosome, while ignoring features marked as 0. The scores produced by these new feature descriptors are plotted, stored and passed on to the next epoch.

With each epoch, chromosomes either die, live on, reproduce or mutate. The fittest chromosomes are set to live on to the next generation, and new offspring are produced by means of cross over from two random parent chromosomes. By performing cross over, offspring inherit some traits from one parents, and others from the other parent, mimicking the real biological process. Offspring may also mutate with a certain probability. This mutation is a random change in the offspring chromosome – not brought about by either of the parents. This, once again, mimics what occurs in the natural world, in the hope that some mutations may be good, helping to create healthy offspring.

This cycle is repeated a number of times, till a stopping condition is met. When well-chosen parameters are set, with each epoch, the overall fitness of the population improves, generating more suitable chromosomes. The end result is the fittest chromosome computed, which will represent the set of features to be kept (1s) and the set to be discarded (0s). Similarly to the codebook generation, in this implementation, before computing a new feature selection chromosome, a test is made to see if a chromosome from a previous execution is available, to save resources.

Classification

Once feature descriptors for all images have been computed, classification of the test data is required. Two possible classification approaches have been defined: K-Nearest Neighbour and Support Vector Machines.

K-Nearest Neighbour

The *nearest_neighbour_classify* algorithm is defined to perform the task of classification by means of the KNN algorithm. For each test image, the distance from its feature descriptor to all training images is computed, and a list of training images is created in ascending order of distance. The K closest neighbours are considered, and a majority rule dictates the prevailing label for the test image.

Support Vector Machine

The *svm_classify* function performs classification of test images through the execution of SVMs. One SVM for each category is trained upon the training set, with all images labelled as positive if they refer to the current category (example SVM for Kitchen) and negative otherwise. Once the SVM is trained, the test data is processed, and a score is returned for each test image, for each category. The function then iterates through all the test images, assigning the classification label relating to the SVM classifier which produced the greatest positive score.

Two parameters for the training of the SVMs are the sigma value and the kernel mapping. These values were experimented with and tuned to achieve the best possible accuracy rating. A discussion of these parameters will be held in the next section.

Visualising the Results

Now that all test images have been assigned a label, all that remains is to compare the assigned label to the true label to gauge the accuracy of the system. The results were visualised by means of a confusion matrix, comparing the true image labels to the predicted ones. A function was readily provided to compute this confusion matrix, and publish a webpage displaying samples of true positive, false positives and false negatives.

Parameter Evaluation

A number of algorithms and methods implemented within the system require parameters set by the user. These parameters effect the success of the procedures in a number of different ways, and must be well selected for the task in hand. The system was executed multiple times with tests being conducted on the validation set so as to gauge the effects of the changes in parameter values. The following are a number of parameters which had large effects on the accuracy ratings:

SIFT – Step Size

One of the parameters for the SIFT keypoint extraction algorithm is the step size. The step size dictates the minimum pixel space between any two keypoints extracted. A small step size would dictate a large number of extractions, taking much longer for keypoints to be extracted for each image. As was mentioned when discussing feature extraction, it is not always beneficial to extract and store large amounts of feature data, as some of this data may be a hindrance when it comes to accurate classification. Following a number of tests with modifications to this value, it was found that a variable step size to form a 30x30 grid for keypoints on each image was the most beneficial, taking a respectable amount of time to extract the features of each image while achieving satisfactory levels of accuracy.

Another decision taken for the sake of resources was to run SIFT in ‘Fast’ mode. This meant that a piecewise-flat window, rather than a Gaussian window, was used during computation, reducing slightly the accuracy of the algorithm but increasing speeds tremendously.

Vocabulary Size

One highly influential parameter is that of the vocabulary, or codebook, size. The fine tuning of this parameter is paramount, as it effects the description of all images in each dataset. The fine tuning of this parameter boils down to the best selection for the value K in the K-Means algorithm used to form the vocabulary. For this calibration, Dunn’s Index was used.

Dunn’s Index is a model for the evaluation of cluster groups. This measure is based upon the inter and intra distance measures. The intra distance measure is the mean Euclidean distance between every

point forming part of a cluster, and the centroid. This is the variance measure of the cluster. The inter distance measure is the Euclidean distance from any cluster to its closest neighbouring cluster. These two values are combined to form a validity measure by dividing the intra distance value, by the inter distance value. Therefore, the best value of K is one which forms tightly knit clusters which are spread far apart from their neighbours.

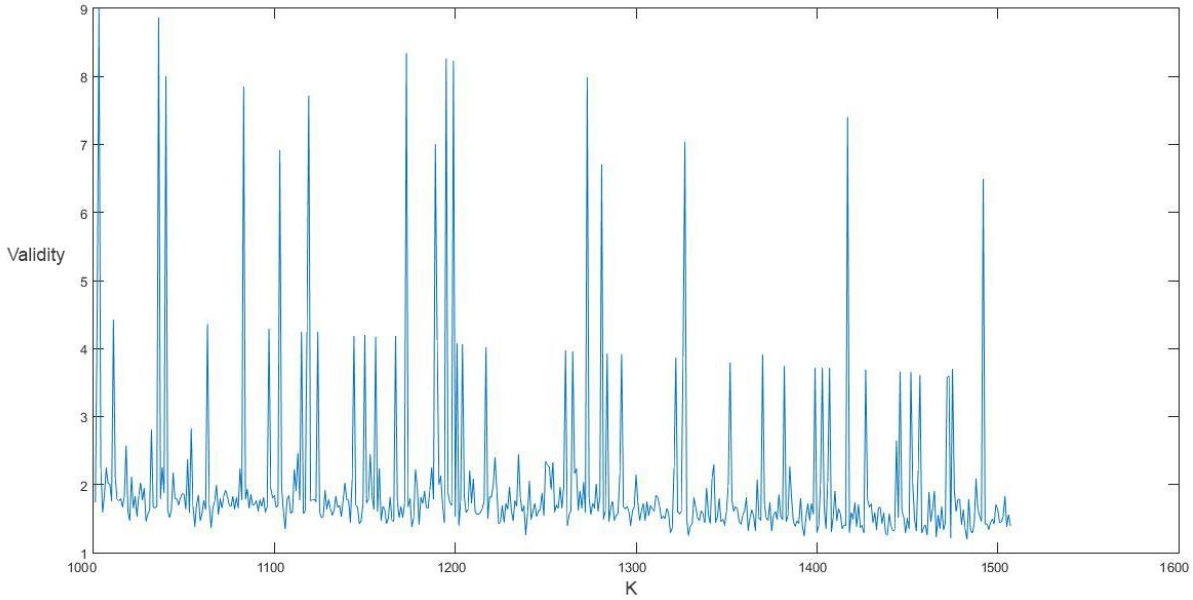

FIGURE 9 VALIDATION PLOT FOR DUNN'S INDEX

A Dunn's Index test was conducted for values of K between 1000 and 1500. The above figure displays the results of the test, with the best value for the vocabulary size being found to be 1482 words.

K-Nearest Neighbour

Similarly to the K-Means algorithm, the value of K for the K-Nearest Neighbour algorithm has a direct effect on the classification accuracy of the test data sets. Given the large number of possible labels assigned to the test image, a higher value of K should be assumed to overcome the case of outlying labels. Following multiple tests with the value of K, it was concluded that the best value was that of 25.

Support Vector Machine – Sigma & Kernel

SVMs are highly flexible and can be easily adapted to achieve the best accuracy rates for any given problem. Two parameters having the greatest effect on the accuracy of the SVM are the kernel mapping and the sigma value. The sigma value regularizes the kernel used to map the data values onto the feature space, while the kernel type defines what mapping function to apply to the data values. Multiple tests were conducted to achieve the best combination of the two parameters to find the best possible hyperplane for SVM classification. The findings of these tests were that the best classification accuracy was achieved when setting sigma to be 0.0001 and the kernel mapping to be a polynomial one.

Spatial Tiling

The dimensions of the spatial tiles applied when extracting feature descriptors plays a big role in the accuracy of classification. Larger dimensions result in a larger feature descriptor for each image, and more processing. Very large dimensions can have the effect of overfitting, restricting keypoints to small spatial domains, only matching with very similar images. For this reasons, tests were conducted with spatial tiling dimensions ranging from 2x2 to 6x6. The results showed that the greatest accuracy was achieved midpoint, with spatial tiling of 4x4.

Results

Following fine tuning of all parameters, the system was switched to test mode, using all training examples to train the system as no validation set was required. A number of tests were conducted to understand the effects that the different approaches to the same problem have on the accuracy of the system. The following is a complete table of results:

	Standard	**Spatial Tiling**	**Principal Component Analysis**	**Feature Selection through Genetic Algorithms**
Tiny Images + KNN	21.1%	-	-	-
Tiny Images + SVM	19.9%	-	-	-
Bag of SIFTS + KNN	50.6%	54.3%	26.0%	54.4%
Bag of SIFTS + SVM	66.2%	71.2%	72.1%	71.0%

TABLE 1 ACCURACY RESULTS FOR DIFFERENT APPROACHES

The above table illustrates the classification accuracy ratings achieved through the different approaches to image categorisation. Seeing that the application of spatial tiling brought about improved accuracy when compared to the standard approach, the Principal Component Analysis and Feature Selection through Genetic Algorithms tests were conducted with spatial tiling of 4x4 dimensions.

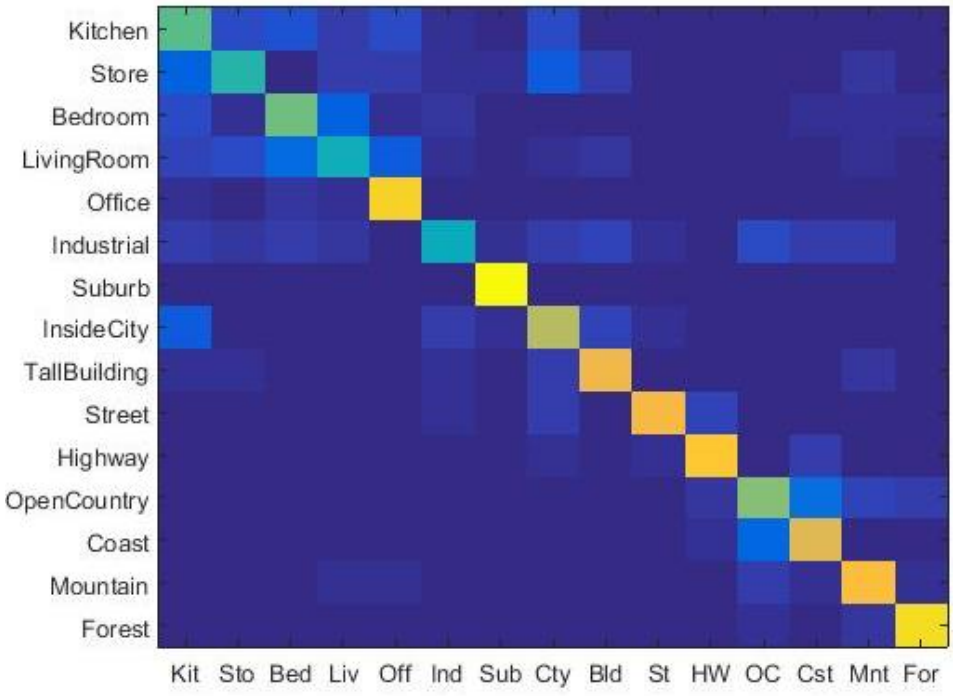

FIGURE 10 CONFUSION MATRIX FOR BAG OF SIFTS + SVM + SPATIAL TILING

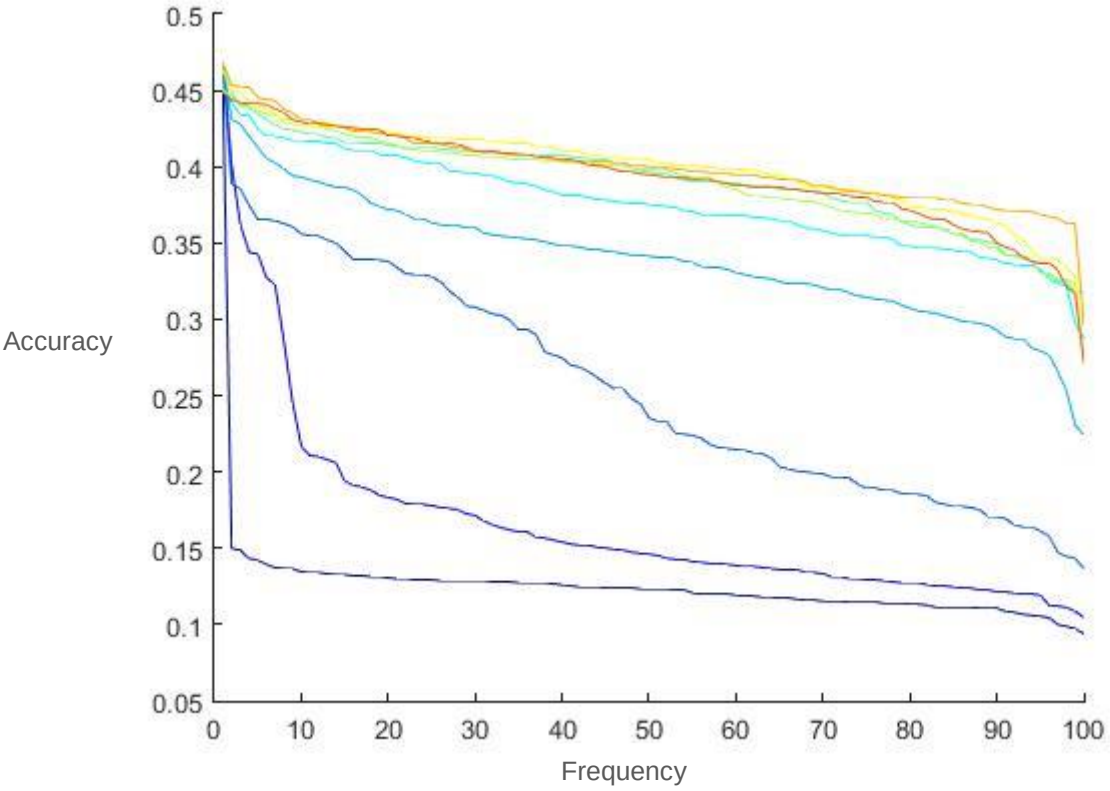

FIGURE 11 DEPICTION OF CHROMOSOME ACCURACY RATES FOR EACH GENERATION WITHIN THE FEATURE SELECTION THROUGH GENETIC ALGORITHMS CYCLE

Discussion

By observing the table of results above, we may better understand the effects of the different approaches to the same problem. Firstly, we must consider that this is a 14 class problem. This means that a random permutation (toss-of-a-coin style classification) would achieve, on average, an accuracy rate of around 7%. Keeping that in mind, it is immediately apparent that a technique as simple as tiny image extraction improves the classification accuracy threefold. This technique requires very little processing power and employs no advanced feature extraction algorithms. Given the sporadity and chaotic nature of this data, it is not surprising that a more advanced classification technique such as SVM achieved slightly sub-optimal results when compared to the KNN classification.

Moving on to the more advanced and suitable Bag of SIFTS approach, we immediately see a much greater improvement in accuracy. Across the charts, we may see that SVM classification consistently achieves greater results than KNN. This is due to the sophistication of the algorithm, together with the use of the most suitable kernel and sigma value. The standard application of the Bag of SIFTs together with the SVM classification achieved accuracy ratings of around 66%, improving upon random assignment by a factor of 9.

The benefits of spatial tiling are clearly brought out in these results, as we see an increase of around 5% when compared to the standard approach. The inclusion of spatial information is clearly helpful when performing comparison and classification. Furthermore, the general success of feature selection has been proven, with PCA bumping up accuracy rates on average by around 1%, compared to the genetic algorithm approach which roughly maintained consistently similar accuracies. These

techniques, however, are best combined with SVM classification, as they do not indicate to bring about beneficial changes when performing KNN classification.

Overall, the system has achieved satisfactory results and has demonstrated the pros and cons of different combinations of techniques for image retrieval and categorisation. Further testing and fine tuning of the system parameters may see accuracy ratings rise to the upper 70 percentiles. Future improvements may also see further feature information being incorporated into image descriptors to increase accuracy rates further, entering the domain of state of the art systems, existing around 80-85% accuracy.

Conclusion

We may conclude that the design, implementation and evaluation of the system has been successful, with a fully functional system being deployed and achieving satisfactory results. A great deal has been learnt about the different computer vision techniques employed within this project, together with their specifications and suitability for different applications. This project has also led me to gain respect for the various image classification applications used in the real-world, and the planning and effort that is put into them.