Image Categorisation I

James Rogers, 100062949

1 Feature Detection Review

1.1 Introduction

Feature detection is an area of study that has proved to be very difficult. Multiple techniques look at specific aspects of a feature in an image such as the geometrical cues and colours [1]. Geometric cues are considered to be the most reliable [2], whereas colour can be used for routine behaviour [1].

In this section, three types of feature detection methods will be reviewed. The first being a colour feature detection algorithm proposed by Swain et. al [1] named Histogram Intersection. This looks at comparing colour histograms of images by summing the number of identical colours that occur. This algorithm has proven to recognise images that contain the same features to a high accuracy, even there is occlusion and many distractions in the background [1]. However, the illuminance of an object in an image can hinder the method [1, 3].

Furthermore, a method named visual words which was proposed by Sivic et. al. [4] will be reviewed. This technique looks at identifying certain characteristics of a features using a technology called viewpoint invariant segmentation. This technology is able to represent a feature in two different ways, shape adaption (SA), and maximally stable (MS).

Pyramid kernels is the final feature detection strategy that was reviewed. Pyramid match kernel, proposed by Grumman et. al. [10] looks at identifying features by comparing multi-resolution histograms using a weighted histogram intersection sum [10]. I also look at an adaption of this algorithm called Spacial Pyramid Matching by Lazebnik et. al. [6] which applies the pyramid matching kernel to a pyramid of subsections of an image.

1.2 Colour Indexing

Colour is considered an effective, and efficient approach when identifying features in an image [1]. Swain et. al. [1] have proposed an algorithm named histogram intersection, which is used to identify an object in a known position. The use of colour histograms have some useful properties, they are invariant to translation and rotation, and are not affected much by multiple viewing directions, and occlusion [1].

To perform a histogram intersection between two images, the histograms must first be created. Given an RGB image, the histogram will be a three dimensional matrix which contains bins for each R, G and B value. These bins will hold the number of occurrences of each colour in the image array. Since each R, G, and B value can range between 1 - 255, this histogram will be of the size 255x255x255 which is over 16.5 millions bins. This will be extremely inefficient in time, and memory. To overcome this problem, the image can be quantised to some arbitrary value, say 16. Now when storing the occurrences of each colour in the histogram, the histogram will be of size 16x16x16, 4096 bins, which is much smaller.

To compute a histogram intersection, the difference between two histograms must be calculated. Given two histograms, I and M in which both contain n bins, the equation (1) will sum the number of pixels that contain the same colour. The larger the result, the more likely it is to be the same.

$$\sum_{i=1}^{d} \min(\mathbf{I}_j, \mathbf{M}_j) \tag{1}$$

Colour histogram intersection is able to handle common problems that hinder recognition quite well, such as distractions in the background, the view point, occlusion and image recognition [1]. However, the illuminance of the object can greatly affect the overall accuracy [1, 3]. One solution is to perform a colour-constancy algorithm named supervised colour constancy which uses a colour chart to provide information about the illuminants. This colour chart provides an extremely reliable reference criterion, as a large number of coloured squares provide enough constraints to calculate more basis coefficients than other colour constancy methods [7]. Another solution is to use HSV colour space. HSV separates the colours HS, from the illuminance, V. Building the histogram with only the HS data will make it less sensitive to illuminance [8]. With one of these proposed methods in place, the illuminance of the object will not impact the recognition of the feature as much.

1.3 Visual Words

Visual words [4], is a form of feature detection that is used to recognise objects in images. These visual words, represent a visual representations of features. The algorithm is able to identify particular objects with ease, speed, and accuracy, which is great for real time applications [6].

To identify objects, a database, know as a Visual Vocabulary contains visual descriptors of the objects [4]. To build a visual vocabulary, it must be trained using a set of training data. Firstly, invariant segmentations must be constructed for each training image. Once all of the invariant segmentations have been identified, the descriptors must be vector quantised into clusters.

To create a visual descriptor of a word, a technique called viewpoint invariant segmentation can be used. The viewpoint invariant is an effective technology at describing objects, as it is "largely unaffected by a change in camera viewpoint, the object's scale, scene illumination and will be robust to some amount of partial occlusion" [4]. There are two types of viewpoint invariant regions. The first type, shape adaption (SA), constructs an elliptical shape about an interest point which encompasses the feature [9] . The second algorithm, named maximally stable (MS), is built by selecting areas form an intensity watershed. Both of these methods are used as they represent different image areas Figure 1 [4].

To vector quantise these visual descriptors, an unsupervised image classification algorithm, k-means clustering is performed using the Mahalonobis distance equation (2)

$$d(\mathbf{X}_1, \mathbf{X}_2) = \sqrt{(\mathbf{X}_1 - \mathbf{X}_2)^T \sum^{-1} (\mathbf{X}_1 - \mathbf{X}_2)} \qquad (2)$$

Each viewpoint invariant strategy, SA and MS are clustered separately as each of them describe different types of data. Each cluster created by the k-means clustering is likely to represent the same word.

A method named, a bag of words can be used to make up features. When visual descriptors of an image are compared to the visual vocabulary, if the majority of the visual descriptors match the words that make up a face, it can be determined that a face is in the image. Using a bag of words to identify image features can provide accurate results providing there is no occlusion [4].

1.4 Pyramid Kernels

A pyramid kernel named pyramid match kernel proposed by [10], maps unordered feature sets to multi-resolution histograms, and then compares a weighted histogram intersection [1] to approximate the similarity [10].

The pyramid match kernel works by increasing coarser grids over the feature space and taking a weighted sum of the number of matches that occur at each level of resolution. [6]. The equation (3) [10], finds the similarities between the test set and each training set.

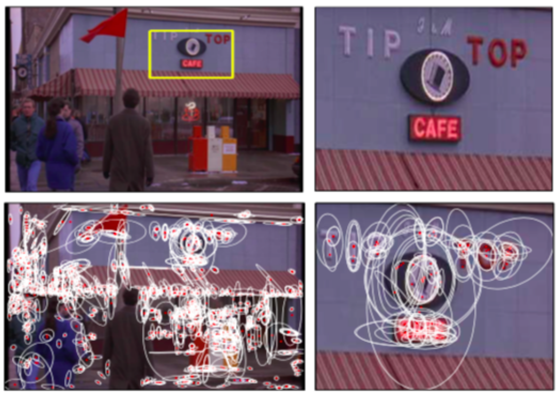
Figure 1: An image visualising all detected visual descriptors.

$$\mathbf{K}_\Delta(\Psi(\mathbf{y}), \Psi(\mathbf{z})) = \sum_{i=0}^{L} w_i \mathbf{h}(\mathbf{y}_i, \mathbf{z}_i) \qquad (3)$$

Where $\mathbf{K}_\Delta$ is the similarity between each set, $\Psi(\mathbf{x})$ is a vector containing each subsequent histogram that has bins doubling in size. L, the number of histograms for each set. w_i the weight, and $h(\mathbf{y}_i, \mathbf{z}_i)$ denotes the histogram intersection function (1) [1].

The pyramid match kernel is linear in the number of features $0(n)$ [10] and is much more efficient when compared to other algorithms such as the Match Kernel [11]. It has proved to be robust to clutter as it does not penalise the presence of extra features [10]. A problem with this approach is that it discards all spatial information in the image. Another problem is the quality of the approximation to the optimal partial match, provided by the pyramid kernel degrades linearly with the dimension of the feature space [12]. The results produced by [10], using a support vector machine show that the algorithm does not perform well with images that contain signification clutter and occlusions, giving 43% success rate. However, results using the publicly available ETC-80 database returned a success rate of 83% [10].

An adaption of the pyramid matching kernel named Spatial Pyramid Matching [6] looks to improve upon the algorithm with accuracy, particularly in challenging scenes. This algorithm subdivides the two dimensional image into equal regions as shown in Figure 2. Then the pyramid match kernel (3) [10] is performed on each sub division [6].

According to the results carried out by [12] the algorithm produces promising results, however, the spatial pyramid method is not meant as a sufficient or definitive solution to the general problem of scene recognition or understanding [12].

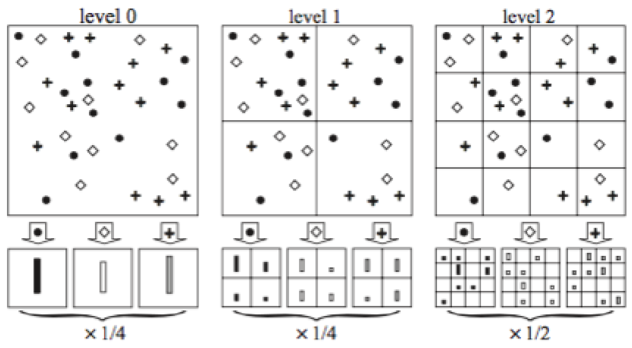

Figure 2: An example of constructing a three level pyramid. There are three freature types in this figure, circles, diamonds, and crosses. For each level of resolution and each channel, we count the features that fall in each spatial bin.

1.5 Conclusion

Focusing on colour to recognise an image is able to handle occlusion and multiple viewing directions elegantly [1], however, the illuminance of an object blinds the algorithm and causes incorrect results [1]. Although methods of counteracting illuminance of an object improves upon the accuracy, it is still relatively low [1]. Using colour to identify objects in an environment where the lighting can be controlled, in a factory for example will drastically improve performance. However, this approach is not feasible in a environment where lighting cannot be controlled [1, 2].

Visual words uses geometric cues to identify features in an image. Recognising the geometric information handles illuminance very well, unlike colour [2]. Occlusion however, can greatly effect the visual words performance, where as colour is able to handle this gracefully [1]. The bag of words is a good way of distinguishing features by identifying multiple geometric ques that make up a feature [4].

Although the adaption of the pyramid matching kernel, spatial pyramid kernel provides good results [12], “it is not meant as a sufficient or definitive solution to the general problem of scene recognition or understanding” [12].

2 Image classifier Review

2.1 Introduction

Image classification is a highly difficult and complex problem that has attracted much attention in the recent years. There are a various number of classifiers that can be used with images. Many classifiers fall into categories of supervised, or unsupervised learning. A supervised learning classifier takes in labeled training data, where as an unsupervised learning classifier does not. There are also parametric classifiers and non-parametric classifiers. Parametric classifiers are statistics based on parameterized families of probability distributions, non-parametric classifiers are not. There are also classifiers that use different kind of pixel information, some of these being per-pixel classifiers, sub-pixel classifiers and per-field classifiers. Per-pixel classifiers consider every pixel individually, which can cause inaccuracies from noise such as salt and pepper. Sub-pixel classifiers provide a more appropriate representation and accurate area estimation of images than per-pixel approaches. Per-field classifiers average out the noise by using land parcels (called fields), which are generally more accurate than per-pixel classifiers [13].

In this paper, various per-pixel, supervised image classification techniques will be reviewed to determine their strengths and weaknesses.

2.2 Nearest-Neighbour

One of the oldest Machine Learning algorithms, the Nearest-Neighbour (NN) [14] is a per pixel classifier that uses a supervised training model. It classifies data by finding the smallest distance between the test sample, and each of the training samples. The test sample will be classified as the classification of the closest training sample. The distance is typically calculated by using the Euclidean distance function

$$\mathbf{D}_{Euclid} = \sqrt{\sum_{i=1}^{d}(q_i - c_i)^2} \tag{4}$$

However, the Nearest Neighbour algorithm supports any measure of distance such as Minkowski distance, Manhattan distance etc.

To reduce the generalisation error, a modification of the algorithm called K-Nearest-Neighbour has been introduced. Instead of classifying data based on the closest training sample, a majority vote of the K nearest neighbours will be chosen to produce the classification. To eliminate an equal number of votes, K is usually an odd number, however, this only works with a binary class problem. If an even number of votes do arise on a multi-class problem, a random classification will be picked.

The NN classifier is considered to be slow as it classifies each test sample in a time of O(n) [14], where n is the number of training samples. However, the speed of the algorithm can be improved by removing samples from the training set that does not affect the decision boundary [15].

An advantage of the NN classifier is it produces good results with multi-modal classes because the basis of its decision is based on a small neighbourhood of similar objects [16]. The disadvantage of the NN classifier is it weights all of the features equally which can cause classification errors. Another disadvantage, is it requires

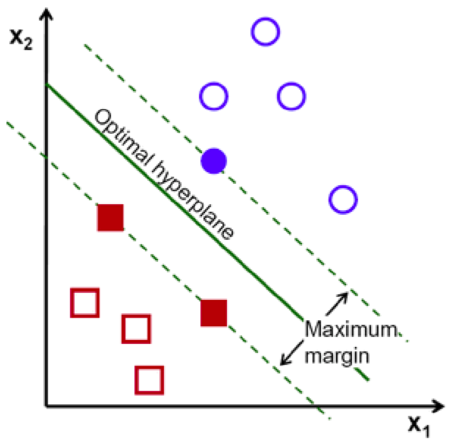

Figure 3: An example of a support vector machine. The optimal hyperplane successfully seperates the data with a maximum possible margin.

the storage of N data samples. If N is large, then it may mean an excessive amount of storage [17].

2.3 Support-Vector-Machine

The Support Vector Machine (SVM) [13], is another per-pixel classifier that uses a supervised learning model to classify test data. The SVM is able to classify linearly separable data, as well as non-linearly separable data when using what is called a kernel trick [18].

To build the classifier on a binary, linearly separable problem, a hyperplane must be defined to separate the data sets. To do so, find the shortest line segment between the convex hulls of each data set. The training patterns that define the closest points are known as Support Vectors. The hyperplane will be perpendicular to the mid point of this line segment. This hyperplane maximises the margin between the datasets Figure 3 to lower the generalisation error [16].

To classify the test data using the model created by the training data, use the equation (5)

$$\mathbf{W} \cdot \mathbf{X} + b \tag{5}$$

Where $\mathbf{W}$ is the hyperplane, $\mathbf{X}$ is the datapoint under test, and b is some offset. The sign of the result will determine which side of the hyperplane the data point lies.

When classifying non-linearly separable data, a hyperplane cannot yet be found as there is no linear separation. To overcome this problem, a kernel trick is used to transform the data into a high-dimensional space that is linearly separable [19] Figure 4. Kernel functions such as Polynomial and Gaussian radial basis can be used to transform the data into this new space. To define the hyperplane, the same concept of Support vectors can be used. To classify the test data using this new model, the data must be transformed into the same high-dimensional space as the training data, which can be done by applying the same kernel function [18].

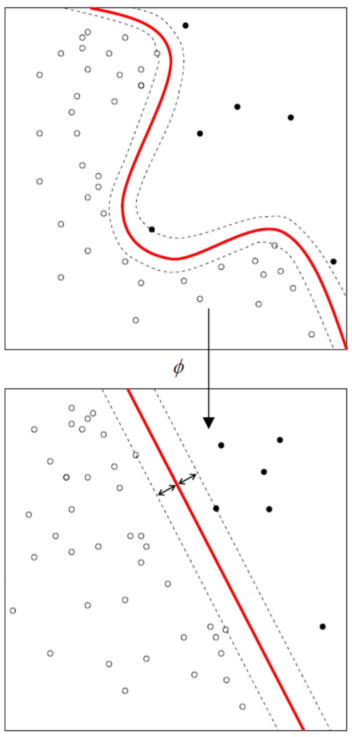

Figure 4: An example of a support vector machine in which the data is transformed into the high-dimensional feature space using a kernal.

One of the main advantages of the Support Vector Machine is that it can produce very accurate results [16]. The SVM also yielded the most accurate classifications when compared against three other classifiers, Maximum Likelihood, Artificial Neural Network, and a Decision Tree in land cover classification [19]. Another advantage is that is has the capability of classifying linearly and non-linearly separable data when given a kernel function. One of the disadvantages of this classifier is it does not natively support multi class problems, however, several methods have been proposed to overcome this problem [5].

2.4 Artifical Neural Networks

The Artificial Neural Network (ANN) is a non-parametric, per pixel classifier which also uses a supervised learning model. This classifier is modelled after biological neurons. The neuron sums inputs from dendrites via synapses. The synapses weight the inputs from other neurons, then fires if this sum exceeds some threshold via an axon. The learning involves change in the synaptic weights. The Perceptron is a simple artificial model of the neuron, it calculates a weighted sum of all of its inputs known as the activation.

It then outputs a non-linear function of this activation. The perceptron is able to classify a linearly separable problem with ease, however, it cannot classify non-linearly separable data.

The Artificial Neural Network is a network of these Perceptrons, which enables the capability of classifications on non-linearly separable data. This network is made up of layers of Perceptrons Figure 5. The first layer, called the input layer, takes in some inputs and passes them to the next layer. The middle layers compute the weighted sums of their inputs, and pass them on to the output layer. The output layer is the final layer which contains the final result. The number of hidden layers in the ANN greatly contributes to the accuracy of the overall classifier.

When building and training the ANN classifier, a training algorithm called Back-Propagation is used which updates the weights iteratively after each pattern until either some threshold is met, or the classifier achieved an accuracy of 100%. Other training algorithms exists such as Newtons method which is good for training medium size networks, Conjugate Gradients, which is good for training very large networks, and Levenburg-Marquardt. [18].

There are many advantages that come with the Artificial Neural Network, such as its non-parametric nature, arbitrary decision boundary capability, easy adaptation to different types of data and input structures [22]. However, the ANN typically suffers from very slow training times, depending on the number of hidden layers.

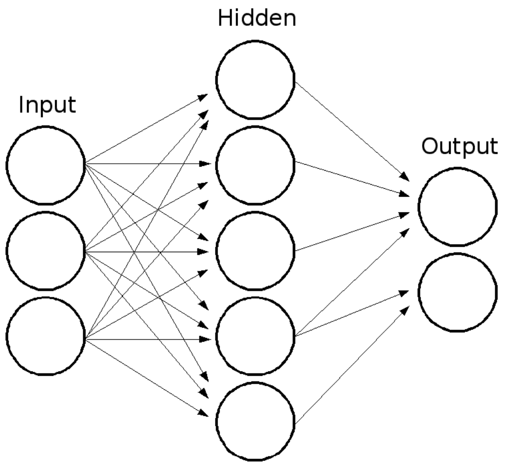

Figure 5: An artifical neural network. The left most layer represents the input layer which takes in some input values. The middle layer represents hidden layers which compute weighted sums from the input layer. Finally, the right most layer signifies the output layer which contains the weighted sums generated by the hidden layers.

Chen et. al. [20], compared the ANN using back-propagation to a dynamic learning ANN in SPOT high resolution visible multispectral imagery. The results indicate that the use of dynamic learning greatly reduced the training time when compared to back-propagation. The accuracy of the both classifiers was shown to be excellent. However the DL was considered to be the most effective as it produced promising classification results in terms of training time and classification accuracy [20].

2.5 Conclusion

Nearest neighbour classifiers are the simplest, and easiest to implement of the three classifiers reviewed here. It produces good results with multi-modal classes[16] and natively supports multi-class classifications, however all features are weighted equally which could hinder the performance of the classifier. Nearest neighbour methods also take a long time to classify data as the distance between the test data and all of the train data needs to be calculated and compared, if the training data is large, it can also take up a lot of memory [17]. Although, the previous two problems can be improved upon by removing training data that does not affect the decision boundary [15].

Support vector machine classifiers provide a very high accuracy when classifying images. Their ability to classify linearly and non-linearly separable data given a kernel function makes them very attractive and flexible [18]. However, their non-native support for multi-class classification requires modifications to such as one-vs-all [21].

Artificial Neural Networks are very powerful and provide accurate results depending on the number of hidden layers. It is made up of many perceptrons and aims to replicate the biological network of neurons. It is a non-parametric classifier [22] and a universal approximator rendering it very flexible. The ANN however, suffers from slow training times, again depending on the number of hidden layers.

In conclusion, a basic explanation has been provided of the nearest neighbour, support vector machines, and artificial neural network methods, as well as strengths and weaknesses.

3 Report

3.1 Introduction

Image classification and scene recognition is a huge field of research that has recently become very popular. Various algorithms and techniques are used to represent and recognise an image either geographically, like shapes, or colours. Geographic scene recognitions are considered to be the most accurate and reliable, and colour is considered a secondary to geometric cues in most cases [14].

This report aims to detail provided implementations of tiny images, colour histograms, and k nearest neighbour in depth, and address findings to determine how feasible these algorithms are in scene recognition.

3.2 Implementation

Two scene recognition algorithms, tiny images, and colour histograms have been implemented using Matlab. In this section, each step of the algorithms will be thoroughly explained, and any design choices made will be discussed in depth.

3.2.1 Tiny Images

The tiny images algorithm reduces images to a small fixed size, around 16x16. The idea is to allow for efficient per pixel comparisons between images while ignoring small details.

The implementation takes in the following variables as parameters, image_paths, dimension, and colour_space. The first parameter image_paths, is a cell vector where each row holds a string representation of an image path. dimension, is an integer that defines the target size of the image. colour_space is a string that represents the target colour space of the image e.g. Grayscale, RGB, YCbCr, and HSV. The function returns a matrix named image_feats which contains multiple vectorised images.

Firstly, the image_feats matrix is initialised to a dimension that is dependent on the image dimensions, and colour space. Next, to generate the images, the algorithm iterates over each element in the image_paths vector. With each iteration, the following seven steps are performed:

- Open the image using the image path in the current iteration of the image_paths vector.
- Resize the image to n x n where n is equal to the parameter, dimension.
- Convert the image into the colour space defined by the variable, colour_space.
- Transform the image matrix to zero mean.
- Normalise the image matrix.
- Vectorise the image matrix.
- Place the vectorised image into the appropriate row of the image_feats matrix.

Once the iteration of the image_paths has completed, the function will return the image_feats matrix.

The reason for transforming the image so that it has a mean of zero and is of unit length is because it increases efficiency, allowing the algorithm to perform at a faster rate. Vectorise operations have neen used where ever possible to produce cleaner code, eliminate further iterations, and increase efficiency. The steps 4, and 7 are vectorised to benefit from the stated advantages.

3.2.2 Colour Histogram

Colour histogram provides a colour histogram with bins which hold the number of occurrences of each colour. This algorithm is designed to compare images using colour similarities.

This implementation requires the following parameters, image_paths, is a cell vector where each row holds a string representation of an image path. dimension, is an integer that defines the target size of the image. quant, represents the number of quantisation levels. colour_space is a string that represents the target colour space of the image e.g. RGB, YCbCr, and HSV. The returned variable in this function is a matrix named image_feats which contains multiple vectorised images.

The image_feats matrix is firstly initialised to a dimension that is able to hold each vectorised image histograms. The number of rows will be defined by the number of images, which can be retrieved from the number of rows in the image_paths matrix. The number of columns will be quant ^{3} as the histograms are three dimensional. Next, to generate the colour histograms, the algorithm iterates over each element in the image_paths vector. With each iteration, the following nine steps are performed:

- Open the image using the image path in the current iteration of the image_paths vector.
- Resize the image to n x n where n is equal to the parameter, dimension.
- Convert the image into the colour space defined by the variable, colour_space.
- Convert the image matrix into doubles in preparation for quantisation.
- Quantise the image matrix using the given number of levels, quant.
- Initialise a three dimension integer histogram to a size of n x n x n where n is the value, quant. Fill this histogram with zeros.
- Increment each index in the histogram by one. The indices is given by the colour of each pixel in the image matrix.
- Vectorise the image matrix.
- Place the vectorised image into the appropriate row of the image_feats matrix.

Once the iteration of the image_paths has completed, the function will return the image_feats matrix.

Swain et. al. [1] concluded that colour histograms are not dependent on image resolutions, therefore the flexibility of resizing the image was provided to increase computational performance, and not greatly sacrifice detection accuracy.

3.2.3 K Nearest Neighbour

K-Nearest-Neighbour classifier to classify images provided by the tiny image, and colour histogram.

The following input parameters required for this function are train_image_feats, which is a matrix that contains vectorised training image feature data. train_labels, is a cell vector where each row contains a string of the category of the corresponding index in train_image_feats. test_image_feats, is a matrix much like the train_image_feats matrix, expect filled with test feature data. k, is an integer representing the number of nearest neighbours to be considered when casting a vote. The function returns a cell vector named predicted_categories where each row contains a string of the predicted category of the corresponding index in train_image_feats.

The first task in this method is to initialise the predicted_categories vector to a size that is of the same length as the number of rows in train_image_feats matrix, i.e. the number of images. Next, we iterate over each test image and perform the following seven steps:

- Get the test image from the test_image_feats matrix.
- Calculate the absolute difference of each pixel between the test image, and each train image. Store this as a matrix of n rows where n is the number of train images, and m columns, where m is the number of pixels in the image.
- Sum each row of the difference matrix and store in a vector with n rows where n is the number of train images.
- Sort the differences into ascending order.
- Store the labels of the k smallest differences in a cell vector.
- Find the most common category (vote).
- Store the vote into the appropriate index of the predicted_categories vector.

The use of vectorisation when constructing the matrix of the absolute differences between the test image and the training images produces cleaner code, eliminates further iterations, and increase the efficiency. The voting system in this algorithm works by calculating the mode of votes. If there is no distinguishing majority vote, the first majority vote will be chosen as the classification.

3.3 Test Data

In this section, a basic description of the test data will be provided.

A data base of 3000 images was used to perform the tests on the three algorithms. The images in the database fell into one of fifteen categories, kitchen, store, bedroom, living room, house, industrial, stadium, underwater, tall building, street, highway, field, coast, mountain, forest. These categories are distributed equally across the database, meaning 200 images represent each category. To issue test and train datasets, this database is divided in to two subsets, distributing the number of categories equally across each subset.

These images vary in characteristics as some categories represent a room indoors, and some represent urban and natural outdoor environments. Certain categories such as bedrooms and living rooms should be extremely difficult for these algorithms to distinguish as their tiny images and colour histograms are likely to be very similar. Indoor and outdoor categories such as store and forest however should be easier to distinguish as these images are completely different.

3.4 Tests

This section will highlight all of the tests performed, and detail reasons behind any choices made. An explaination of why certain parameters have, and have not been useth justification..

In order to find the best possible results, the tests performed used a wide range of parameter combinations. Tables 1 and 2 give an overview of all parameters used for each algorithm.

3.4.1 Tiny Images

The size parameter has been used at 8, 16, and 32. These three values have been chosen to identify patterns when the size of the image is halved consecutively. Although colour space was not required for tiny images, a test was performed using these parameters to identify if any considerable impact can be obsevered. Since colour space can expose different elements and completely change the look of an image, results may vary considerablty. K, for the k nearest neighbour ranged from a relatively high value, 15 to 1. If the nearest neighbour classifier is able to match the majority of the test images to their respective category, an increase in K should give more accurate results, which is why I went for a high value, 15.

Table 1: Tiny Image Test Parameters

	1	2	3	4	5
Dimension	8x8	16x16	32x32		
Colour Space	Grayscale	RGB	YCbCr	HSV	
K	1	3	5	9	15

Table 2: Colour Histogram Test Parameters

	1	2	3	4	5
Dimension	8x8	16x16	32x32		
Colour Space	RGB	YCbCr	HSV		
Quantisation	8	16			
K	1	3	5	9	15

3.4.2 *Colour Histogram*

The size of the colour histogram is going to be used in the range of 8, 16, and 32, just like tiny images. The reason for doing so is because the tests need to be kept fair. Another reason for choosing these are because it will be interesting to see if size effects the histograms. As stated by [1], the resolution does not greatly affect the colour histogram. Colour space again, is changed due to the fact that this is a colour histogram, and the change in colour space should drastically alter the outcome. RGB, YCbCr, and HSV colour spaces will be used. HSV should be interesting as it separates the illuminance from the colour, which should improve improve the reliability of the colour histogram [1]. The colour quantisation is going to be tested using 8, and 16 levels of colour. The reason for not setting this value any higher as it will quickly hinder performance. The number of quantisation levels will increase the number of bins in the histogram cubically. Like tiny image, K was chosen to range from 1 - 15.

3.5 Results

This section will give details of the results achieved in the tests. As there were many combinations of results, A brief overview of the findings wil be detailed, as well as descriptions of graphs that represent specifc tests of interest.

3.5.1 *Tiny Images*

After running all of the variations of the parameters for tiny images, certain trends were successfully observed. The first trend being the size, appeared to produce higher overall accuracies as the image decreased in resolution. It is suspected that the increase in accuracies is due to the loss of specific details in which gives a better overall scene recogniser.

Another trend observed was the difference in accuracies due to colour space. It appears that RGB consistently provides the best accuracies, followed by HSV, then YCbCr, and finally grayscale. Grayscale produced considerably terrible results when compared to the other three colour spaces in all tests.

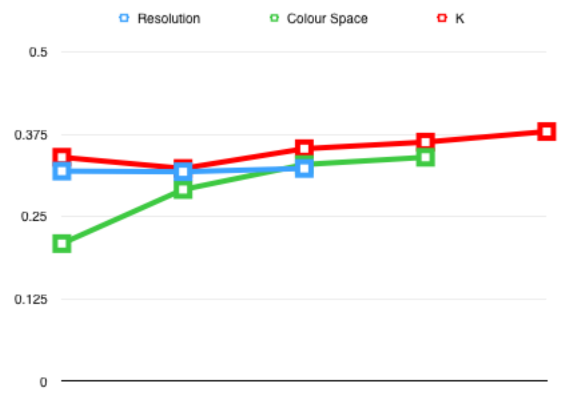

Figure 6: Tiny Image results displaying multiple plots of the results with different parameters. Resolution (32x32, 16x16, 8x8). Colour Space (Grayscale, YCbCr, HSV, RGB). K (1, 3, 5, 9, 15).

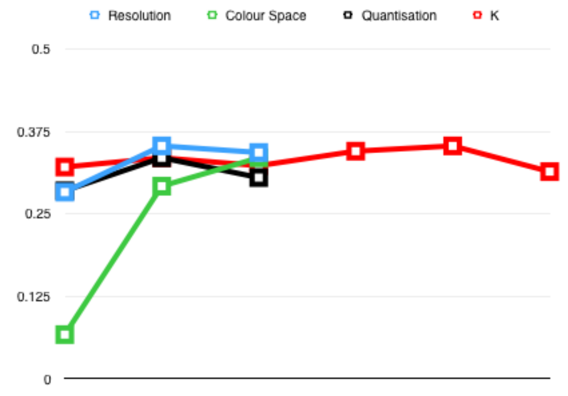

Figure 7: Colour Histogram results displaying multiple plots of the results with different parameters. Resolution (32x32, 16x16, 8x8). Colour Space (HSV, YCbCr, RGB). Quantisation (4, 8, 16). K (1, 3, 5, 9, 15).

The last trend witnessed was with K nearest neighbours. Judging from the experiments, accuracies increase along with K, suggesting the classifier was able to find majority matches of correct categories.

Figure 6 represents the three trends observed in the experiments for Tiny Images.

3.5.2 *Colour Histogram*

Unlike the tiny images results, only one pattern with the colour histogram could be identified. This pattern was identified with the change of colour space. RGB appeared to perform consistently better than the other two colour spaces. HSV however, saw 0.067% accuracy which is equal to 1/15 across all tests. As stated by [1], the HSV should have performed better since it removes some of the illuminance from the colour. I suspect that this has not been implemented correctly.

The resolution of 16x16 was seen to give the best results. This may be because a resolution of 8x8 is too small and destroys most of the colours, and 32x32 holds the information of too many colours. Perhaps 16x16 is the optimum resolution for a colour histogram, however, no such conclusion can be made until more extensive testing is done.

Quantisation and K neighbour results appeared random. There was no direct correlation with the change in parameters and classifier accuracy. More testing will need be done to better understand the impact of quantisation and k nearest neighbours.

Figure 7 represents plots of the four parameters.

3.5.3 Overview

After analysing the results of tiny images and colour histograms, some attributes have been identified that consistently provide strong accuracies.

The colour space, RGB, appears to be the most accurate out of Grayscale, YCbCr, and HSV across both algorithms. However, HSV is suspected to have not been implemented correctly for colour histograms as it achieved an accuracy of 0.067%, compared to 0.292% in tiny images.

Multiple values of K 1, 3, 5, 9, 15 was tested on each algorithm. The accuracy in tiny images appeared to grow as k increased, where as colour histograms did not show any correlation because of the random/unpredictable accuracies. Further experiments in the future could be performed to achieve even higher accuracies by testing tiny images with higher values of K.

Various resolution sizes 32x32, 16x6, and 8x8 where performed on tiny images and colour histograms to identify optimal resolutions. The accuracies in tiny image increased every time the resolution halved, meaning 8x8 provided the strongest results. Colour histograms on the other hand achieved higher accuracies with 16x16, which just outperformed 8x8 by 1%.

The quantisation values 4, 8, and 16 in the colour histogram showed that 8 levels of quantised colours gave the best results, followed by 16, then 4. It is believed that the quantisation of 4 levels eliminated too much colour data from the images which led to the weakest results.

The highest results achieved in each parameter of have been combined to produce the most accurate possible classification for these algorithms. The tiny Image classifier achieved an overall accuracy of 0.379 Figure 8, and the colour histogram achieved an overall accuracy of 0.353 Figure 9

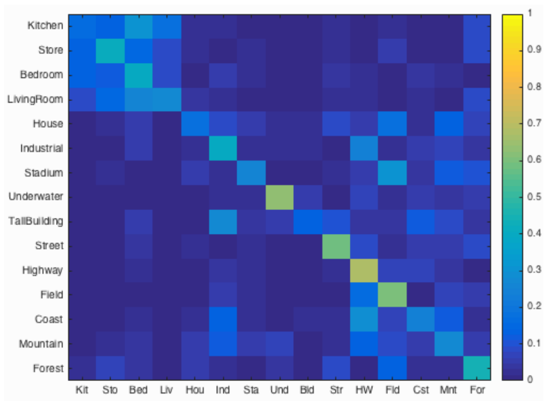

Figure 8: Tiny Image confusion matrix. Resolution 8x8, Colour Space RGB, K-15

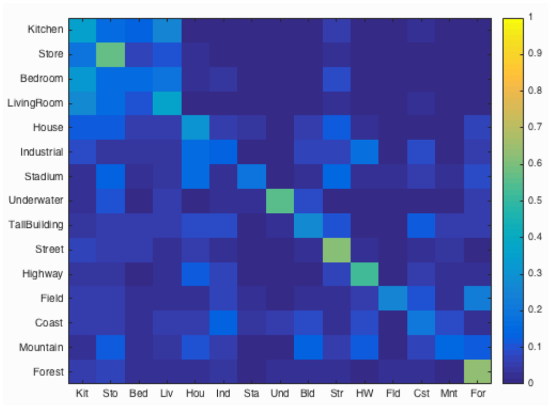

Figure 9: Colour Histogam confusion matrix. Resolution 16x16, Colour Space RGB, Quantisation 8, K-15

3.6 Conclusion

To conclude this report, two scene recognition algorithms, tiny images, and colour histograms have been implemented. Aan image classifier, k-nearest-classifier has also been implemented to classify feature data provided by the two scene recognition algorithms.

A series of tests has been performed on the three algorithms by training them on a 1500 image data set, and testing them using another 1500 image data set. The tests consisted of classifying each image to 1 of 15 categories.

Further work can be pursed by implementing more advanced scene recognition algorithms and classifiers reviewed in this document, like spatial pyramids, and support vector machines.

References

[1] J. Swain and D. Barrard, "Color indexing," in *International Journal of Computer Vision*, 1991, pp. 11–32.

[2] I. Biederman, "Human image understanding: Recent research and a theory," in *Computer Vision, Graphics, and Image Processing 32*, 1985, pp. 29–73.

[3] M. Yang, D. Kreigman, and N. Ahuja, "Detecting faces in images: A survey," in *IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 24, No. 1*, 2002, pp. 34–58.

[4] J. Sivic and A. Zisserman, "Efficient visial search of videos cast as text retrieval," in *IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 31, No. 4*, 2009, pp. 591–606.

[5] C. Hsu and C. Lin, "A comparison of methods for multi-class support vector machines," in *Neural Networks, IEEE Transactions on (Volume:13 , Issue: 2)*, 2002, pp. 415–425.

[6] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognising natural scene categories," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2169–2178.

[7] C. Novak and S. Shaffer, "Supervised color constancy using a color chart," 1990.

[8] O. Chapelle, P. Haffner, and V. Vapnik, "Svms for histogram based image classification," in *Neural Networks, IEEE Transactions*, 1999, pp. 1055–1064.

[9] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Fourth Alvey Image Vision Conf*, 1988, pp. 147–151.

[10] K. Grauman and T. Darrel, "The pyramid match kernel: Discriminative classification with sets of image features," in *In Proceedings of the IEEE International Conference on Computer Vision*, 2006, pp. 1458–1465.

[11] C. Wallraven, B. Caputo, and A. Graf, "Recognition with local features: the kernel recipe," in *In Proc. IEEE International Conf. on Computer Vision*, 2003, pp. 256–264.

[12] S. Lazebnik, C. Schmid, and J. Ponce, "Spatial pyramid matching," in *In Object Categorization: Computer and Human Vision Perspectives*, 2009, pp. 256–264.

[13] D. Lu and Q. Weng, "A survey of image classification methods and techniques for improving classiffication performance," in *International Journal of Remote Sensing*, 2007, pp. 823–870.

[14] E. Fix and J. Hodges, "Discriminatory analysis, non-parametric discrimination," in *consistency properties, Tech Report for USAF School of aviation and medicine*, 1951.

[15] D. Bremner, E. Dermaine, J. Erickson, J. Iacono, S. Langerman, P. Morin, and G. Toussaint, "Output-sensitive algorithms for computing nearest-neighbour decision boundaries," in *Algorithms and Data Structures*, 2003, pp. 451–461.

[16] J. Kim, B. KIM, and S. SAVARESE, "Comparing image classification methods: K-nearest-neighbor and support-vector-machines," in *Applied Mathematics in Electrical and Computer Engineering*.

[17] A. Webb, "Statistical pattern recognition," in *Statistical Pattern Recognition, Second Edition*, 2002.

[18] C. Cortes and V. Vapnik, "Support-vector networks," in *Machine Learning*, 1995, pp. 273–297.

[19] C. Huang, L. Davis, and J. Townshend, "An assessment of support vector machines for land cover classification," in *Int. j. remote sensing*, 2002, pp. 725–749.

[20] K. Chen, Y. Tzeng, W. Kao, and C. Ni, "Classification of multispectral imagery using dynamic learning neural network," in *Geoscience and Remote Sensing Symposium, 1993. IGARSS '93. Better Understanding of Earth Environment., International*, 1993, pp. 869–898.

[21] C. Burges, "A tutorial on support vector machines for pattern recognition," in *Data Mining and Knowledge Discovery 2*, 1998, pp. 121–167.

[22] J. Paola and R. Schowengerdt, "A review and analysis of backpropagation neural networks for classification of remotely-sensed multi-spectral imagery," in *International Journal of Remote Sensing Vol. 16 Issue. 16*, 1995, pp. 3033–3058.