MODULE: CMP6035B - Computer Vision
ASSIGNMENT TITLE:
Image categorisation II

Student ID: 100057593

Introduction

The purpose of this assignment is to implement image recognition techniques/algorithms via various methods. Using a set of provided training data and test data, these implementations can be contrasted; additionally, some techniques/algorithms have relevant parameters which can be altered to manipulate the outcome, in these instances the same algorithms can be contrasted in order to produce a comparison between two instances of the same algorithm.
The resulting effect of different methods and varying parameters is to then be evaluated based upon the accuracy and reliability of the outcomes produced.

This report will discuss the problem domain and core concepts/techniques, before going on to document the implementation and testing of each implemented algorithm, with an evaluation regarding its performance and applicability.

Problem Domain

Image categorisation is a very broad field with an abundance of techniques and approaches, with the quality of output of each of these techniques/approaches being dependent on application in question. Within the context of this assignment, the methods/algorithms used can be segregated into two types, *Image Classification* algorithms and *Image Feature Extraction* algorithms. During the development of the first coursework, Image Categorisation 1, the k-Nearest Neighbour (kNN) image classification algorithm was used along-side the Tiny Images image feature extraction algorithm to produce rather inaccurate results and do not provide a very good performance. This implementation is to be extended to include an additional *Image Classification* algorithm known as the Support Vector Machine (SVM) classifier. In regards to the image feature extraction algorithms, the Tiny Images algorithm is to be replaced with two others; the first being the Scale-Invariant Feature Transform (SIFT) algorithm - applied to both greyscale and colour images - and the second being the Spatial Pyramid Kernels, although not strictly an image feature extraction algorithm, it allows for images/scenes to be represented and pixel weightings to be applied and manipulated.

Core Topics

The following section explores the various concepts and techniques, outlining their basic functionality, purpose and differences. This will give an indication on the predicted behaviour and results of these concepts/techniques prior to implementation. Subsequent to the discussion of each concept, the implementation approach will be described.

K-Nearest Neighbours (kNN)

Given a set of training images and test images, the kNN image classification algorithm identifies the *k* nearest neighbours of a specified point. After determining and retrieving the *k* amount of nearest neighbours, each of these nearest neighbours has a vote regarding its classification; the votes are tallied and the specified point is given the classification with the highest total votes. In the case that *k* is equal to 1, a region of space is defined by each training image resulting in a *Voronoi* partition. During testing, each image is classified dependant solely on the region in which it is placed. Visual representations of how the algorithm performs under these two *k* conditions can be found within Figure 1 and Figure 2.

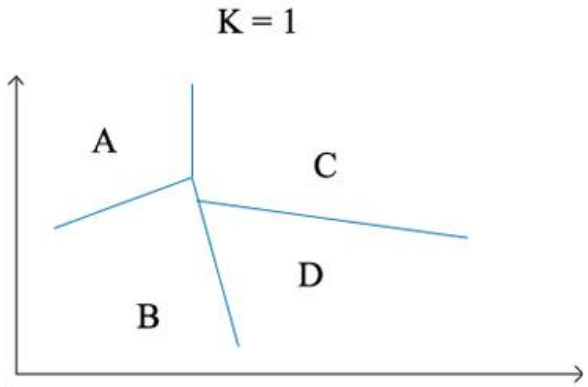


Figure 1

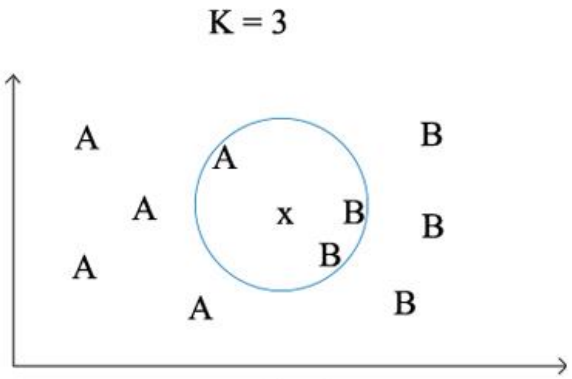


Figure 2

The kNN is considered a *lazy learner* algorithm as it relies on the whole training data set in order to categorise an input image. As a lazy learner the kNN algorithm does not perform any generalisation regarding test data prior to receiving an input image, resulting in minimal processing and time required to execute the training phase at the expense of time efficiency during the testing phase. Non-parametric is also a term used to describe the kNN algorithm as it makes no assumptions about the distribution of data, a quality which can prove desirable as real-world/practical data sets do not consistently meet assumptions formed by other image classification algorithms.

Lastly, it is important to note aspects which could be considered disadvantages of the kNN algorithm. Although adding further diversity to the algorithm via the addition of a *k* variable, this also introduces the requirement of tuning such parameters; similarly the chosen distance metric chosen to calculate the nearest neighbour can determine the correctness of results and may also need tuning. Furthermore, it can prove sensitive to outliers as if a point is very far away it would never be used as the distance is too great.

Implementation process for kNN classifier:
1) Loop through the data set of images, reading in the current image on each loop iteration.
2) For the current image, calculate/store the distance between the training images.
3) Find the label with the least distance to the current image.
4) Looping through the rows in results to find each set of unique labels to the current image.
5) Assign the label for the current image which has occurred the most.

SVM

SVM is used with data sets which are linearly separable and aims to implement a hyperplane, at a suitable point of separation, which classifies all points of the training data set into two individual classifications. It is desirable for the hyperplane to be positioned so that it has the maximum distance/margin from the two different classifications. The SVM classifier is defined as an *eager learner* as it is able to dismiss irrelevant training data, using a subset of the training data instead. In contrast to a *lazy learner*, an *eager learner* can prove more time efficient during the testing phase.

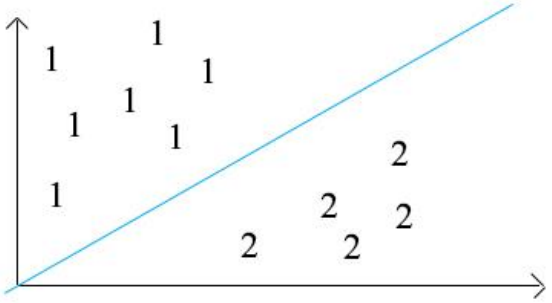


Figure 3

A visual representation of the SVM hyperplane and data separation can be found in figure 3.

The hyperplane which separates the two classifications can be represented by an equation; this equation results in values greater than one for all points classified on the first side of the plane, and results in values less than one for all points classified on the opposing side.
Similarly to the previously discussed kNN classifier, SVM requires parameter tuning in order optimise the resulting output. Two parameters which can be altered and are considered critical include the *Kernel* and the *c* variable. The kernel allows for the SVM classifier to perform as either a linear or non-linear function, while the *c* variable represents the cost of classification. When a large value of *c* is set it results in a low bias but high variance, whereas setting *c* to a small value results in the contrary - high bias but low variance; the *c* variable allows for a trade-off to be made between stability and error-penalisation.

Implementation process for SVM classifier:
1) Create a binary representation of the data set of images.
2) Loops through each category and creates a SVM using the *vl_svmtrain* function.
3) Calculates the score for each of the test images, storing the max score index.
4) Determines the predicted category by using the label with the max score.

Visual Words/Vocabulary

During the performance of image classification, models such as the *Bag of Words (BoW)* model can be used to represent image features as visual words. There are a number of stages to the model which allow for this representation; initially any points of interest (such as corners, edges and junctions for example) are to be located. Neighbourhoods are then built around and based upon these selected points of interest, which can be described by features such as SIFT features.
Once all points of interests within an image have been located and given a descriptor, a histogram of visual word occurrences can be representative of the image. A histogram for each image is created and any two similar images should result in similar histograms as the difference in visual word occurrence should be minimal. Histograms can be compared using a histogram intersection, allowing the similarities and differences of any two images to be found. A collection of visual words is referred to as a visual vocabulary.

Implementation process for *build_vocabulary* function:
1) Loop through the data set of images, reading in the current image on each loop iteration.
2) Convert the current image to a grayscale image.
3) Convert the grayscale image data to single precision using the *single* function.
4) Obtain the SIFT features using the *vl_dsift* function.
5) Store these features within a matrix – concatenating on each loop iteration.
6) Obtain the centroids of each word using the *vl_*means function.
7) Centres are then stored into the output variable (in this instance, titled *vocab*).

SIFT (greyscale/colour)

SIFT is an image feature extraction algorithm which aims to extract distinctive features before correctly matching said feature to others found within similar images. This algorithm is scale and rotation invariant, highly distinctive and robust to illumination and occlusion; this means that image perspective is not an issue, multiple images can be created regardless of feature object sizing and features can be found despite clutter or noise within an image. SIFT consists of the following steps/stages: scale-space peak detection (identifying/locating the peaks within the scale-space which is generated from versions of the same image with various smoothing filters applied), key-point localisation (locating the distinctive features accurately), orientation assignment (an

orientation value is assigned to each of the determined point of interest), key-point descriptor (given a point of interest, how would said point be described as a vector) and clustering (Similarity/difference of features represented in a space with the dimensions of scale, orientation, location - the maximum point within that space represents a similarity transform).

Implementation process for *get_bag_of_sifts* function:

1) Loop through the data set of images, reading in the current image on each loop iteration.
2) Convert image to grayscale if grayscale SIFT is desired.
3) Obtain SIFT features for each channel (single channel if grayscale image).
4) Determine the pairwise distances for each colour channel.
5) Find the minimum distances and create histogram with these values.
6) Normalise the resulting histogram

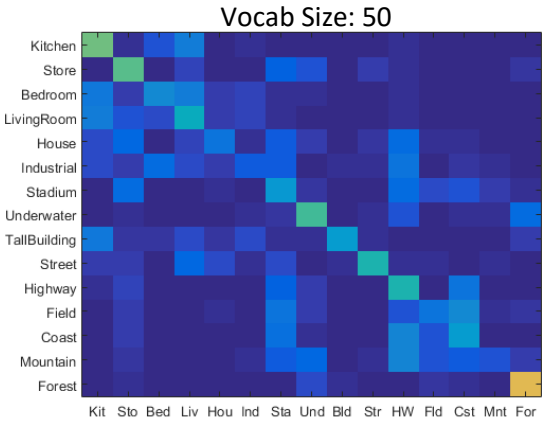
Spatial Pyramids

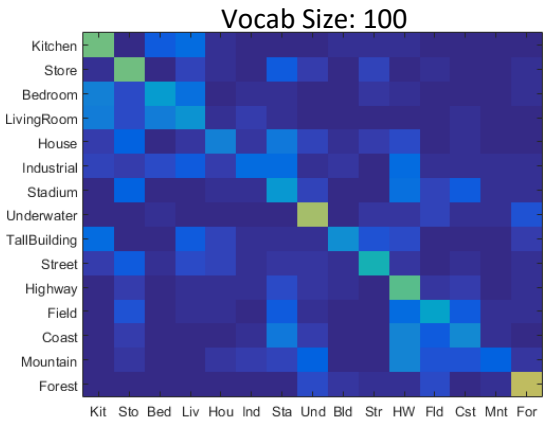
Spatial pyramids can be used to represent the scene of an image, it does this by splitting the image into different sections, a histogram for each section is to be computed and combined with the histogram of the whole image to create the feature vector. As the image is split up into sections, a pixels value will then be represented in the whole image and its corresponding subsections of the image. Therefore, by applying greater weights to these subsections it is possible to create a precedence which can be used to depict the scene of an image as it is known that the greater the weighting, the higher occurrence.

Testing & Evaluation

During this section the concepts and implementations discussed previously are to be tested and evaluated to determine which conditions produce the most accuracy within a reasonable execution time – accuracy throughout testing will be represented via confusion matrices. Testing will consist of two stages; the first stage, *initial testing,* will involve altering algorithm parameters whilst the second stage includes a comparison of algorithm combinations (image classification algorithms & image feature extraction algorithms) in addition to various colour models.

Vocabulary Size

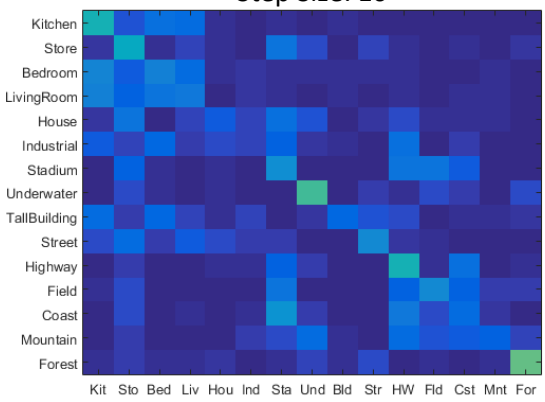




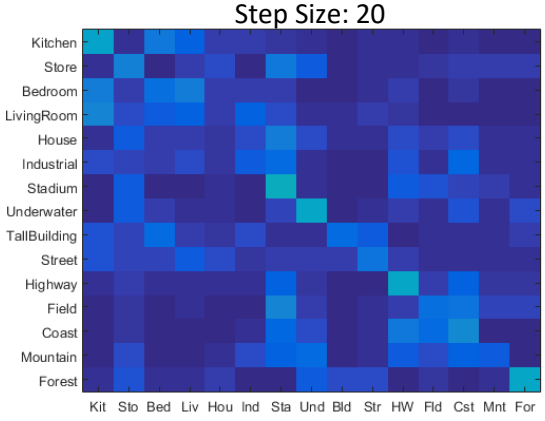
As shown in the figures above, altering the vocabulary from 50 to 100 increases the accuracy (from 0.376 to 0.398) alongside computational time of execution. The additional computation time required when executing the program with a vocabulary size of 100 is rather drastic considering the result is independent of any further parameter modifications; consequently, the vocabulary size will remain at a constant 50 for the remainder of the testing.

Bag-Of-SIFTS (Step Size)

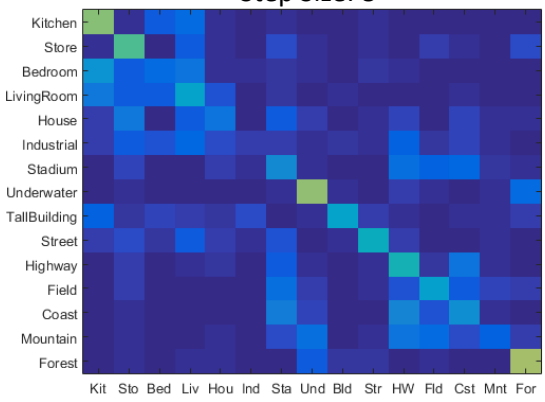
Step Size: 16



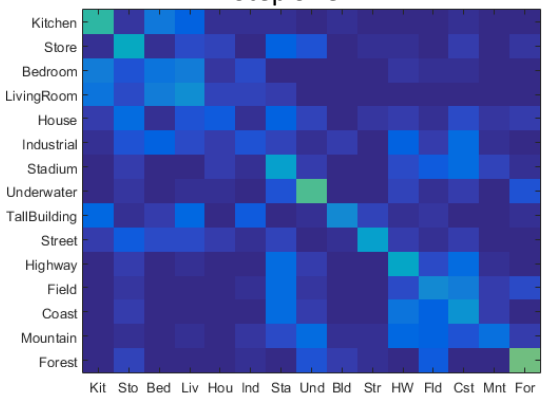
Step Size: 20



Step Size: 8



Step Size: 12

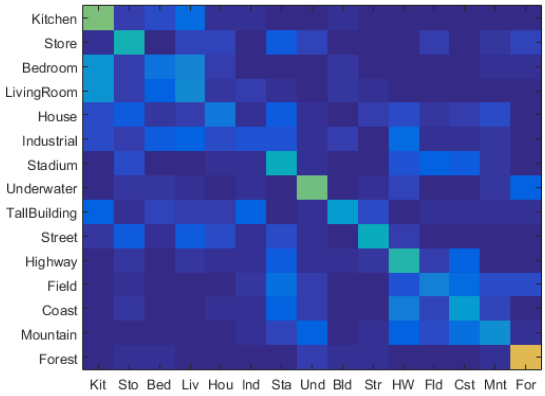


As shown in the figures above, setting the step size to 8 gives the most optimal result with an accuracy of 0.376. The accuracy results of the other step sizes tested are as follows: 12 (0.321), 16 (0.289), 20 (0.239).

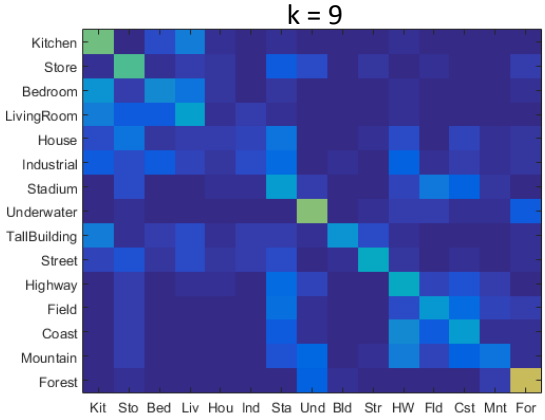
k-NN (k value)

As shown in the figures below, increasing the number of neighbours (*k)* increases the accuracy. Setting the value to high would increase in computation time would be to great for testing. The accuracy results are as follows: 15 (0.390), 12 (0.378), 9 (0.368), 6 (0.379).

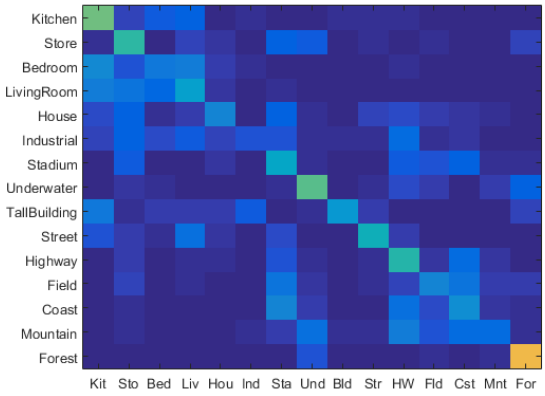
k = 6



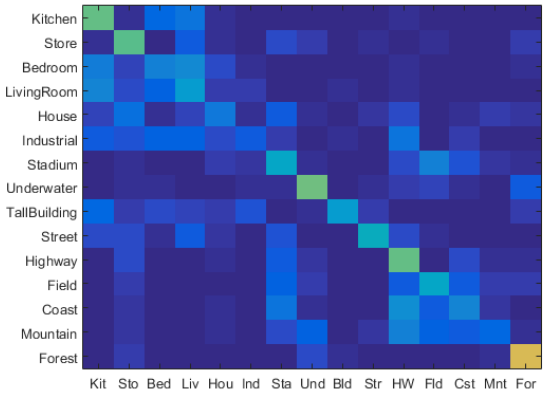
k = 9



k = 12

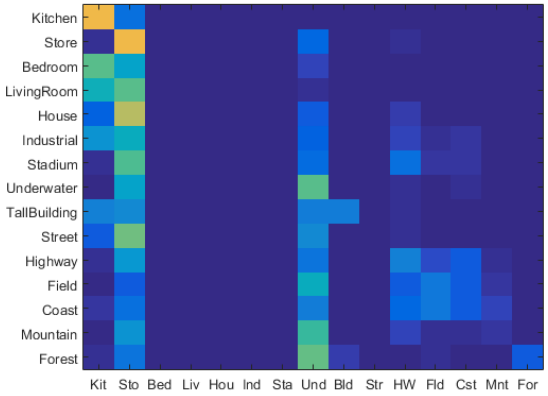


k = 15

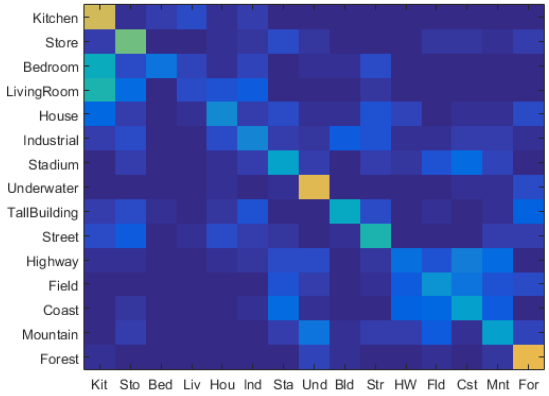


SVM (Lambda)

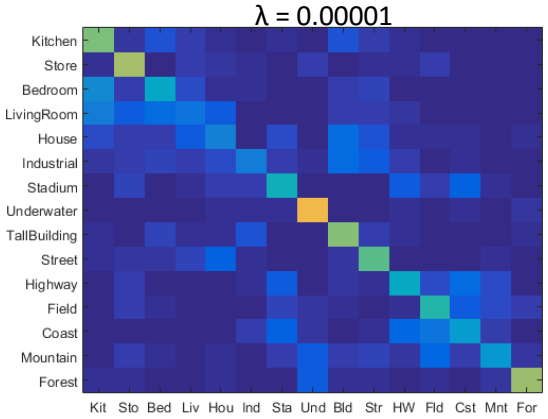
λ = 0.1



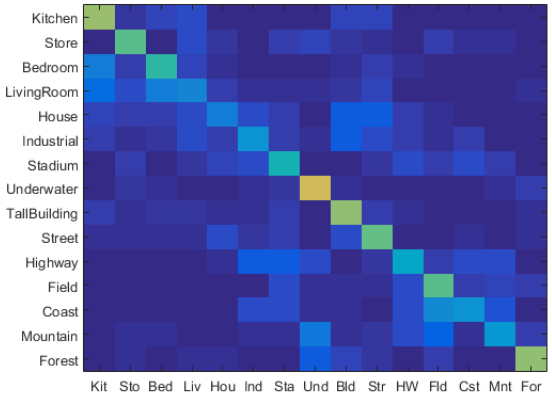
λ = 0.001



λ = 0.00001



λ = 0.0000001



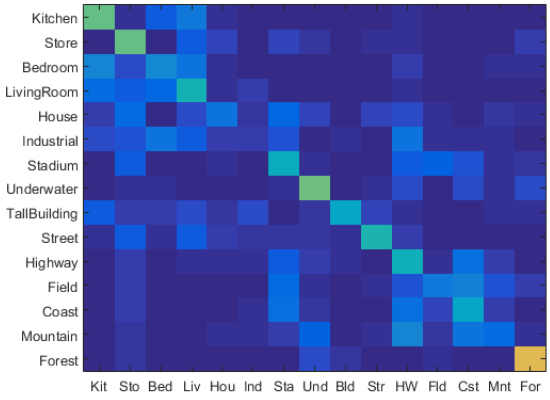
As shown in the figures above, the larger the value of lambda(λ) the worse the accuracy becomes – tests clearly show that a lambda value of 0.0000001 is most optimal giving an accuracy of 0.470. The other value results are as follows: 0.00001 (0.463), 0.001 (0.407), 0.1(0.209).

In conclusion of initial testing the following values will remain constant for the tests which follow:
Vocabulary Size = 50, Step Size = 8, k = 15, λ = 0.0000001

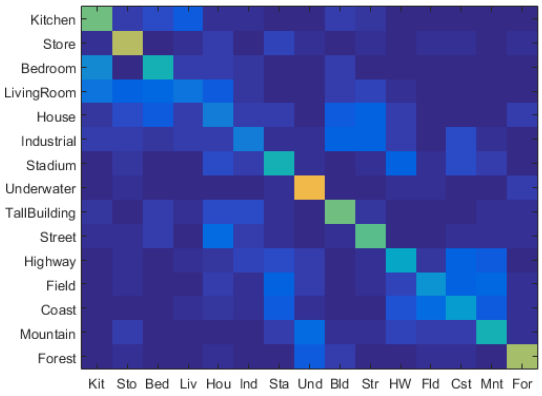
kNN/SVM & Bag of SIFT (grayscale)

As shown in the figures below, the combination of the SVM and SIFT(grayscale) results in a higher accuracy of 0.461 when compared to the combination of the kNN and SIFT(grayscale) which results in an accuracy of 0.394.

kNN + SIFT (Graysale)

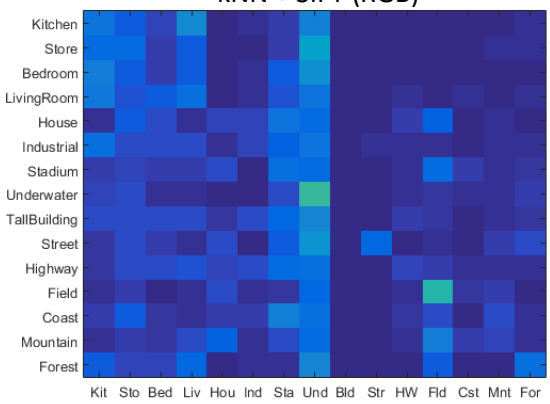


SVM + SIFT (Graysale)

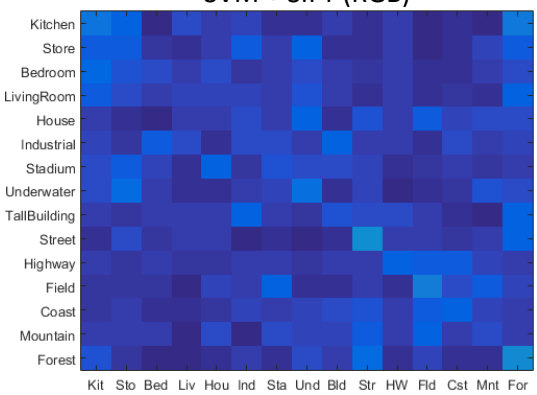


kNN/SVM & Bag of SIFT (RGB)

kNN + SIFT (RGB)



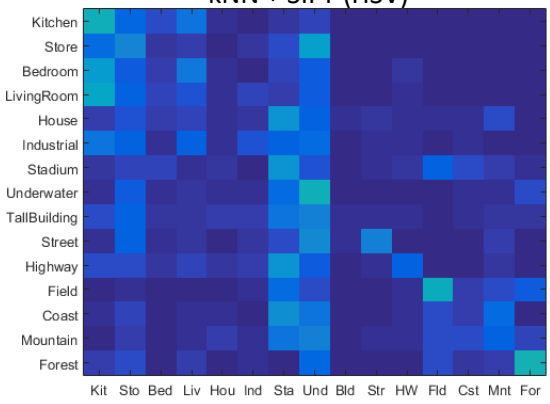
SVM + SIFT (RGB)



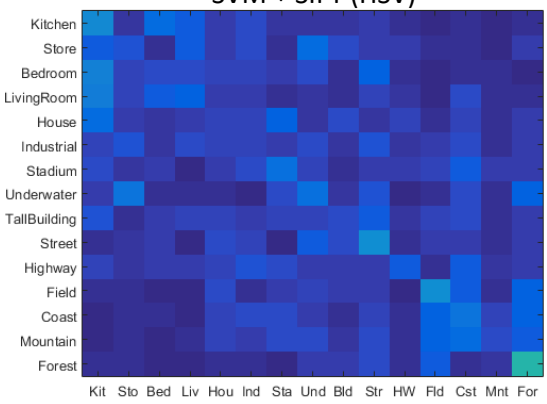
As represented by the figures above, when combining the kNN image classification algorithm with the colour (RGB) bag of SIFTS it results in an accuracy of 0.211. Combining the SVM classifier with the colour (RGB) bag of SIFTS results in a lesser accuracy of 0.175.

kNN/SVM & Bag of SIFT (HSV)

kNN + SIFT (HSV)



SVM + SIFT (HSV)



The confusion matrixes shown directly above represent the resulting accuracy when combining the kNN classifer with the colour (HSV) bag of SIFTS (giving an accuracy of 0.160) in comparison to the combination of the SVM classifier with the colour (HSV) bag of SIFTS (giving an accuracy of 0.142).

kNN vs. SVM

When comparing the accuracy results of the two different image classification algorithms kNN and SVM, it reflects that certain algorithms are more suitable to specified conditions in order to provide optimal outputs. In this instance, SVM proved to provide the highest accuracy value when incorporating the bag of features with grayscale imagery. However, when incorporating the bag of feature with a different colour model, (in this instance HSV and RGB colour models were used) the kNN classifier provided the highest accuracy. Lastly, it is important to note that the variation of performance between the classifications was most prominent during the processing of grayscale images – with minimal variance in regards to the HSV and RGB colour models.

Further Development

Development of the software could be taken further, extending the functionality of the system to include a wider variety of possible algorithms. Firstly, implementing the spatial pyramids which were discussed previously would allow for pixels with the greatest weighting/occurrence rate to be presented. Additional Image feature extraction algorithms such as the *Histogram of Gradient Orientations* (also referred to as HOG) can also be introduced; implementing would allow for testing to be carried out which would reflect the difference in accuracy when processing a whole image in comparison to the use of a sliding window where only a region of the image is processed at a time. More specifically, HOG computes sections of an image before calculating the sum of gradients over each pixel within an image.

A method of implementing Histogram of Gradient Orientations:
1) Divide the image into smaller regions to be processed individually.
2) Using a centred the gradient is calculated which is applied horizontally/vertically in gradient.
3) Colour images require all 3 channels to be calculated - using channel with the highest magnitude.
4) The relevant histogram bin is proportionally incremented by each pixels gradient orientation to the gradient magnitude.
5) A block is created, consisting of 2-3 cells. The whole image is represented via blocks.
6) Normalisation is performed within each block.
7) Histogram features can be created from the blocks, producing a feature vector which can be used for classification.

Subspace methods could also be introduced into the system, an example of these methods would be *principle component analysis* (PCA) and *linear discriminative analysis* (LDA). Both methods explore that images can be represented as a subspace of the original vector. This means that to distinguish the classes among the discriminant mode is simpler, whilst discriminating the class via the principle component becomes more complex.

Conclusion

The system which has been developed incorporates various algorithms, both image classification algorithms and image feature extraction algorithms. The accuracies achieved by the implemented algorithms are of a decent standard when analysing with the grayscale colour model; accuracies to drop significantly when using the two implemented colour models RGB and HSV. Implementing the ideals stated in the Further Development section would allow for a more reliable comparison between algorithms, providing additional detail into which conditions are most suited, in relation to colour model, to produce the most optimal accuracy results.

References

[1] Cunningham, P. and Delany, S.J., 2007. k-Nearest neighbour classifiers. Multiple Classifier Systems, pp.1-17.

[2] Peterson, Leif E. "K-nearest neighbor." Scholarpedia 4.2 (2009): 1883.

[3] Lazebnik, S., Schmid, C. and Ponce, J., 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on (Vol. 2, pp. 2169-2178). IEEE.

[4] Shah, D. (2016). *Computer Vision*. [online] crcv.ucf.edu. Available at: http://crcv.ucf.edu/courses/CAP5415/Fall2012/Lecture-5-SIFT.pdf [Accessed 17 May 2016].

[5] Statsoft.com. (2016). *Support Vector Machines (SVM)*. [online] Available at: http://www.statsoft.com/Textbook/Support-Vector-Machines [Accessed 20 May 2016].

[6] Lopes, A.P., de Avila, S.E., Peixoto, A.N., Oliveira, R.S. and de A Araujo, A., 2009, August. A bag-of-features approach based on hue-sift descriptor for nude detection. In *Signal Processing Conference, 2009 17th European*(pp. 1552-1556). IEEE

[7] Hmeidi, I., Hawashin, B. and El-Qawasmeh, E., 2008. Performance of KNN and SVM classifiers on full word Arabic articles. *Advanced Engineering Informatics*, *22*(1), pp.106-111.

[8] Lazebnik, S., Schmid, C. and Ponce, J., 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In*Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on* (Vol. 2, pp. 2169-2178). IEEE.

[9] Wei-Xue Liu, Jian Hou, and Hamid Reza Karimi, “Research on Vocabulary Sizes and Codebook Universality,”Abstract and Applied Analysis, vol. 2014, Article ID 697245, 7 pages, 2014. doi:10.1155/2014/697245

[10] Ling.upenn.edu. (2016). *Subspace Methods*. [online] Available at: http://www.ling.upenn.edu/courses/ling525/subspace_methods.html [Accessed 20 May 2016].

[11] Rosebrock, A. (2014). *Histogram of Oriented Gradients and Object Detection - PyImageSearch*. [online] PyImageSearch. Available at: http://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/ [Accessed 20 May 2016].

[12] Lyons, M.J., Budynek, J. and Akamatsu, S., 1999. Automatic classification of single facial images. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (12), pp.1357-1362.

[13] MartõÂnez, A. and Kak, A. (2016). *PCA versus LDA*. 1st ed. [ebook] Available at: http://www2.ece.ohio-state.edu/~aleix/pami01.pdf [Accessed 20 May 2016].