BSc Computing and Artificial Intelligence
Third Year Project Report

Fast Retrieval for Image Organisation Assistant

Hamish I A Morgan
Candidate Number: 56405

Supervisor: Dr. David Young

2010

School of Informatics
University of Sussex
Falmer, Brighton
BN1 9RF, UK

Statement of Originality

This report is submitted as part requirement for the degree of BSc Computing and Artificial Intelligence at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

Hamish I A Morgan – 29^{th} April 2010

Summary

This report details the design and implementation of a system that enables a user to be assisted in the organisation of their image collection. The solution was achieved using a fast image retrieval paradigm, building on the work of David Nistér, Josef Sivic, and David Lowe (Nister and Stewenius, 2006; Sivic and Zisserman, 2003; Lowe, 1999). In this paradigm images are pre-processes to single fixed length vectors, called the image *signature*s. Comparing two signatures directly gives a measure of similarity between the images. Thus, order can be applied to a set of images based on their similarity to some query image. The user is assisted in the organisation of their images, by suggestions images that are similar to those that have already been labelled.

The project extends the idea of single image queries, by allowing queries to be composed of multiple images signatures. This enables querying based on a subset of the image database, along arbitrary criteria decided by the user. Having given several images the same label, the user is provided with recommendations based on commonality in those images. Recommendations improve as more images are labelled, without the need for classification.

The images are pre-processed first, by finding and describing invariant key-points in the image, using the popular Scale Invariant Feature Transform (SIFT) algorithm. (Lowe, 1999) The key-point *descriptors* are then quantised using the Hierarchical K-Means algorithm. Pushing the descriptors of an image onto the resultant cluster tree produces a histogram of the occurrences on each node. This histogram is then normalised and weighted to produce an image signature.

The prototype was implemented and evaluated in the Matlab. There is no user interface because work on this would have distracted from the central research goals of the project. User functionality can however be simulated by making calls to the various functions and methods in the Matlab interactive command environment.

Preliminary evaluation of the solution has been positive. Using a relatively simple data-set of pre-classified images, the retrieval accuracy was found to be approximately 75%, for a multiple image query. As the number of query images increases, so too does the accuracy. Several metrics of signature similarity where tried, and it was found that taking the inner-product of the signatures produced the best results. It was also found that the solution generalises well to unseen data. This indicates that once initialised, new images can be added to the databases without excessive computational overhead.

The work opens up numerous avenues of potential further study, in a new research area. It also demonstrates that the technology can be used to solve real world problems including assisted organisation of images.

Contents

1. Introduction

The aim of this project is to investigate computational systems and methods for assisting a user in organising their image collection. Advancement in technology has enabled the storage of large image collections on personal computers. These collections can be managed by software solutions such as iPhoto[1], Picasa[2], and online services such as Flickr[3]. These applications perform a range of tasks upon images including: reorientation, touch-up, album publication, and organisation. This work is interested in the latter task of organisation. In terms of an image collection this typically involves the grouping or labelling of images to facilitate retrieval (Rodden and Wood, 2003). Sadly the process of organisation is tedious and error prone. Therefore it is proposed that computational techniques can be employed to aid the user.

The *ultimate* goal is to create an end user application that assists with the task stated above. There are no existing solutions, so the primary objectives of the project are speculative and research orientated:

1. Analyse the problem areas to produce both functional and non-functional requirements for a solution.
2. Conduct a survey of relevant research in this area. This should include attaining image data-sets for the evaluation of prototype solutions.
3. Designs and implement one or more prototypes that may fulfil the requirements.
4. Evaluate the solutions according to the requirements.

The structure of the report is as follow: Chapter 2 gives an overview of ethical and professional considerations that have been relevant to the project. Chapter 3 presents the research done with an overview of key foundations to the project. It also lists a number of functional and non–functional requirements for the prototype, and for a finished product software. Chapter 4 outlines a high level design for the software, including step by step details of user interaction. Chapter 5 details key components of the system including the feature detection algorithm, clustering, similarity metrics, and a measure of retrieval accuracy. Chapter 6 gives a brief overview of the prototype implementation; including the directory structure and the purpose of each function. Chapter 7 evaluates the prototype in terms of accuracy and computational performance. Various parameters of the systems are varied here to measure their effect, and to find an optimal configuration. Chapter 8 outlines areas of future work. The report is concluded in Chapter 9. The appendix contains only the project log; source code is provided by digital submission.

[1] iPhoto is part of the iLife software pack from Apple Inc. It is both available for purchase and pre-installed on all new Apple Macintosh computers. More information can be found at http://www.apple.com/ilife/iphoto/ (Last visited 5^{th} December 2009). Version 8 has a face recognition and classification system.

[2] Picasa, by Google, is a free image organisation program that can be download from http://picasa.google.co.uk/ (Last visited 5^{th} December 2009).

[3] Flickr is an online image organisation and publication website from Yahoo that has both a free and paid subscription version. It is available at http://www.flickr.com/(Last visited 5^{th} December 2009).

1.1. Conventions

The report contains some mathematical equations. The following conventions are used throughout.

- Scalar values are expressed in lower-case italic characters: $v, x, \alpha, \sigma, \mu$.
- Vectors are bold face: $\boldsymbol{s}, \boldsymbol{v} = (v_1, v_2, v_3)$.
- Matrices are denoted by upper-case characters in a calligraphic font: $\mathcal{A}, \mathcal{B}, \mathcal{X}$.
- Sets are denoted with upper-case italic serif $S = \{s_1, s_2, \ldots, s_n\}$.
- Set cardinality is denoted $|S|$, which is the same as scalar absolute value $|a - b|$, but the meaning is disambiguated by context.

2. Professional Considerations

There are ethical issues that have been taken into consideration during this project. In particular the British Computing Society's Code of Conduct for BCS Membership (BCS, 2006) and Code of Good Practise (BCS, 2004) detail a number of areas that have been relevant, including:

- The project has used libraries, source code, and data provided by third parties. In line with the BCS (2006) Section 3 intellectual property laws, both local and foreign, have been adhered to in the use of third party material.
- A dissertation project has a strict time-table of deadlines in which work must be completed. It was, therefore, imperative that the project be completed on time, and within any budgetary constraints. This is not unlike professional requirements as stated in the BCS (2006) Section 7.
- As a research orientated project it is particularly important that unbiased evaluation and study be carried out, and that the performance of prototype implementations are not unfairly represented, in line with BCS (2006) Section 9.
- Although the project has been produced by a single author, there have been cases where other professionals are consulted for their experience and knowledge. For example meetings with the project supervisor. It is important that all such interactions have been carried out with integrity. (BCS, 2006, Section 11)
- The field of Artificial Intelligence (AI) is broad and developing quickly. It has therefore been important to maintain awareness of technological developments when conducting research. (BCS, 2006, Section 13)
- Selecting a project that is both challenging and feasible is not an easy task. BCS (2006) Section 15 states the importance of not exaggerating competence in the field, and only attempting work which achievable. While the *ultimate* goals of this project are largely beyond the scope of a third year project, this has not been misrepresented in any way.

3. Background Analysis

This section gives an overview and investigation into the task. Relevant previous research is noted, and defines acceptance criteria for the project.

3.1. The Task

The core task addressed in this project is to assist a user in the organisation of their digital image collection. One could look further back in the requirements and ask "Why does the collection need to be organised?", but since this is an AI project, not a Human Computer Interaction (HCI) project, it was felt that the necessity of organisation should be taken for-granted. There is some HCI work on comparing various enhancements to image organisation software with inconclusive results, (Rodden and Wood, 2003) but their failure may just be a product of the techniques used.

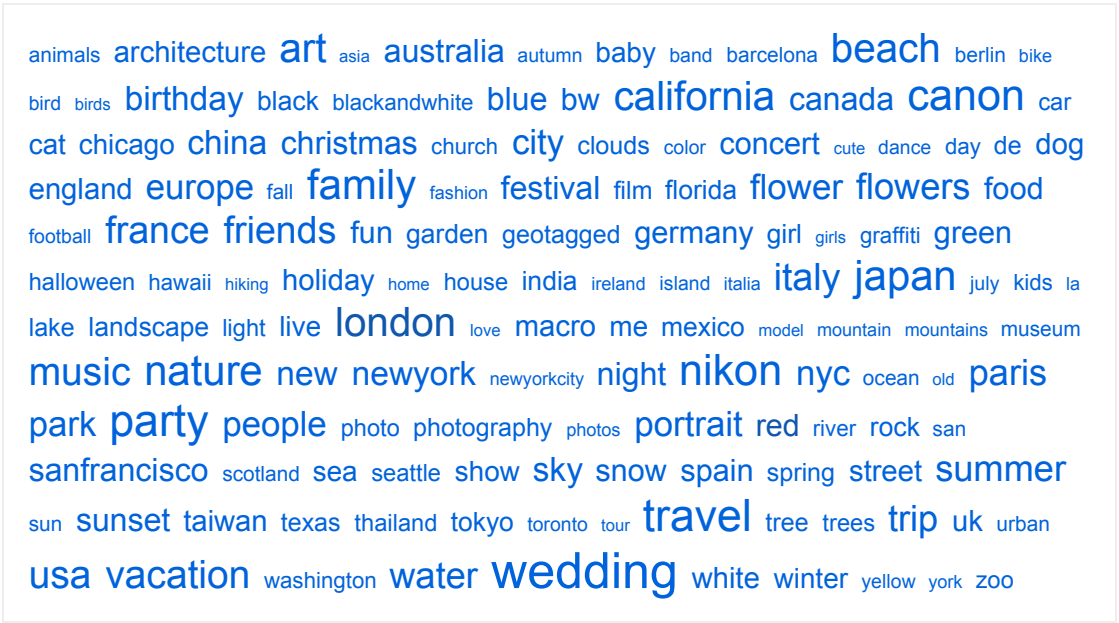

Figure 3.1.1.: A "tag cloud" of the all time most populars tags used on, the photo sharing website, Flickr. The font size is proportional to term frequency. Available at http://www.flickr.com/photos/tags/ (Last visited 5^{th} December 2009).

The purpose of this organisation was presumed to be enabling retrieval according to user specific criteria, since this has been an active area of research for some years (Bradshaw, 2000). These criteria are highly varied; as Flickr's "most popular tags" demonstrates (see Figure 3.1.1). *Tags* are the equivalent of category labels or keywords used to organise the images. The list includes locations, geographic features, animals, events, colours, and objects. An important observation of this list is that the terms are not disjoint or

structured. It is likely that an ontological approach to modelling this organisation would fail because classes are not concrete, relationships are highly variable, and meaning is subjective.

It is not without reason that the term "organisation" has been used thus far to describe the intended process. There are other terms that are far more prevalent in the literature: classification, and categorisation. Often they are used synonymously but from an information theoretic perspective they have significant differences: *Categorisation* is the arrangement of items based on perceived similarity into groups with indistinct boundaries, while *classification* is a systematic approach of arranging items into disjoint classes based on necessary and sufficient characteristics (Jacob, 2004). A user engaged in the organisation of their image collection is *un*likely to be performing formal classification, using objective symbolism. Rather they *categorise* to enable subjective retrieval of images in groups that are meaningful to them. Therefore, this project will not follow the standard Computer Vision literature in assuming that organisation is a classification problem, but will instead view it first as categorisation problem (although later that notion will be also be questioned).

In the Computer Vision literature, there are a number computationally similar areas. These are detection, recognition, and retrieval (Csurka et al., 2004), along with matching and classification. Classification, as defined above, is the arrangement of entities into formal classes. It depends on the properties of the image being found with *detection*. *Matching* is the processes of confirming or denying the existence of something based on a prescribed template. For example: To confirm the existence of a circle in an image, the application of a generalised Hough Transform algorithm (Duda and Hart, 1972) would be matching. In contrast *recognition* usually involves the system learning a shape from the features of training data. Finally *retrieval* is searching over and within documents to find relevant results based upon a query term or terms. Typically, effective retrieval is dependant upon a robust organisational structure of the documents such that an exhaustive search of the database is not required.

As stated previously, the aim of this project is to assist the user in the organisation of their image collection. Since there is no effective way to know in advance along which criteria the user will perform this organisation, the criteria must be learnt. It is assumed then that, in order to function, the user must have first organised some of their image collection to act as a training set. One way of approaching this problem is as classification. We use the training samples to build a classifier, which is then applied to the unclassified samples. The primary problem here is the domain of image organisation software. A user will manually label some images, and later may label some more. The classifier must be retrained each time the user labels and image. Further more, the labels are not classes in the strictest sense because they are not disjoint. This means that a classifier will have to be trained (and retrained) for each label. Because of these problems a better way to approach the task is to find some representation of an image from which similarity can be easily and directly measured. Using this representation, labels can be suggested simply by measuring the distance between images with a particular label, and images without it. The closest images can be proposed as also having that label.

We can express the problem without needing to know what a label *is*, other than say it is an abstract symbol attached to some group of images. The result is that we can treat assisted classification as a retrieval problem; we search our database for the images that correspond most closely to the query set.

3.2. Requirements

It is important to consider the project from a software engineering perspective, even if it is speculative research. This is because the practicalities of end user software should inform the design. This section outlines the acceptance criteria against which the prototype software can judged.

3.2.1. Functional Requirements

The following is a list of criteria under which the success of any proposed end-user application can be evaluated. It is worth re-iterating, however, that this is a research orientated project so an end-user application may not be build. Never the less, the algorithms used should bear in mind the following:

1. The system should allow a database of images to be viewed.
2. The system should allow images to be labelled with textual descriptors. Each image can have zero or more labels. Each label is defined by the images it holds. Therefore a label can be applied to one or more images.
3. By selecting a particular label, the system should allow the user to view the images that have that label.
4. The system should be able to make suggestions about what images are most likely to be added to the group denoted by a label.
5. The system should allow suggestions to be confirmed by the user and add them to the group. Conversely suggestions can be denied, removing them from the result set.
6. The system should cope with as wide a variety of labels as possible, and be able to effectively retrieve images based on these labels.

3.2.2. Non-functional Requirements

There are a number of requirements, in addition to those given above, under which the operation of any final system should be judged:

1. The system should run on a modern personal computer. The hardware should be something equivalent to the following: 2 GHz processor, 1 GB RAM (Random Access Memory), 80 GB Hard-Disk space.
2. The retrieval (suggestion) process should take less than a second for an average sized (≈ 1000) image databases. Ideally the time taken should be asymptotically invariant to the images sizes, and bounded by $O(n \log n)$, where n is the number of images.
3. Pre-retrieval computation must also be fairly fast. This includes initialising a new database and adding a new label or image to the database. However, this could be a task that runs in the background, so it is not a critical requirement. It is hoped that the pre-retrieval computation will take less that 1 minute per average size image ($\approx 1024 \times 768px$), and that adding an image or label will not require regeneration of the database. In addition the asymptotic running time of building a database of n is the number of images should be bounded by $\Omega(n)$ and $O(n^2)$.
4. Image databases typically consume a large amount of hard-disk space. Because of this meta-data used to organise the images may be quite large without being significant.

The organisational meta-data will need to be wholly or partly loaded into RAM, and since RAM is typically much more limited than disk-space, the disk-space limitations should be governed by RAM limitations.

5. RAM usage should be controlled. For example it would be entirely unfeasible to hold an entire image database in RAM. It may be necessary to hold image meta-data used for indexing in memory, and this should not exceed 300 MB for a typical database of 10k images. Therefore we should expect to use about:

$$\frac{200 \times 2^{20}}{10000} \text{ Bytes} \approx 30 \text{ KB per image}$$

However, if some system was implemented, where by not all of the meta-data was required for fast search, then this figure could be increased. In any case we can state the upper bound of memory usage asymptotically must be $O(n)$, where n is the number of images.

6. The software should be platform and architecture independent. However, this is a prototype phase, and we can limit it to a single platform.

7. Any suitable programming language could be used the meets the previous function and non-functional requirements. For the prototype a mixture of Matlab, C, and Java has been used. The prototype has primarily been written in Matlab, but with certain library calls from C and Java where Matlab functionality was not available.

3.3. Software Interface

Although this project is largely concerned with the algorithmic feasibility of a solution to assisted organisation and image retrieval, it is felt that some overview of the user interface may be helpful.

3.3.1. Apple iPhoto Faces

Apple iPhoto gave some inspiration to this project with the release of version 8, in which they added the "Faces" feature. Faces allows you to organise your photographs by the people they contain. The user annotates images with labels to denote the named persons face, visible there. (See Figure 3.3.1) Once marked-up in this way the collection can be browsed by person (see Figure 3.3.2). This in itself is not such an extraordinary feature, and is present in many image collection applications, but iPhoto augments this by attempting to learn the faces from the images you have previously denoted. As can be seen in Figure 3.3.3, when you view the photo's a particular person, iPhoto will make suggestions, at the bottom, for faces that may also be this person. It is the intention of this project to work towards building something like iPhoto faces, but for general categorisation.

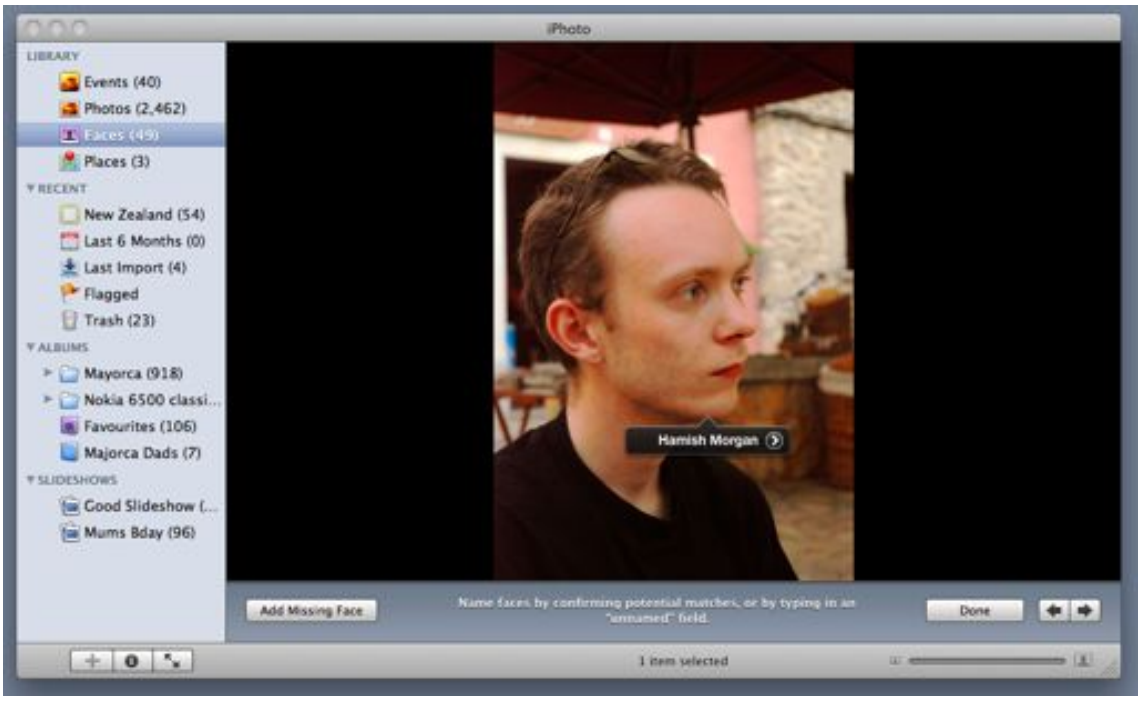

Figure 3.3.1.: Screenshot of Apple iPhoto showing how faces can be manually named by the user.

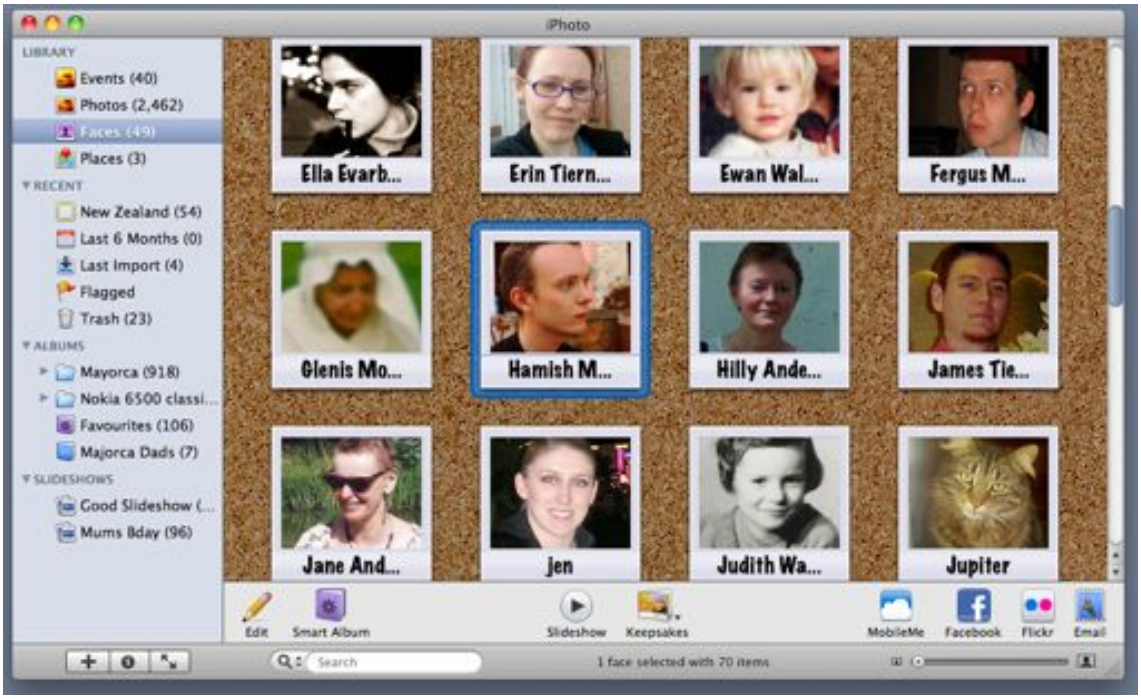

Figure 3.3.2.: Screenshot of Apple iPhoto showing that once faces have been named, you can browse your photos by person.

Figure 3.3.3.: Screenshot of Apple iPhoto show the contents of faces identified as "Hamish Morgan". Notice at the bottom iPhoto is suggesting other photos it thinks have a similar face.

3.3.2. Interface Design

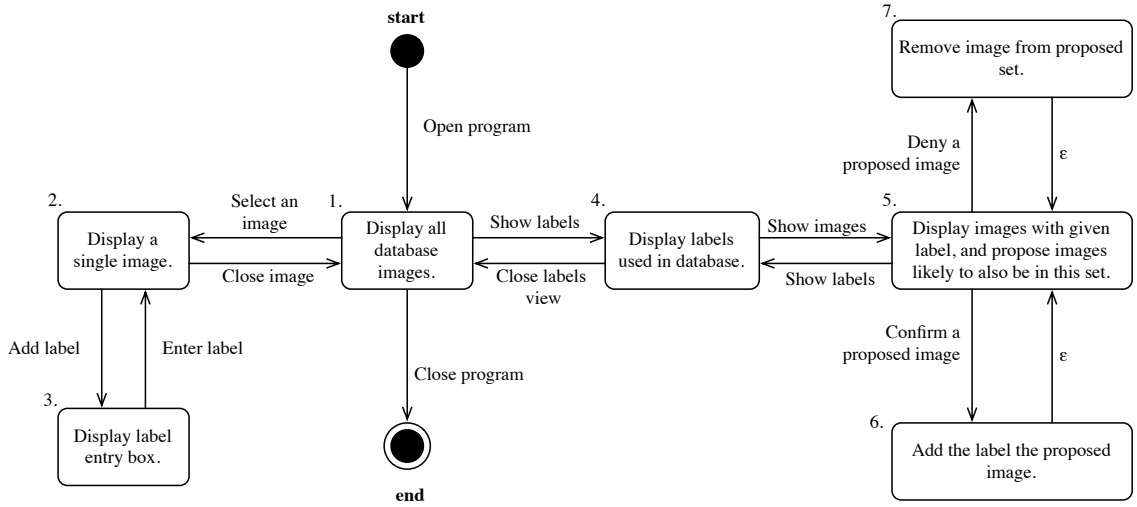

Figure 3.3.4.: A state graph showing the basic interaction possible from the user interface. Edges represent actions by the user, vertexes are the state of system after the action has been performed.

The interface design is somewhat described in the functional requirements (Section 3.2.1), but it was felt that some elaboration would be helpful. Figure 3.3.4 shows a interface state diagram, and denotes basic interactions from the the users perspective.

3.4. Image Processing

Since the seminal book Vision by David Marr (1982), computer vision systems have been modelled as a pipeline of processing modules that transform a 2D image into some high level representation. In the case of Marr this representation was a 3D model, but more resent research has moved away from this as a target model. The pipeline paradigm makes sense because it breaks down a highly complex process into more manageable chunks. It has roots in algorithmics but Marr and others started to show how it was neurologically plausible, and in particular how it could be used to model biological vision systems.

3.4.1. Processing Pipeline

A major limitation of machine learning techniques is the kinds of data regularities they can represent. Each technique offers a bias towards a particular kind of structure, and will perform poorly if the regularity is not well represented by that structure. For example K-Nearest-Neighbour, and K-Means represent clustering structure and will perform well on data arranged in clumps. A more powerful (weaker biased) technique such as Multi-Layer-Perceptrons (MLPs) can represent complex area based structure, but are harder to train and will perform poorer than K-Means on clump structure. Images are in the worst class of data, from a machine learning perspective, because their regularities are expressed through relational structure. The hue and brightness of a particular pixel is utterly meaningless on its own in expressing the subject of a photograph. Only through the combined information of pixels, in a particular configuration, can the subject be modelled. The relational structure necessitates some pre-processing that will make the relational structure of the data tractable.

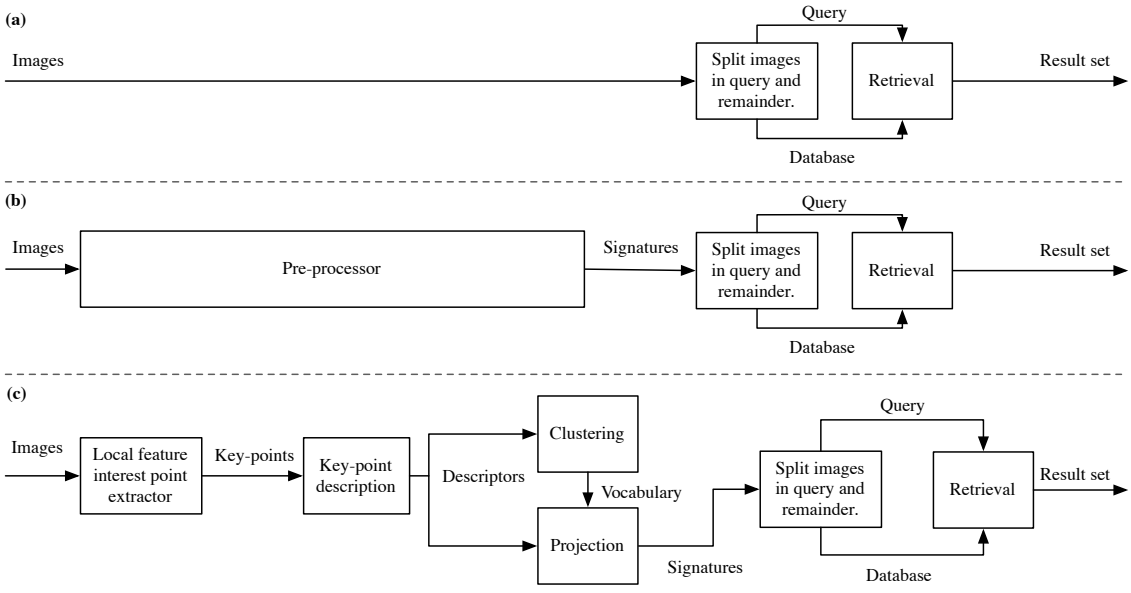

Figure 3.4.1.: Diagram showing the processing pipeline in various stages of expansion.

From the analysis of the previous section we can surmise that our system takes the form given in Figure 3.4.1a. The inputs are two sets of images; a query set and its complement (all the other images in the database.) The output is a result set of images suggested to be similar to those in the query.

How the "Retrieval" box works is unclear; somehow it is supposed to correlate pixels in the raw images to find matches. A reasonable suggestion is to reduce each image down to a fixed length vector of features that describe it in some way. If we have a feature vector for each image, then we can measure the distance between them as their degree of similarity. We shall call the feature vector of an image its *signature*. This expansion is show in Figure 3.4.1b. Now we have a Pre-processor, which computes the signature of each image, and a Retrieval system.

The work of David Nistér and others (Nister and Stewenius, 2006, Sivic and Zisserman, 2003) has inspired the pipeline design to be expanded to that given in Figure 3.4.1c. Here the pre-processor is implemented as follows: First, interest points are found in each image; typically these are corners or the centroids of regions. These key-points are then described in some way such that they can be compared with other key-points. The result of this is a set of fixed length descriptor vectors for each image. Unfortunately the number of descriptors per image, and their position, is variable. This makes it hard to compare the similarity of images. To resolve this the descriptors, of all the images, are clustered into a fixed number of bins to create a *visual vocabulary*. This is known as a Bag-of-Features approach since we are ignoring the interrelationships of features.

If we project the descriptors of a particular image on this vocabulary, and note the usage of each bin, we can produce a *signature* for each image. The signature is a fixed length vector (the number of bins in the vocabulary) and allows images to be compared in terms of signature similarity. This basic design is the foundation of the project.

3.4.2. Feature Extraction

The diagram in Figure 3.4.1a shows the need of some algorithm for the extraction and description of image features. There are two major classes of image feature: Scene features are measurements of global properties, such as the over-all brightness, or the size of the image. They could also include more complex information like an approximation of the vanishing point or orientation. Local features are key-points in the image. They denote interest in locations that are indicative of local structure, such as corners or the centroids of regions. It is possible to incorporate both local and scene features in a computer vision system, but the Bag-of-Features paradigm being used deals primarily with local features. There are many algorithms for the purpose of local feature extraction, two of which are described briefly below.

3.4.2.1. Scale Invariant Feature Transform

The Scale Invariant Feature Transform (SIFT) algorithm was proposed by David Lowe (Lowe, 1999; 2004) as a means of finding and describing features in a way that allows them to be matched robustly between images. To achieve this the descriptors are invariant to rotation, scale, affine-transform, and partially invariant to illumination. One of the main limitations of SIFT is that it can be quite slow, depending on the size of the image. Another limitation is that it is subject to a patent[1] so may not be deployed in commercial software. See Section 5.1 for a detailed description.

[1]The SIFT algorithm is patented under "Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image" David G. Lowe, US Patent 6,711,293 (March 23, 2004). Provisional application filed March 8, 1999. Assignee: The University of British Columbia.

3.4.2.2. Speeded-Up Robust Features

As the name implies, Speeded-Up Robust Features (SURF) is designed to be faster than SIFT. Research has shown that it can perform better than SIFT , while maintaining its speed advantage and using less memory. (Bay, Tuytelaars and Van Gool, 2006; 2008) Another benefit of SURF over SIFT is that it is not subject to a patent.

3.5. Evaluation

This work is largely research orientated, and as such there is a need for evaluation of the various methods tried. Potential criteria of evaluation fall into three main categories: User Study, Acceptance, and Accuracy.

Although accuracy is the most obvious metric of success, another would be whether or not the application actually saved the user time. To assess this, a study could be carried out, the results of which would give an indicator of real world performance. This will not be attempted during this project due to temporal economy, but it is important to mention.

The acceptance is a matter of confirming that functional and non-function requirements have been met as stated in Section 3.2.

Figure 3.5.1.: Examples of images from the Caltech-256 data set.

To test prototype methods for accuracy, operation can be compared against pre-classified data-sets. There are many such data sets available; ones explored during this project are listed below.

Caltech 6 A data set produced by researcher at the Computer Vision Group at Caltech (The California Institute of Technology). It consists of 6351 images divided into 6 categories: *aeroplanes*, *bottles*, *faces*, *motorbikes, cars*, and *leaves*. There is a 7^{th} special category of *background* containing miscellaneous images to provide noise. (Philip, Updike and Weber, 2001) Although rather limited it has been used extensively in classification research and so may be useful for comparison.

Caltech 101 A much larger data set containing 9248 images, in 101 object categories, plus a *background* category. (Fei-Fei, Fergus and Perona, 2004) Unfortunately there can be as few as 40 images in a single category which is rather limited. The software works by taking a number of images as a query and returning similar images. For a query size of 10 there will be only 30 measurable matches, which is insufficient.

Caltech 256 The largest of the Caltech data-sets containing 30867 images in 256 object categories (Griffin, Holub and Perona, 2007). The minimum category size is now 80 making it preferable to Caltech 101. In addition the authors give details on the demonstrated tractability of each category. This should allow the selection of a simple sub-set for early prototypes, moving to more difficult subsets as the results become saturated. There is no known system that can effectively classify the full data-set.

LabelMe A collaborative project from the Computer Science and Artificial Intelligence Laboratory at The Massachusetts Institute of Technology (MIT). It aims to create a

gold-standard image data-set for object recognition (Russell et al., 2008). The project is still a work in progress but already contains hundreds of thousands of images.

Flickr This project is aiming towards providing an application to work on an end-users image collection, so it is plausible that a fairer representation of an average users images would be found on an image sharing website such as Flickr. Flickr could be accessed via its API[2] to retrieve groups of tagged images on which to evaluate a prototype implementation.

Buildings/People A final set that was used is a collection of 128 photographs, half of which have people in them. This is a highly simplistic data set, but had the advantage of being very small. During development the data must often be repeatedly re-processed and this is extremely time consuming. To alleviate the problem this small data set was employed.

Having found data-sets, some metric for measuring the accuracy of predictions is required. In typical classification this is simply a matter of counting the number of items that were incorrectly classified. In retrieval systems generally the input is one or more query terms, and the output is zero or more similar results. With this software, the query terms are whole images. The returned values are an ordered list of images, from most similar to least similar. Therefore we must construct a special error metric, described in Section 5.5.

In addition to direct accuracy measurement of query results, it is important to consider the performance of the software on unseen data. The image organisation software can be seen as a closed world, where there is no unseen data; but this rules out the ability to add new images to the system without rebuilding the database. If the software generalises well to unseen data then we can reduce the amount of computation required. Another reason to measure the accuracy of unseen data is that it warns when the model is being over-fit to data noise. The data can be know to be over-fit if the training error is low but generalisation error is high. This allows the model bias to be strengthened towards the regularities of the data that is actually descriptive.

Cross-validation is routinely employed to measure the generalisation error of a classification or regression algorithm. One common method of cross-validation called n-fold, splits the data into n partitions. The classifier is trained on n-1 partitions, then tested on the remaining partition. The process is repeated n times such that each partition has been used for testing once. The mean error is then calculated over all the training runs.

[2]The Flickr API (Application Programming Interface) allows non-commercial third-party applications to communicate with the Flickr infrastructure. It employs specially crafted XML requests that can retrieve data such as tags, photos, and user information. More information is available at http://www.flickr.com/services/api/ (Last visited 5^{th} *December* 2009)

4. High Level Design

The initial design was built around the idea of using a vocabulary tree, based on feature descriptors, to characterise images. In Sivic and Zisserman (2003) this technique is used for fast video frame retrieval. The basic pipeline was as follows: First calculate features and descriptors using the SIFT algorithm (described in Section 5.1). Next, quantise the descriptors of each image by K-Means clustering (described in Section 5.2). The resulting cluster centroids become the *vocabulary* of our image sets; descriptors clustered around the same mean can be said to be contextually similar. The descriptors of each image is projected on the vocabulary to produce a histogram of the frequency of vocabulary occurrences (visual words), called the image *signature*. The signatures are then weighted by the Inverse Document Frequency (IDF) for each visual word. This results in the retrieval weighting known as "Term Frequency - Inverse Document Frequency" or TF-IDF (described in Section 5.3). The signatures can now be used to query the database: Given the signature of a query image, similar images in the database can be quickly retrieved by finding the most similar signature to the query signature.

A critical advantage, with this system, is that once the vocabulary has been built for a database, no further learning techniques are required. Instead of having to re-classify every time the user changes a label, the images are pre-processed to such an extent that the signatures can be used directly as a measure of image similarity. Adding a label is purely a convenience to the user, who can then select all images with a particular label as a query to find other images that are similar. The only time learning may need to be repeated is when images are added or removed from the database. However, even here the system has been shown to generalise well to unseen images without rebuilding the database (see Chapter 7).

In Nister and Stewenius (2006) the above method was extended to be efficient on much larger data-sets. This was achieved by using Hierarchical K-Means clustering. This produces a vocabulary tree, rather than just clusters. Each node is associated with an *Inverted File* structure, that points back to images that had descriptors projected through the node. The result is that similar images can potentially be found much faster.

It is important to consider the assumption implicit in this design, some of which are known to be false but make the problem tractable. A major assumptions that underpins the representational models, along with almost every machine learning data model, is that abstract entities can be represented in a geometric space. Unlike physical objects in 3D space, the objects of this system have no natural spacial relationship. Making the assumption that they are nominal spacial entities allows their similarity to be a function of their pseudo-geometric distance. This assumption is made on both to the SIFT descriptors, and image signatures. In the case of SIFT descriptors we are making an even stronger assumption, that similar descriptors will form clusters. In other words, for the K-Means to be a correct quantisation tool, the data should already exists as distributions around a number of centroids. Further more these pre-existing clusters should hold descriptors that are nominally identical; their only difference being noise.

Another assumption implicit in this work is that a users labelling of images is based significantly on information held by and images. If an image is labelled based wholly on the

content of the image, then there is a high probability that suitable computational technique exists to model that regularity. Naturally this does not imply the techniques used in this project will work. A users labelling of images may be only partially based on it visual contents. They could instead label based on memories of events, or some phenomenological property of the image. Under these conditions a computational technique must have real world knowledge to interpret the regularities of the image. This is clearly outside the scope of the project and so it will be assumed that a user labels primarily based on the visual contents of the image.

The remainder of this chapter informally describes various operations of the prototype software, from a macro perspective.

4.1. Building the Database

The following describes key steps required to create an image database that can perform queries.

1. Instantiate a new empty database object, that will hold information about the images and the vocabulary tree. The object will provide methods for adding images, labels, building the vocabulary tree, and performing queries.
2. Add some images to the database, denoted by their file-system path. For each image path, information about the image is retrieved such as format and size.
3. Run the SIFT algorithm (described in Section 3.4.2.1) on each image in the database, and store the associated feature and descriptor vectors. This SIFT algorithm in described in detail in Section 5.1. For now the feature vector can be thought of as pixel positions in the image. Descriptor vectors are like feature contexts, they describe a patch around each feature in a semi-invariant way. There are a number of important parameters to SIFT we can consider here, but for simplicity we shall use parameters that have been demonstrated to work well in previous research. Optionally we could throw away the features vector (the positions) at this point to save memory, because it is no longer required.
4. Build the vocabulary tree by hierarchically clustering the descriptors into d layers of k bins. Here d and k are constants, and must the chosen before hand. Note that typically k is effectively the branching factor of the tree. The parameters will be critical in finding representational structure. If k or d is too low the vocabulary will saturate resulting in highly distinct descriptors being clustered together. If they are too high then the vocabulary will be sparse, resulting in similar descriptors being grouped separately.
5. Compute a weighting for each bin that normalises occurrences; so a vocabulary term with high occurrence is less indicative than one with low occurrence. This is akin to the Inverse Document Frequency (IDF) that is common in Computational Linguistics.
6. For each image in the database push its descriptors onto the vocabulary, and count the occurrences on each bin. Multiply these occurrence by the weights calculated in step 5. These weighted frequencies can be expressed as a vector that becomes the image signature. Previous research has demonstrated improved results by processing the signature further. Techniques include normalisation and quantisation. The latter is especially useful for the inverted-file performance (discussed in Section 5.2.3).

The database is now fully initialised and ready for queries. Note that we have not labelled any images yet, and do not need to do so, because labels are only incidentally related to the retrieval process. They are applied by the user for recording image groups, and validation purposes.

4.2. Querying

The following describes the steps required to query the database. Performed from a user perspective, it also includes some details of the computation processes. It assumes the database has been initialised as described in Section 4.1.

1. The user browses the images with the intention of grouping similar ones. Practically this would be with a GUI but can be simulated with method calls to the database object for the prototype implementation.
2. The user selects a number of images, and gives them all the label x. This is achieved by first adding the labels to the database object, and then assigning the label to each of the selected images. Note that here images and labels are a many-to-many relationship; an image can be assigned zero or labels and a label can be assigned to zero or more images.
3. The user performs a query on the database using all the images they have labelled x. They are presented with an list of images, ordered by their perceived similarity to the query images. Similarity is calculated using some metric of distance between image signatures. For each image in the query, the distance is computed to each image in the database. These distances are then summed to arrive at a single value for each image in the database. The images are ordered by this value (ascending in the case of literal geometric distance). There are many potential similarity metrics, and these will be investigated later.
4. The user can now view query results and evaluate them. Those that are correct can now also be labelled as x, those that aren't can be ignored. Note that the user will probably note view the entire result set, only the few highest ranked matches.

Here the user can repeat the above steps, with larger queries to locate less obvious matches. They can also perform the same operations on different labels until the image collection is organised fully.

4.3. Accuracy Estimation

In addition to proving user orientated functionality there needs to be some way to verify that the system works as intended, and gauge the effect of various parameters and configurations. In Machine Learning literature it is typical to talk about training and generalisation error. Training error indicates how well the model represents the regularities of the data used to build it. Generalisation error (sometimes called testing error) indicates how well the model represents unseen data. It is easy to build a model with very low training error that completely fails to generalise to unseen data. This can occur when the model has overfit the noise on the training data, or when the training data is a whole unrepresentative sample.

To calculate generalisation we shall use the n-fold Cross Validation technique. This involves randomly assigning each of the training samples to n partitions. The model is then trained

on $n-1$ partitions, and the error is calculated using the samples in the remaining partition. This process is repeated n times such that each partition is held out once. The final generalisation error is the mean of the error across each held out partition. In this work 5-fold cross-validation has been used.

The design of this system complicates the Cross-Validation process considerably because of the extended processing pipeline. The primary machine learning model to be validated is the hierarchical K-Means tree, but the inputs and outputs to k-means are very different to the inputs and output of the system. An error metric is defined in Section 5.5, which takes the inputs to the system as images, and the outputs as recommendation labels. The K-Means algorithm on the other hand takes the descriptor vectors for all images and produces a cluster centroid hierarchy. In addition we need to validate various parameters to the processing pipelines such as the SIFT algorithm, and signature distance metric. It is possible to compute both the testing and generalisation error for the network by performing the steps given in the following subsections.

There is a case to be made here that the generalisation error is not so important. The reason is the stated objective and method of this project. The image database is a closed world, where we are not going to be presented with any images that are not already in the database. Further, a formal classifier of the labels is not being build, so there is no need to retrain the model when labelling images. The only time the K-Means generality comes into question is when new images are added to the database, and here it may be acceptable to re-compute the tree. So the only good reason to calculate a generalisation error for the system will be if we need to use the vocabulary tree with unseen images, and this is an extension rather than a primary objective.

4.3.1. Training Error

The following assumes the steps described in Section 4.1 (Building the Database) have first been completed.

1. Label every image in some appropriate way. There should be at least two labels, and at least 20 images for every label. One way to achieve this is to use one of data sets, described in Section 3.5, such as Caltech-256. In this case the labels will be the image classes defined by the data set.
2. Queries can be performed with one or more images, so the query size must be decided for this error estimation run. Typical values for query size are 1, 2, 4, or 8 images.
3. For every image in the database run a query as defined in Section 4.2. If the query size is greater than one then add a random selection of images with the same label, to bring the query up to the correct size. For example: If the first query image has the label x, and query size is 4, then find 3 additional images that also have the label x. Compute the errors for each query, using the metric defined in Section 5.5.
4. Compute the mean error over all the queries, as the estimated training error of the system.

4.3.2. Generalisation Error

Generalisation error can be estimated using the following steps. As before it assumes the steps described in Section 4.1 (Building the Database) have been completed. However it is important that not all available images where used to build the vocabulary tree. A sub-sample must be held back.

1. Having build the vocabulary tree, add the held back images to the database. Calculate their signatures by projecting the SIFT descriptors onto the previously generated tree. Keep note of which images where held back.
2. Label every images and decide upon a query size, as described in Section 4.3.1 steps 2 and 2.
3. For every image that was held back when the vocabulary was build, query the databases and compute the error as described in Section 4.3.1 steps 3 and 4.
4. The mean error is the generalisation error because the images were not used for training the vocabulary.

5. Detailed Design

The previous chapter gave a broad overview of the systems operations. This chapter will give much more detail about some of the key components.

5.1. SIFT Algorithm

As stated earlier, the SIFT algorithm (Lowe, 1999) allows features of an image to be described such that they are invariant to rotation, scale, affine-transform, and partially invariant to illumination. With this invariance, features in one image can be compared to features in another similar image. The SIFT algorithm is defined in three separate stages: Feature identification, description, and matching. The three stages are largely independent and so it is possible to replace one or more stages if required. A brief outline of the algorithm follows.

5.1.1. Feature Identification

The first stage aims to find points in the image that are likely to re-occur in similar images. Here, a similar image is taken to be one that has undergone some change in rotation, scale, or illumination. An example of this is two photographs of an object, taken from different positions and under different lighting conditions. The features must have some measure of orientation so they can be aligned to match other features. Finally, because they must have some measure of scale. Feature identification is achieved through the following basic steps:

1. The image is grey scaled. For this project the images will also be sub-sampled if it is very large. Some of the images in Caltech-256 are as large as 7916×7916, which cause considerable performance problems. They also consume huge amounts of memory ($7916 \times 7916 \times 4$ Bytes per pixel $\approx$ 239MB). To resolve this images are scaled to be at most 1024 pixels on any dimension.
2. The image is repeatedly smoothed and sub-sampled, by some factor, to product an image pyramid. This step is actually combined with the next step for efficiency, but has been separated here for clarity. Nominally the sub-sampling factor is set to 0.5, so each level of the pyramid will be half the size of the previous level. The real value is somewhat more complicated, and beyond the scope of this brief overview.
3. Each image in the pyramid is convolved with a Different of Gaussian (DoG) mask. DoG is a fast approximation of the Laplacian of Gaussian (LoG) mask. This has the effect of emphasising brightness changes. Flat areas will have DoG values close to zero, while corners and edges will have strongly positive or negative values.
4. The interest points are found by searching the image pyramid for 3-dimensional, 27-way local minima and maxima. These are pixels that are the maximum or minimum of their 8 surrounding pixels, and the 9 pixels on both the preceding and proceeding layers in the pyramid. The interest points are then filtered using various techniques.

5. Finally, the orientation of each feature is measured as the dominant gradient of the area surrounding the pixel.

These steps produce a features vector, where each feature is composed of the row and column value of the interest point, the orientation, and the scale space (pyramid layer) it occurred on. There can be any number of features found in an image but typically it is the range 100 to 1000. The features are already invariant to rotation, scale, affine-transform, and illumination, but they do not give enough information about the point such that it can be differentiated from other points.

5.1.2. Description

To differentiate features they need to be described in some way that maintains invariance. The goal here is to produce, for every feature, a descriptor vector that will correlate closely to the descriptor of a corresponding feature in another image. The descriptor for a feature is computer as follow:

1. Place a grid over the feature point in the scale space it was found, and align the grid to the features orientation. The grid can be of any size; here we have used 4 by 4.
2. For each cell in the grid, measure the gradient of the pixels it contains. Produce a histogram showing the frequency of various gradients measured over the pixel in the cell. For this work we have used 8 bins per histogram.
3. The final feature vector is produced as the concatenation of histograms from each cell. For this project the result is a vector of length $4 \times 4 \times 8 = 128$.

A descriptor is produced for each feature we found in the previous section. These descriptors can be compared with each other using standard metrics, but it still leaves the problem of relational structure between feature points.

5.1.3. Matching

To incorporate relational structure in the comparison process, a number of techniques have been developed. One of the most popular involves comparing clusters of features using a generalised Hough Transform. A description of this, however, is outside the scope of the project, because the matching portion of the algorithm has not been used. Instead of worrying about relational structure we shall opt instead to make a simplifying assumption, that the relative position of features is not important, only their occurrence. This assumption underpins the Bag-of-Features technique.

A consequence of this assumption is that it weakens some of the requirements on the SIFT algorithm. In particular feature identification is designed such that interest points are likely to occur in similar clusters on corresponding images. This degree of rigour is no longer needed if relational independence is assumed.

5.1.4. Random Features

The modular design of SIFT allows components to be replaced or removed. One such replacement is to change the feature identification to a system where points are chosen at random. The distribution of these random points is typically uniform, but it could be Gaussian to emphasise description of the central focus of an image. The scale space must

also be found, but again we can chose this at random. The orientation can be measure, just as in full feature identification, or it can be set to some constant value.

The generation of random feature points has major benefits. Some images produce few features using the normal technique. In particular low resolution images, and images containing few corners can be sparse. Many of the Caltech-256 data-set produce as few as 8 features. Clearly this is not going to provide much descriptive power. Conversely, some images produce a disproportionately large number of SIFT features, and the temptation here is to remove a random sub-set of them, so the vocabulary does not become unduly biased towards these images. Additionally, decreasing the number of descriptors improves the computational performance of building the vocabulary tree. Random features resolve these problems neatly because the number of feature can be decided before hand, and will be the same for all images.

Another benefit to random features, is that they are significantly faster because the maxima and minima do not have to be exhaustively searched for. However, it is not a massive improvement because the description phase still requires a scale space pyramid. Calculating this pyramid is the most costly part of SIFT feature extraction, and since it has to be generated anyway the performance improvement is not a great as one would hope.

The greatest benefit from random features is that they have been demonstrated to improve accuracy. Research compared random key-points against LoG and Harris-Laplace based techniques. When used with Bag-of-Features classification, it was found that the simpled minded approach of random key-points performed best. (Nowak, Jurie and Triggs, 2006) At first these results seem counter intuitive, but consider that it exchanges relational compliance for descriptive variation and density. With Bag-of-Features we are already disregarding relational structure so it is redundant in the feature identification. But a vocabulary tree is extremely sensitive to the variance of descriptors so we can improve the solutions performance.

5.2. Vocabulary Tree

Have generated SIFT descriptors for each image, the goal now is to order images in the database according to some search criteria. To achieve this, a measure of similarity must be found between the search criteria (query images) and database images. It is possible to use SIFT descriptors directly for the measure of similarity, using the matching technique mentioned in Section 5.1.3. But this approach is much too computationally expensive. Matching two images can take seconds, but a search requires matching every query image against every other image. For example a typical database may contain 1000 images, and be queried with 8 images. If it takes 1 second to perform a second match at complete query will take 1000×8 seconds ≈ 2 hours. Clearly this is unacceptable.

One solution is to bin the descriptors, forming a vocabulary of visual words. In this way descriptor variance is quantised such that similar descriptors are treated as identical. Once quantised the image can be described by the frequency of occurrences of descriptors in each bin. Quantisation can be achieved by clustering descriptor vectors using one of the many approximate[1] clustering algorithms. In this work the Hierarchical K-Means (HK-Means)

[1]These algorithms are invariably only approximate because optimal clustering is in the problem class NP-hard for the general case. For a fixed number of clusters and vector dimensionality clustering is polynomial, but of an order that makes it prohibitively slow; asymptotically bounded by $O(n^{dk+1} \log n)$, where d is the vector dimensionality, k is the number of clusters, and n is the number of vectors. Typical values, in this project, for the above parameters are $d = 128$, k=1000, and $n = 100000$.

algorithm has been used for cluster descriptors. HK-Means is a variation on the classic K-Means algorithm. A descriptions of both algorithms follows.

5.2.1. K-Means

K-Means is a Machine Learning technique that can be used for modelling cluster (or clump) regularities in some distribution of variables. It takes as its principle parameters: The number of clusters k, a set of points (or vectors) to be clustered X, and some metric of distance between vectors $\mathrm{d}(\boldsymbol{a}, \boldsymbol{b})$. It produces a set of k centroids that are approximately optimal for the data. The algorithm follows the following basic steps.

1. Randomly generate initial positions for the k centroids. This is often achieved by setting each centroid to the position of a random input point.
2. For each $\boldsymbol{x} \in X$, assign $\boldsymbol{x}$ the centroid that minimises the distance metric. The points assigned to a centroid are knows as the captured points. In this project we shall be using the squared Euclidean distance metric. This is equivalent to Euclidean but is much faster to calculate.
3. Move each centroid to the position that minimises the distance to all of its captured points.
4. If any centroids moved then repeat from step 2.

In implementing this algorithm a number of other considerations must be made. For example centroids that capture no points must be handled. It is also possible to have scenarios where the algorithm never terminates. For a more detailed description see Pelleg and Moore (1999).

5.2.2. Hierarchical K-Means

HK-Means extends K-Means by making trees of cluster centroids. The data is first partitioned into k clusters, then each partition is again clustered. The process is repeated until some limiting condition is reached. Typically the limit is the density of points in a cluster, but in this work a different method has been used. Figure 5.2.1 shows the clustering of some generated data using an HK-Means algorithm. The goal is to produce an image signature; a vector that can be used as a measure of image similarity. To easily compare vectors they must be of the same length and structure. If we simply stopped clustering at a certain saturation level there would be little control over the length of the image signatures. The solution is stop recursing when the number of nodes in a given layer equals or exceeds a stopping parameter `nleaves`.

The final cluster hierarchy shall be referred to as the vocabulary tree. Once generated the descriptors of each image are pushed down the tree, a count is taken of how many times descriptors pass through each node. Image signatures represent occurrences of descriptors on the whole vocabulary tree. [2]

The `nleaves` parameter is always less than the signature length for all $\texttt{nleaves} > 1$. The true signature length can be calculated for a signature $\boldsymbol{s}$, given some value of `nleaves`,

[2] Note that image signatures are not strictly 1-dimensional histograms; they are in fact d-dimensional histograms of size k, where d is the depth of the vocabulary tree. In this project, the signatures are nominally treated as being single dimensional for the purposes of comparison, but it is important to remember they are not. In particular geometric assumptions of signature distance may be called into question.

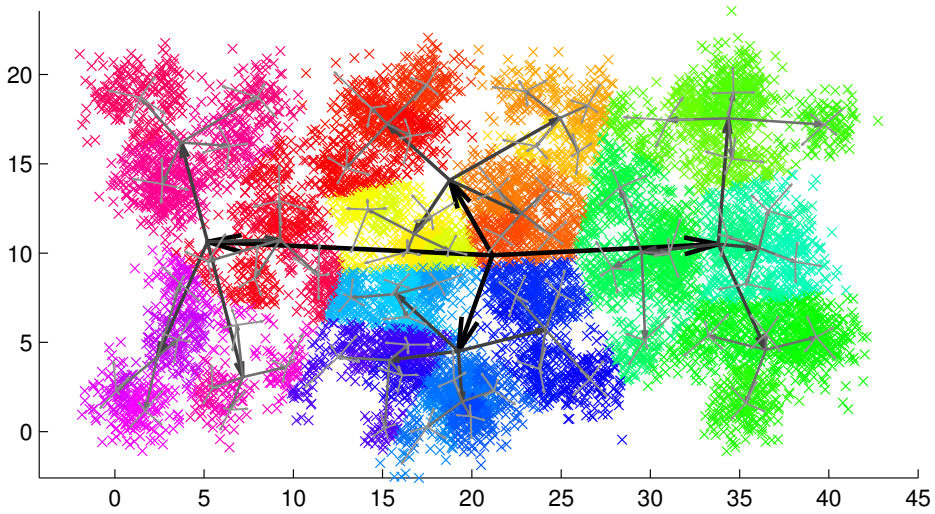

Figure 5.2.1.: Plot showing 10000 randomly generated samples being hierarchically clustered, using an HK-Means implementation. It was parametrised: $k = 4$, max depth $= 4$.

and a branching factor k:

$$\text{size}(\boldsymbol{s}) = \sum_{i=0}^{d} k^i, \text{ where } d = \log_k \texttt{nleaves}$$

For example: With `nleaves` $= 1000$, and $k = 10$, the signature length is $10^0 + 10^1 + 10^2 + 10^3 = 1111$.

The benefit of HK-Means over K-Means is in computation performance. Standard K-Means requires $\Omega(kn)$ calculations of distance for each epoch of the algorithm. With large values k this quickly becomes very expensive. HK-Means alleviates the problem by allowing a much smaller k to produce the same number of clusters. It is also faster to push vectors onto an HK-Means tree for the same reason. This does not affect query performance since signatures are always pre-calculated. There is an extension to this design that enables HK-Means to improve query performance, which is discussed in the next section.

5.2.3. Inverted File

An significant performance improvement that can be made to querying, which requires a tree based vocabulary structure. The HK-Means approach was proposed in Nister and Stewenius (2006) because it could be augmented with an inverted file structure on each node, pointing back the images. If the vocabulary tree was suitably sparse with respect to each image, a similarity metric would not have exhaustively evaluate images that did not share vocabulary nodes at the higher levels.

An inverted file structure (or inverted index) is used where content values are used as keys that reference the document they come from. It is typically employed in full text search

of documents. In the case of a documents, each word from all the documents would be an index that references the documents that contain it. When the database is searched for a particular word, the software only needs to look up the inverted file index for that word to retrieve all the relevant documents. Clearly this is potentially significantly faster than exhaustively searching every document in the database, given sparsity and a balanced tree.

In Nister and Stewenius (2006) they attempted to apply this technique to visual words defined by the descriptors centroids. Each node of the tree referenced the images which have descriptors captured by that cluster. A query on the database can then be performed by taking the signature of the query image (a histogram of vocabulary occurrences) and using it to search the inverted file vocabulary tree. The search process would proceed as follows, starting from the root node of the vocabulary tree:

1. For each node at the current level of the hierarchy:
    a) If the corresponding signature value is zero then continue.
    b) For each document referenced by the inverted file on this node:
        i. Calculate a score value from the signature value and weighting of this node.
    c) If this node is not a leaf then recurse on its descendants.
2. Return all the referenced images, ordered by the sum of their scores.

The advantage here is that descendants are only processed if their parents produced a non-zero score. Without the inverted file structure every signature value must be compared against every other signature value for all query and result images. This gives an asymptotic tight bound of $\Theta(mnl)$ comparisons, where n is the number of query image, m is the number of search images, and l is number of nodes in the vocabulary tree. With the inverted file, the lower bound is reduced by an amount proportional to the sparsity of the image signatures. This allows much higher fidelity vocabulary trees; as demonstrated in Nister and Stewenius (2006) where the best results where found with vocabularies of 1 million nodes.

As mentioned above, this optimisation depends on vocabulary sparsity. If the vocabulary tree is too saturated then this technique will be slower because of the increased constant time work of descending the tree. To maintain performance it will be important to implement some mechanism where each signature is pre-processed so it contains a predominance of zero values. This can be easily achieved by thresholding signature vectors.

The inverted file structure is important to mention, because it helps explain many of the design choices made in this work. It has not, however, been implemented yet, because it is merely a performance improvement. The primary objectives of this project are in producing proof of concept prototypes, not end-user software. Another reason this technique has not been implemented is because in Nister and Stewenius (2006) only single image queries where used, but in this project it is critical that queries using multiple images are possible. Combining multiple images is feasible using the inverted file, but summing signatures will inevitably reduce the sparsity making the naive application of this technique less beneficial.

5.3. Vocabulary Weighting Scheme

Having projected each image's descriptors onto the the HK-Means vocabulary tree, the resultant raw signatures are simply histograms of vocabulary usage per image. A key problem with these are that high value bins are not necessarily significant. Consider a lexical vocabulary as apposed to the visual vocabulary we are dealing with here, and

text documents instead of images. Words such as "the" appear with great frequency in the English language, and their occurrence in a document is insignificant because of this. Conversely the word "Minkowski" is much less common, and so it is likely to be highly indicative of a document containing it frequently.[3] In terms of comparing the similarity of documents the relative proportions of "the" is likely irrelevant, but if both contain a disproportionately high occurrence of "Minkowski" there is likely to be some relation. The same is likely to be true when comparing visual vocabulary usage. The most obvious choice for weighting scheme is the Term Frequency - Inverse Document Frequency (TF-IDF) scheme common in computational linguistics.

TF-IDF is calculated as the product of the frequency a particular term (or vocabulary element) in a single document, and the inverse proportion of documents that contain that term. In the case of this system a term is equivalent to a visual word, or one of the bins in the vocabulary tree. The raw image signatures already hold the TF in a primitive form. It is, typical, however to normalise the TF by "length" of a document so that for large documents, a proportionaly lower term frequency is not over emphasised. In the case of an image signature this is the number of descriptors in the image. The IDF is calculated by dividing the total number of documents in the system, by the number of documents that contain the given term. Further, the logarithm of this value is usually taken. The TF and IDF must be calculated and stored separately because the TF is unique to a specific signature, while the IDF depends on all signatures. The TF for a signature vector $\boldsymbol{s}$ is calculated:

$$\mathrm{tf}\,(\boldsymbol{s}) = \frac{\boldsymbol{s}}{\sum_j s_j}$$

The IDF for a database of signatures $S = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n\}$ at term i is calculated as follows.

$$\mathrm{idf}\,(S, i) = \log \frac{\left|S\right|}{\left|\{\boldsymbol{s} : s_i \neq 0\}\right| + 1}$$

Note the "+1" is cosmetic; it stops the occurrence of divide-by-zero problems when a term does not appear. Thus the IDF vector for a database S where each signature is of length m, is calculated

$$\mathrm{idf}\,(S) = (\mathrm{idf}(S, i) : i = \{1, 2, \ldots, m\})$$

The final weighted signature $\boldsymbol{s}'$ is produced by taking the product of the signature $\boldsymbol{s}$, the TF, and that IDF of all the signatures in the database S:

$$\boldsymbol{s}' = \boldsymbol{s} \times \mathrm{tf}\,(\boldsymbol{s}) \times \mathrm{idf}\,(S)$$

For optimal accuracy IDF should be recalculate with the addition of each new image. TF on the other hand is independent of other images and so can persist unless the vocabulary tree is regenerated. When building the database we shall incorporate the TF with the raw image signature, the resultant vector is what is referred to elsewhere in this document as simply "the signature". The IDF must be stored in the database object and multiplied with the signatures when a query is run.

[3] Of course in actual text processing words such as "the" are classed as stop words, and are usually removed before processing. Weighting the corpus frequency is still very useful for words that are common. As an aside, current research into applying textual paradigms to image processing has not found the visual equivalent of a stop word.

5.4. Signature Similarity Metrics

A key factor in the performance of the system is the metric used to compute the similarity of image signatures. We shall evaluate a number of possibilities. In each case a matrix of the query signatures $\mathcal{Q}$ is compared with a matrix containing all image signatures not in the query $\mathcal{D}$. The result is a score vector containing a similarity value for each signature in $\mathcal{D}$. This vector is then used to ordered result elements such that the most similar is the first element, which may be either ascending or descending depending on the metric. The scoring methods follow. For simplicity they are described in terms of comparing two signature vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, rather than the signature matrices $\mathcal{Q}$ and $\mathcal{D}$. Final similarity score for each signature in $\mathcal{D}$ is calculated by summing the distance to each signature in $\mathcal{Q}$.

Minkowski Distance The L_p space generalisation of the Euclidean distance metric can used to calculate the similarity of signatures. Various values of p can be tried including the Manhattan distance ($p = 1$), and Euclidean distance ($p = 2$). Clearly here smaller values should indicate similarity and so the resultant score as returned in ascending order.

$$\mathrm{d}\left(\boldsymbol{a}, \boldsymbol{b}, p\right) = \left(\sum_i \left|a_i - b_i\right|^p\right)^{1/p}$$

Inner Product Viewing the signatures a vectors we can get a measure of their similarity by calculating their inner (or dot) product. This will be zero when the vectors are orthogonal, if they have non-zero length. As the angle between the vectors decreases the inner product increase, indicating greater similarity. In addition a larger magnitude (greater significance) of each scalar pair increases the inner product, again indicating greater (and more significant) similarity. Here unlike Minkowski distance small values indicate dissimilarity, and large values indicate similarity, so the results will be sorted in descending order.

$$\mathrm{d}\left(\boldsymbol{a}, \boldsymbol{b}\right) = \left\langle \boldsymbol{a}, \boldsymbol{b} \right\rangle = \sum_i a_i b_i$$

Boolean Similarity Consider an example where two images of the same subject are being compared; one image is a close up, while the other contains more background scene. The SIFT algorithm will hopefully produce subject-like features for both images but will probably produce significantly more for the close-up image. Assuming subject-like features are correctly represented in the vocabulary, the close up will have a higher frequency in these bin. The result is that image signature values corresponding to this bin will be dissimilar in terms of Minkowski distance, despite the relatively high values in both bins being a good indicator of similarity. A simple solution is to ignore the magnitude of the similarity all together. Simply choose some threshold t of absolute scalar difference under which we will assert the distance is 0, and over which the distance is 1. If the signatures contain integer values and are quite sparse then $t = 0$ could be a reasonable choice. Otherwise some other value $t \geq 0$ must be found.

$$\mathrm{d}\left(\boldsymbol{a}, \boldsymbol{b}, t\right) = \sum_i \mathrm{g}\left(\left|a_i - b_i\right|, t\right), \text{ where } \mathrm{g}\left(x, t\right) = \begin{cases} 0, & x \leq t \\ 1, & x > t \end{cases}$$

Other signature similarity metrics have been considered including Mahalanobis distance and the more computationally efficient[4] Normalised Euclidean distance, to account for scale variance between vocabulary bins. This is not necessary however because of the vocabulary weighting scheme which already deals with this problem. The Earth Movers Distance (EMD) (Ling and Okada, 2007) was also considered since the signatures are effectively weighted histograms and EMD has been shown to be an effective measure of histogram similarity. This shall not be implemented, however, because of high complexity and computational overhead of the EMD algorithm.

The need for various similarity metrics means that the metric must be supplied to the database by a reference, either a a metric object, or a lambda function.

5.5. Error Metric

There is need for some way to measure the accuracy of the software; to tell if it works, and if so to what extent. This will also help determine good parameters for the various algorithms. To achieve this training data is needed, against which we can measure our predictions. An assumption being made here is that the classes, as defined by the data-set, correspond with what the user expects. For example the Buildings/People data-set (Section 3.5) has images marked up as either containing a building or a person. It is assumed that this will also be the intended labelling of the user of the system rather than, for example, trees and cars.

Given this assumption, upon performing a query the results should have images near to the top that are of the same class as the query images. A perfect result would occur if all the same class images were at the top, followed by everything else. It follows then that the error should increase for every image, in the top part of the result, that is of a different class. Furthermore, the higher incorrect recommendations are the worse. To clarify with an example: Say there are 120 images in the database, 40 from each of 3 classes: x, y, and z. The user selects 8 images from x as the query, and a result set of 112 images is returned. A perfect (0.0 error) result should occur if the first 32 images of the result are all in x, while the remaining 80 is made up of y and z in any order. A chance (0.5 error) result should occur if $\frac{1}{3}^{rd}$ of the first 32 images are in x. A worst case (1.0 error) result should occur if the first 32 images contain no xs at all. In addition given two recommendations sets: (a) Where the first 2 results are y then x, and (b) Where they are x then y. It is clear that (b) is preferable to (a), so we must account for relative position.

To achieve this we shall ignore results that are of the same class, and images that are outside of the target class partition (the first 32 in the example above). For each incorrect recommendation inside the target class partition the error should be larger if it is near the top.

Given a database containing n images, m of which in class t, a query set x of length $|x|$ containing images of class t, and a result set y. There are $p = m - |x|$ potential matches. The error of y is the sum of errors for each incorrect recommendation in the first m, divided

[4] Mahalanobis distance requires the the covariance matrix of the vectors it is to work upon. For n-dimensional signatures this covariance matrix is of size n-by-n. Depending on the vocabulary configuration, an image signatures can have length $n > 10000$. This results in covariance matrix that consumes $\approx$ 500MB of memory and takes a very long time to compute.

by the total possible error:[5]

$$e_y = \frac{\sum_{i=1}^{p} |\text{class}(y_i) - t| \, (p - i)}{\sum_{i=1}^{p} (p - i)}$$

The problem here is that chance error will only be 0.5 for an image database with an equal likelihood of getting correct or incorrect images. From the 3 class example above there is a $\frac{112-32}{112} = 0.80$ chance error, because there are significantly more images of the wrong class. We can fix a chance error of 0.5 by including a normalisation term:

$$e_y = \frac{\sum_{i=1}^{p} |\text{class}(y_i) - t| \, (p - i)}{\sum_{i=1}^{p} (p - i)} \cdot \frac{0.5}{1 - \frac{p}{n}}$$

[5]Note that bars $|\ldots|$ indicate the absolute value here, where $|x|$ indicates cardinality.

6. Implementation

The software has been developed over several prototyping iterations. Each iteration has been informally evaluated and tested before changes where made. It is primarily build using Matlab, because this development environment offers offers very powerful libraries when dealing with images and complex data. In addition it allows seamless integration with C and Java code, so each language can be utilised as necessary.

The final prototype has the following basic directory structure:

6.1. SIFT

The SIFT algorithm, implemented by Andrea Vedaldi (Vedaldi, 2009) and distributed under the BSD License. Available from: http://www.vlfeat.org/∼vedaldi/code/sift.html (Last visited 2^{nd} December 2009).

`src/sift/sift.m` Function that computer SIFT feature key points and descriptors for a given image.

6.2. HK-Means

This authors implementation of K-Means and HK-Means algorithms. While it works well, it has been demonstrated to be slow compared to available libraries. This is partly because it is write entirely in Matlab, and would benefit greatly from porting to C. This code is not currently used in the prototype evaluation because of its performance limitations. It has been included for demonstration purposes.

`src/kmeans/kmeans.m` Standard K-Means clustering algorithm. Takes a matrix of samples and some value for constant k. It returns the positions of k clusters that minimises the average distance from each sample to its nearest cluster centroid. It is highly configurable with parameters to control the distance metric, initialisation method, and strategies for handling empty clusters. It also has an experimental system for gradual convergence that may help avoid local optima.

`src/kmeans/hkmeans.m` Hierarchical K-Means clustering algorithm. Takes a matrix of samples and some value for constant k. Recursively partitions and clusters the samples using the standard K-Means (above) returning the root node of a cluster tree.

`src/kmeans/plothcluster.m` Utility to plot the results of hierarchical clustering, such as the image in Figure 5.2.1.

6.3. Another HK-Means

Andrea Vedaldi wrote a framework for a Bag-of-Features software prototype (Vedaldi, 2009), however only the clustering functions are used. They are significantly faster than

this authors implementation (above), because they are written in native compiled C, and because they implement various optimisations such as converting the data to be clustered into fixed point numbers. The framework is distributed under the BSD License. Available from: http://www.vlfeat.org/∼vedaldi/code/bag/bag.html (Last visited 2^{nd} December 2009).

`src/bag/hikmeans.m` Fast Hierarchical K-Means clustering working with only signed byte data types. It produces a structured hierarchy of cluster centroids and their descendants.

`src/bag/hikmeanspush.m` Once an HK-Means tree has been build, this function is used to push an images descriptors down the tree. For n descriptors and a tree of depth d, this function will return a d-by-n matrix showing the path each descriptor took down the tree.

`src/bag/signdata.mex.c` Once the paths have been calculated, this function converts it into a raw signature by summing the occurrences of descriptors on each node of the tree. It produces a a vector of length equal to the number of nodes in the tree.

6.4. Utilities

Contains a number of library functions, created for this work, but which are not considered to be project specific.

`src/utils/digest.m` Takes any number of arguments and produces from them a SHA1 hash code. The arguments can be of any type except Matlab or Java objects. This is used to create unique file names for cached data such as SIFT key-points and descriptors.

`src/utils/findfiles.m` Takes an expression argument that is used to search the file-system for matching files. For example it can be given an expression such as `../data/*.jpg` which will return a list of all files inside the `data` directory and its descendants that have a `.jpg` extension.

`src/utils/normalise.m` Takes an n-dimensional matrix and normalises it in range 0 to 1 inclusive, across one or more dimensions.

`src/utils/sizebytes.m` Calculates the size in bytes of its arguments. Used to measure the memory requirements of the image database.

6.5. Image Database

Contains the image database classes and functions.

`src/imdb/Image.m` A class to hold information about a single image in the database.

`src/imdb/ImageDatabase.m` Class to hold all the images in the system. It provides methods to manipulate and search the images.

`src/imdb/dosift.m` Utility that calls the main sift function.

`src/imdb/resultError.m` Calculate the error of a query result using the metric described in Section 5.5.

`src/imdb/smBoolean.m` The Boolean signature similarity metric, as described in Section 5.4.

`src/imdb/smInnerProd.m` The inner product signature similarity metric, as described in Section 5.4.

`src/imdb/smMinkowski.m` The Minkowski signature similarity metric, as described in Section 5.4.

`src/imdb/smRand.m` A random signature similarity metric, to be used as a base line during evaluation.

`src/imdb/runQuery.m` Utility to run a query on the database and display the results.

`src/imdb/runCrossValidate.m` Function to wrap querying and evaluation, such that the prototype can be tested using cross-validation libraries provided by Matlab.

`src/imdb/plotErrors.m` Draw a graph showing the training and generalisation errors over various criteria.

`src/imdb/plotQueryIms.m` Display the images used in a query, with the results in grid.

`src/imdb/plotConfusionMat.m` Draw confusion matrices for a set of queries to the database.

6.6. Demonstrations and Experiments

`src/test_init.m` A script, run before other evaluation scripts that initialises the environment.

`src/test_kmeans.m` A demonstration of this authors implementation of HK-Means clusters.

`src/test_buildingspeople.m` Demonstration using a small data-set, used during implementation.

`src/test_caltech256easy02.m` Evaluation scripts using a 2 category subset of Caltech 256, the results of which are in Section 7.1.

`src/test_caltech256easy10.m` Evaluation scripts using a 10 category subset of Caltech 256, the results of which are in Sections 7.2, 7.3, 7.4, and 7.5.

`src/test_caltech256easy20.m` Evaluation scripts using a 20 category subset of Caltech 256, the results of which are in Section 7.5.

`test_speed_sift.m` Perform tests that evaluate the running time of the SIFT algorithm with respect to image size.

`test_speed_buildvocab.m` Perform tests that evaluating the HK-Means algorithm running time with respect to number of images, and signature length.

6.7. Data

`src/cache/` Once a SIFT key/descriptor pair is calculated, it is stored here for later use.

`src/data/` A number of data-sets used during formal and informal evaluation of the prototype software. Except where noted, they will not be distributed with the dissertation files, because of their extreme size, but can be downloaded from the provided URLs.

`caltech-4/` Available from:
http://www.robots.ox.ac.uk/∼vgg/data3.html
(Last visited 20nd April 2010).

`caltech-101/` Available from:
http://www.vision.caltech.edu/Image_Datasets/Caltech101/
(Last visited 20nd April 2010).

`caltech-256/` Available from:
http://www.vision.caltech.edu/Image_Datasets/Caltech256/
(Last visited 20nd April 2010).

`LabelMe/` Available from:
http://labelme.csail.mit.edu/
(Last visited 20nd April 2010).

`peoplebuildings/` A very simple data set, used during implementation and early testing. The images are included with the source code in the digital submission file.

7. Evaluation

7.1. A Simple Demonstration

To start of the evaluation let us consider a relatively trivial example: An image data set consisting of two very dissimilar sets of images. These two sets (*motorbikes*, and *faces-easy*) where taken from Caltech-256. They where chosen because they have been demonstrated to be among the easiest in Caltech-256 (Griffin, Holub and Perona, 2007). A database was initialised with 160 randomly selected images images, 80 from each set. The vocabulary was build with $k = 10$, and 100 leaf nodes. This gives a depth $= 3$, and signatures length $= 111$. Once, initialised the database consumes $\approx 3.2MB$ of memory excluding the image files themselves. There are a total of 17901 feature descriptors, with an average of 112 descriptors per image.

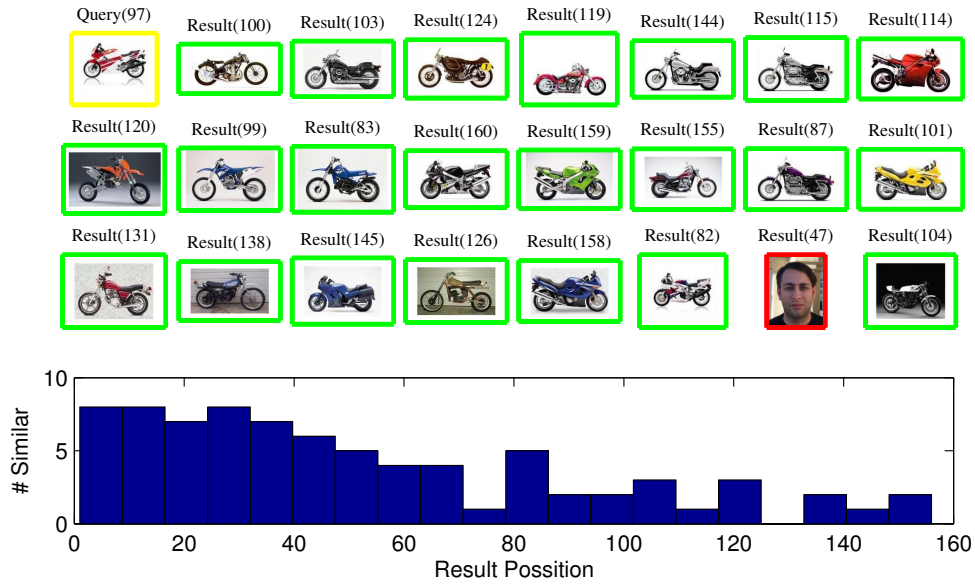

Figure 7.1.1.: Single image query and first 23 results, using a *motorbikes* set image. This histogram shows the distribution of results that are of the same class at the query image.

(Note that incorrect suggestions are highlighted in a red box, correct suggestions are in green, and the query is in yellow. The bracketed number in the title is the index of the image in the database.)

The database was queried with randomly selected single images from the same categories. The Inner Product scoring metric was used for measuring signature similarity. Figure 7.1.1 shows an example of a result set. It was selected because it appeared to be representative of the performance of this database. The error (calculate using the metric described in Section 5.5) was 0.1147. This of course is a training error because the query image was drawn from the database itself. The results are very good, with only one of the first 23 recommendation being incorrect. The histogram bellow shows the distribution of images

of the same class at the query in the result set. Clearly the majority of the results are near the beginning, which is encouraging.

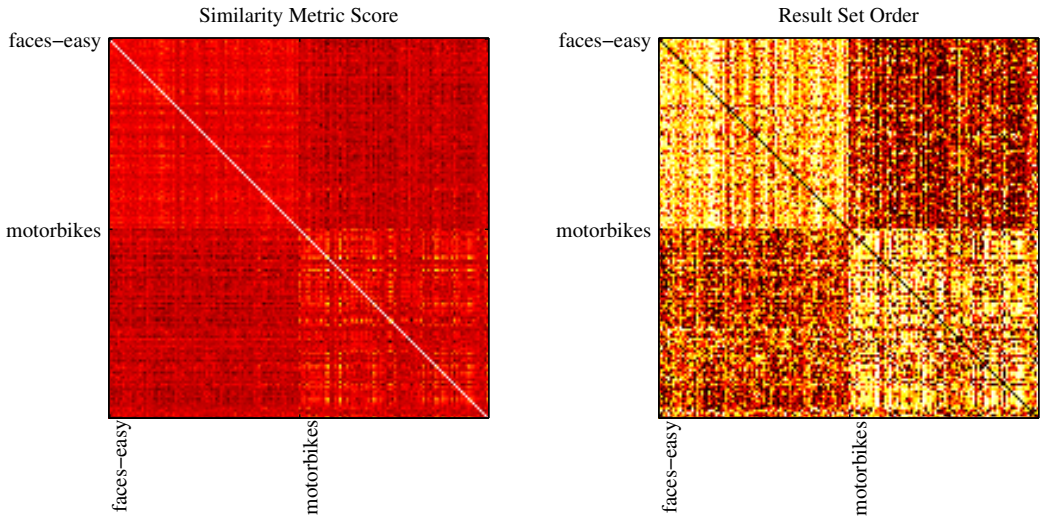

Figure 7.1.2.: Visualisation of a confusion matrix, showing the degree to which each image is similar to other images when performing a query. In both cases lighter colour indicates a stronger similarity.

To inspect how well the software is performing, we can view the results of all single image queries as a confusion matrix. This shows the degree to which a query image recommends every other image in the database. As a matrix there is a row for each query image, and a column for each result image. Rather than give the actual values it is easier to render the confusion matrix as a image. There are two possible ways of denoting recommendation here; the value of the similarity metric, and the index each image appeared in the recommendation list. Figure 7.1.2 shows both of these confusion matrices. Because the images are inserted into the database in order of category, the first 80 are all *faces-easy*, and the remained are all *motorbikes*. This gives a distinctive checker pattern where images of the same category tend to recommend each other more strongly than images of another category. The checker pattern is most evident in the confusion matrix based on the order of results. This is because it is not absolute difference between scores that really matters, but only that there *is* a difference. So *faces-easy* images do have some similarity to *motorbikes*, but usually not enough for *motorbikes* to be foremost in the result set.

Fold	Training Error	Generalisation Error
1	0.1740	0.1843
2	0.1626	0.1899
3	0.2029	0.1946
4	0.2057	0.1853
5	0.1758	0.1614
Mean	0.1842	0.1831

Table 7.1.1.: Table showing the training and testing errors over 5-fold cross-validation.

A concrete impression of the solutions performance with this data set and configuration can be calculated using cross-validated generalisation error (in the manner described in Section 4.3.2.) The configuration was identical above, except that a different random

subset of image sets was taken. Table 7.1.1 shows resultant errors of this experiment. Curiously the mean generalisation error is slightly lower than the training error. While this demonstration is hardly conclusive, it does show that the solution function relatively well on this data set.

7.2. Larger Data Set

The performance on the data-set used in Section 7.1 above was excellent; it was so good that measurement of performance approached saturation. A harder data set was required: 80 images from each of 10 classes from Caltech-256. The classes where: *faces-easy, lawn-mower, tower-pisa, leopards, desk-globe, trilobite, watch, airplanes, car-side, motorbikes.* These sets have been chosen because research has shown them to be among the easier classes in Caltech-256.(Griffin, Holub and Perona, 2007)

Because we are using many more images, and a more diverse selection, it is likely that we will need to increase the descriptive power of the vocabulary. Therefore the the number of leaf nodes in the vocabulary was increased from 100 to 1000. Other parameters stayed the same from the demonstration in Section 7.1: $k = 10$, and the signature similarity is measured using the inner product metric. The new leaf nodes value implies a signature length of 1111.

The database was parametrised as described above, and built using 80 images from each of the 10 classes, for a total of 800 images. The images produced 99323 features with an average of 124 features per image. The total memory required to hold the database is 21.8MB = 27KB per image.

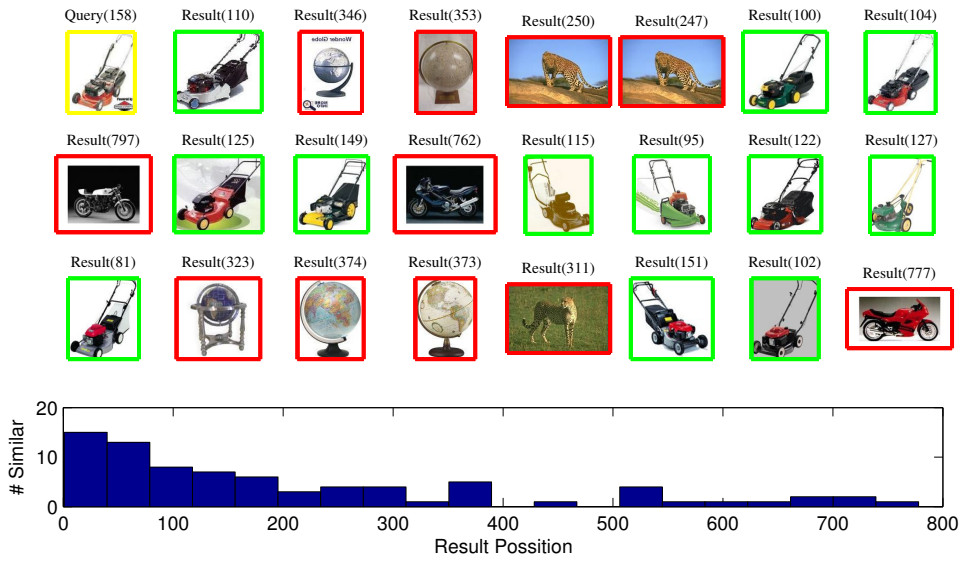

Figure 7.2.1.: Single image query and first 23 results, using a *lawn-mower* set image. The histogram shows the distribution of results that are of the same class at the query image.

Figure 7.2.1 shows an example single image query made upon the database. It was chosen because it seemed to be roughly representative of its accuracy, which is noticeably worse than the the database in the previous section. The error for this query results set was calculated as 0.333398. The histogram distribution demonstrates that while performance

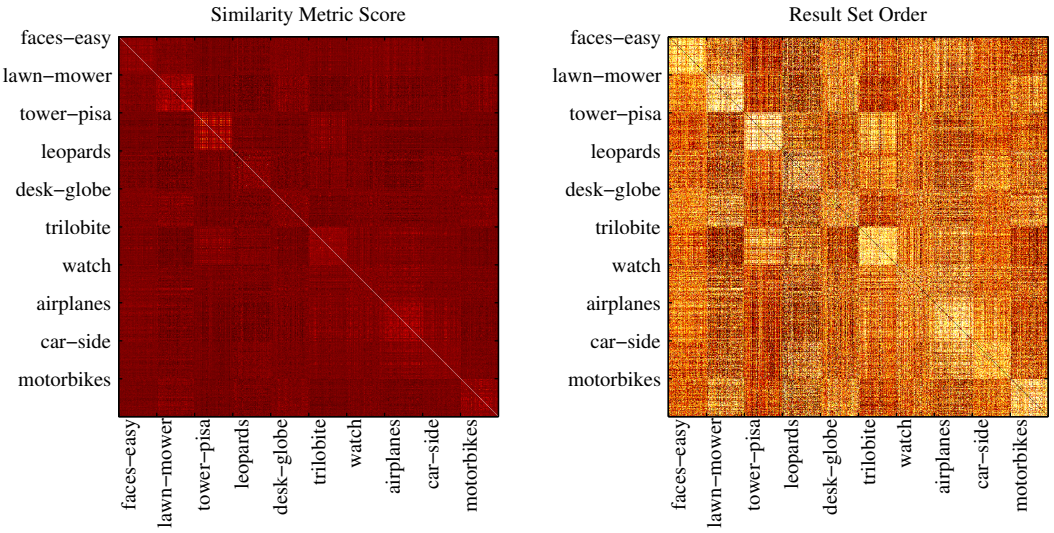

Figure 7.2.2.: Visualisation of a confusion matrix, showing the degree to which each image is similar to other images when performing a query. In both cases lighter colour indicates a stronger similarity.

is poor, the results are far from random with a distinct bias to *lawn-mower* appearing near the top of the result set. Figure 7.2.2 shows confusion matrices for all single image queries on this data set. The checker pattern is still clearly visible, but not as distinctly as before. In particular the result set order matrix says a lot about the data set: *lawn-mower*, and *tower-pisa* have the strongest within-class similarity. The *tower-pisa* and *trilobite* sets have a high degree of inter-class similarity, as do *airplanes* and *car-sides*. The poorest class appears to be *watch* which has a very even spread of similarity across all classes.

7.3. Multiple Image Queries

A key idea, behind the design of this software, was that increasing the size of the query improves retrieval accuracy. Multi-signature queries where achieved by summing the signature distances over a series of single image queries. It was important to verify that increasing the query size does indeed improve retrieval performance.

Figure 7.3.1 shows the results of another query made upon the data-set from the previous section. This time 8 *lawn-mower* images where used for the query. The error for this result was 0.098266; a huge improvement. Visual inspection shows that both the top results and histogram are much better than the single image query. However this is far from conclusive, because either one or both of these results are likely to be unrepresentative.

To evaluate the effect of varying the number of query images, the databases was repeatedly trained with all query sizes in the range 1 to 20. For each query size, 5-fold cross-validation was performed and the mean average training and generalisation error was calculated. Figure 7.3.2 shows a plot of the mean errors for each query size. It clearly demonstrates that, as the query size increases, both the training and generalisation error decrease. For queries of 1 image the training error is 0.3699 and the generalisation error is 0.3706, which is poor compared to the results of the previous section. However, for queries of 10 images the errors have fallen to 0.2403, and 0.2468 respectively. The plot also shows that in the first ten cases the generalisation error is slightly (≈ 0.004) worse than the training error.

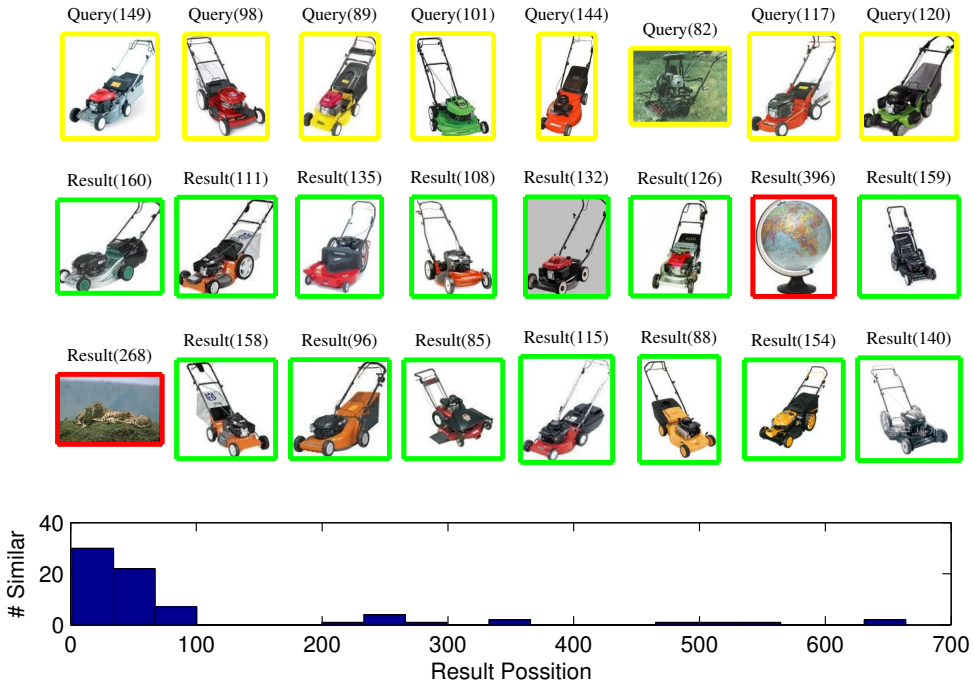

Figure 7.3.1.: Eight image query and first 16 results, using a *lawn-mower* set image. The histogram shows the distribution of results that are of the same class at the query image.

This is an ideal result. A generalisation error was significantly larger than the training error would imply that the model was over-fitting the noise of the training samples. If the generalisation error was below the training error, that would be a sign that the training samples where not representative of the test samples. As the query size increase, so does the gap between the generalisation error and the training error.

7.4. Scoring Methods

Thus far we have been using the inner product scoring metric for signature similarity. However, there are other candidates that may perform better, as described in Section 5.4. To measure the performance we shall compute the mean 5-fold cross-validated training and generalisation error for each of these metrics:

- Random scoring as a baseline with which to evaluate the other metrics. The results are returned in a randomly generated order. The error here should be ≈ 0.5.
- Minkowski Distance for all $p \in \{1, 2, 3\}$, where $p = 1$ is the Manhattan distance, and $p = 2$ is the Euclidean distance.
- Inner product or dot product between signatures.
- Boolean similarity that ignores the frequency of occurrences.

The database was otherwise configured and built using the same parameters: $k = 10$, leaf nodes $= 1000$. The same images where used from the previous section: 10 categories, 80 images per category. Queries where performed with 8 image signatures from the same category. Figure 7.4.1 shows the resultant mean errors for each metric. The random metric is, as expected, almost exactly 0.5 for both training and generalisation error. For the

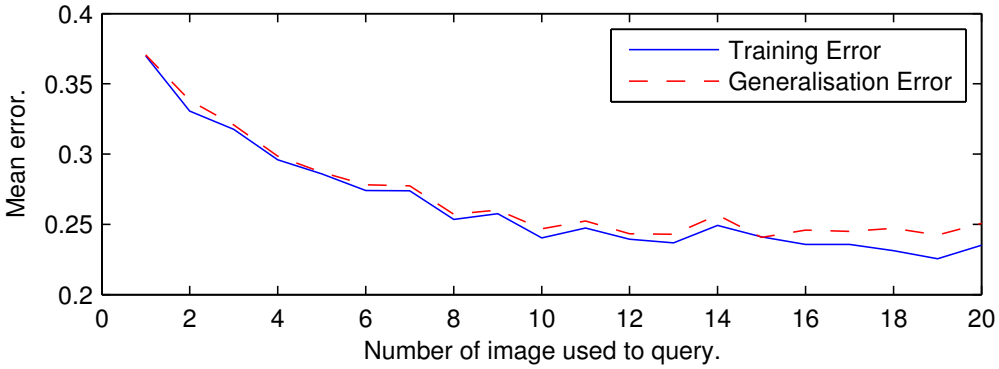

Figure 7.3.2.: Plot showing the mean cross-validated training and generalisation errors while varying the number of images comprising the query.

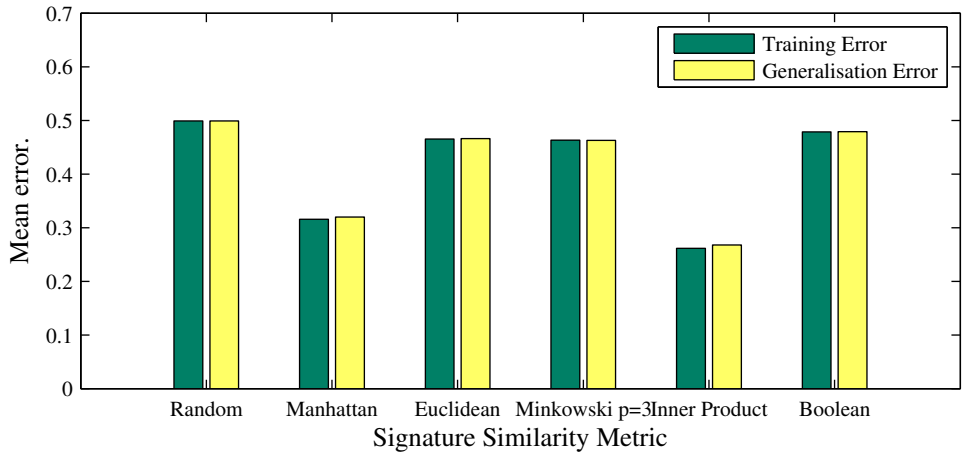

Figure 7.4.1.: Plot showing the mean cross-validated training and generalisation errors while varying the signature similarity metric.

remaining metric the generalisation error is slightly worse in all cases, but by an insignificant amount. The Euclidean metric is only slightly better that random. Likewise the Minkowski $p = 3$, and Boolean metrics are very poor. The Manhattan metric (Minkowski $p = 1$) seems to work quite well. Best of all is the Inner Product metric with a training and generalisation error of ≈ 0.28. It is no coincidence that the Inner Product metric is the one we have been using thus far in the evaluation. Previous work had already shown informally that it performed well. This experiment only confirm these initial results.

7.5. Vocabulary Size

Previous research has shown that `nleaves` $= 10000$ is sufficient for most image databases. (Nister and Stewenius, 2006) However, it is clear from the preceding sections that `nleaves` $= 1000$ performs well for the image database used. To understand the effect of vocabulary size on retrieval performance further evaluation was performed. The database was repeatedly training on the same 800 images, using cross-validation, for all `nleaves` $\in \{25, 50, 100, 200, 500, 1000, 2000, 5000, 10000\}$. The other parameters where kept constant: Inner product similarly metric, and 8 images per query. Figure 7.5.1 shows

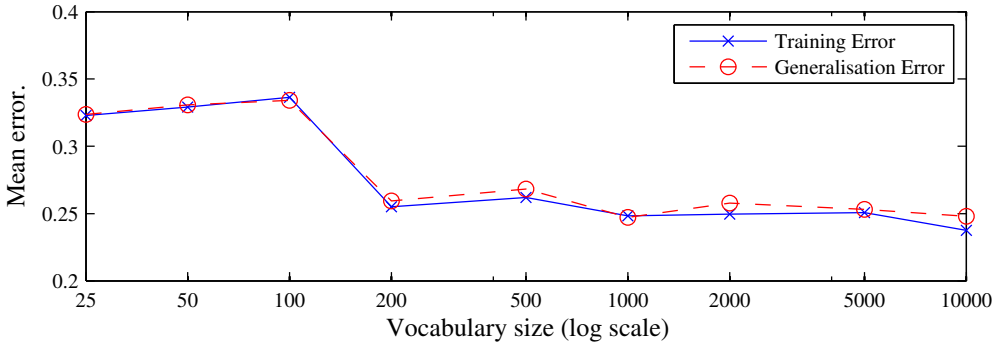

Figure 7.5.1.: Plot showing the mean cross-validated training and generalisation errors while varying the number of leaf nodes in the vocabulary tree, using 800 images from 10 categories.

the resultant generalisation and training errors, from which it is clear that `nleaves` = 200 is approximately optimal for this image set. Lower values worsen the errors drastically, while high values offer little improvement. It is also interesting to note that the generalisation error appears to diverge from the training error as the vocabulary size increases. This may show that a larger vocabulary weakens the bias of the model, allowing it to over-fit noise in the training samples. These results are only for one set of images, and do no really demonstrate how good vocabulary size varies with the number of images.

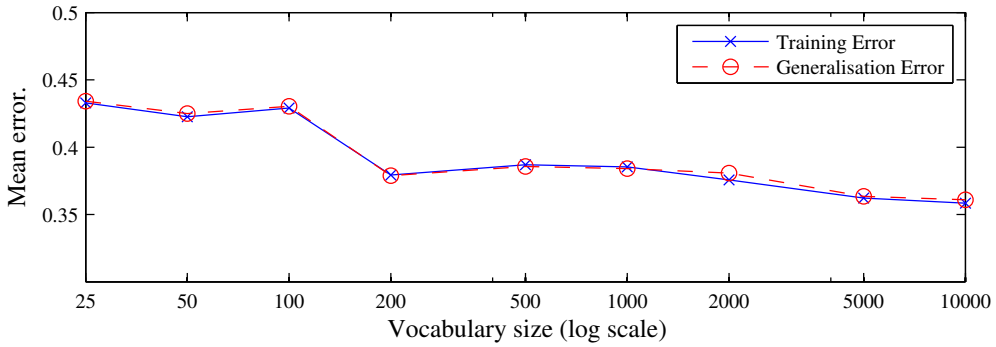

Figure 7.5.2.: Plot showing the mean cross-validated training and generalisation errors while varying the number of leaf nodes in the vocabulary tree, using 1600 images from 20 categories.

To better understand the relationship between vocabulary size and the number of images the previous experiment was repeated; this time with 1600 images from 20 categories. The 10 categories from before where used with the addition of: *american-flag*, *bathtub*, *mars*, *butterfly*, *coin*, *dolphin*, *stained-glass*, *giraffe*, *waterfall*, and *human-skeleton*. Research has show these images to be less amenable to various classification techniques than the initial 10 (Griffin, Holub and Perona, 2007). Figure 7.5.2 shows the mean generalisation and training errors over various `nleaves`. The plot is remarkably similar to the previous plot. For all `nleaves` < 200 the error is high, but at 200 the error drops significantly. The errors remain approximately constant for all `nleaves` ≥ 200, although there is a drop towards the end (10000) that may be significant. The gap between generalisation and training error appears much narrower here, but note that the y-axis range is larger here. The divergence between generalisation and training error (observed above) is less evident here.

The most significant difference between the two Figures is that the mean errors are significantly worse in Figure 7.5.2. For 10 categories the errors where in the range 0.25 - 0.35, where for 20 categories they are in the range 0.35 - 0.45. This indicates that the 10 extra categories really are a lot harder, and they have seriously degraded the mean errors.

Ultimately these experiments raise more questions than they answer. Increasing the number of images has not been shown to significantly altered the best vocabulary size of `nleaves` = 200, but the resolution of the plots is not fine grained enough to be able to state this with any certainty. It may be, for example, that the best vocabulary size for 800 images is 180, while the best for 1600 images is 300. In addition these results are only indicative with respect to this specific data-set. It could be that a vocabulary size of 200 is highly sub-optimal for some other collection of 800 images.

7.6. Computational Performance

7.6.1. SIFT Performance

To optimise the computational performance of the system it would be desirable to bound the SIFT algorithm such that it does not take an excessive amount of time. The time taken to run SIFT on an image varies with respect to the image size, and to the key-point density. In the case of image size, the effect can be measured trivially by recording the time taken for a range of image sizes. Figure 7.6.1 shows the results of such an experiment, where computation time is plotted over image size. A polynomial function has been fit to the data. It is not clear from this whether the relationship is indeed polynomial or linear, but that ambiguity is welcome. A clearly polynomial relationship could be disastrous for the performance of large images. We can use this information to bound computation time quite accurately by resizing the image prior to SIFT. For example, to set an upper bound of 10 seconds we could re-size all images, greater than 2 million pixels, down to that limit.

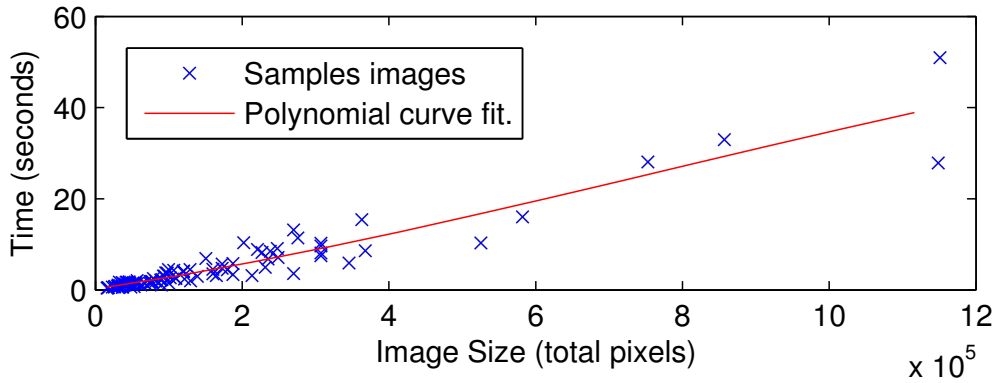

Figure 7.6.1.: Plot showing computation time of SIFT key-points and descriptors with respect to image size.

With regard to the number of key points, there is no easy way to measure key-point density prior to running SIFT. Thus there is no reason to measuring the effect if we cannot handle it. However, if future work used the random key-points technique (described in Section 5.1.4), fine grained control over key-point density would be available. The experiment should be performed then to help find optimal parametrisation of the random key-point algorithm.

7.6.2. Building Vocabulary Performance

After SIFT, the next most significant computation, in terms of time required, is the building of the vocabulary tree. The K-Means algorithm is an approximation, but it can still take a long time. Worse still is that the time taken, and the optimality of the result, varies greatly depending on the random initialisation. This variation cannot be controlled, but there are two factors in the algorithm that we can control: the number of descriptor vectors being clustered, and the number of clusters to make.

7.6.2.1. Number of Descriptors

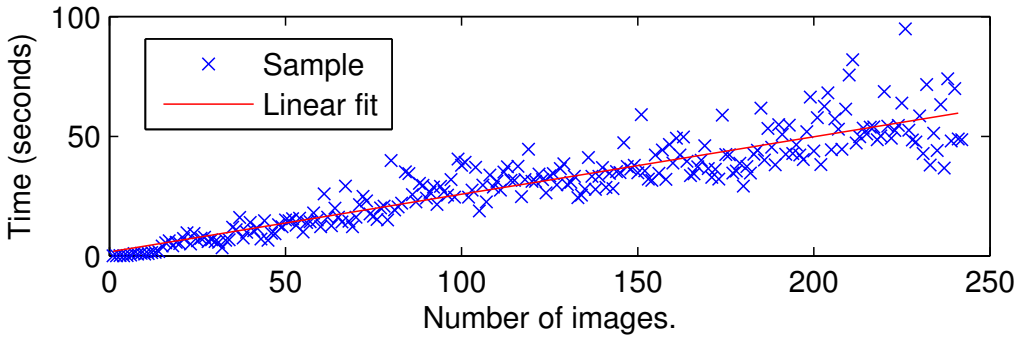

Figure 7.6.2.: Plot showing computation time of building vocabulary tree with respect to the number of images.

The number of descriptor vectors is approximately proportional to the number of images. So to measure the effect of descriptor density the vocabulary was repeatedly re-build while increasing the number of images. Figure 7.6.2 plots the results of this experiment, which indicate that mean computation time increases linearly with respect to the number of images. The variance around the mean also increases significantly. This is troubling because, for a desktop application, highly unpredictable running times are confusing for the user. If this project were to get beyond the prototype phase, this variance would need to be controlled somehow.

Figure 7.6.2 shows that the build time for 100 images is ≈ 25 seconds, and for 200 images is ≈ 50 seconds. Since relationship appears to be linear the build time per images is ≈ 0.5 seconds.

7.6.2.2. Number of Clusters

The other factor that is likely to have a significant effect on clustering performance, is the target number of clusters. In the case of standard K-Means this is the parameter k. In the case HK-Means it is a combination of k and `nleaves`. Figure 7.6.3 plots the time required to cluster image descriptors, while varying k and `nleaves`. The left-hand plot is the whole experiment, while the right-hand plot shows only the detail of signatures less that 400. They seem to indicate that computational time varies sub-linearly with respect to the number of clusters.

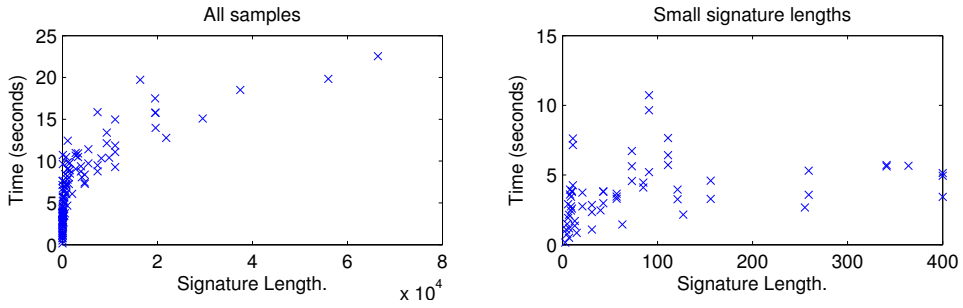

Figure 7.6.3.: Plots showing computation time of building vocabulary tree with respect to the signature length.

7.6.3. Querying Performance

The final important criteria in terms of performance, is the time it takes to perform a query on the database. The major factors in this are the number of image signatures (or just the number of images), the length of the signatures, and the signature similarity metric. There is no need here for an experiment because the asymptotic running time can be calculated directly from the formulae, given in Section 5.4.

For all metrics the running time is $\Theta(n \cdot m \cdot l)$, where n is the number of images in the databases, m is the number if query images, and l is the length of the signatures. This means that query time increases linearly with respect to any one of these factors. However, increasing the number of images may require increasing the signature length, so for effective retrieval the query time may be super-linear with respect to the number of images. This relationship between the number of images and vocabulary size, is speculative. And should be a focus of future research.

7.7. Acceptance

7.7.1. Functional Requirements

Table 7.7.1 shows the acceptance status of functional requirements, as stated in Section 3.2.1. Most of these have been met without reservation. Of note are those marked asterisk (*), which are supported by the model, but for which no explicit functionality has been coded. They are passed in the sense that it is definitely possible to write a user interface that would support these operations. The final item, regarding types of labels that can be retrieved has not been exhaustively evaluated. It is likely that the system will work with object based labelling, but will perform poorly on scene or phenomena based labelling.

Requirements	Met
A database of images to be viewed.	✓
Images can be labelled with zero or more textual descriptors.	✓
A label can be applied to one or more images.	✓
Images can be viewed by label.	✓
Suggestions are made about what images are most likely to be added to the group denoted by a label.	✓*
Suggestions can be confirmed and added to the group.	✓*
Suggestions can be denied, removing them from the result set.	✓*
Effective image retrieval based on a wide variety of label types.	†

* There is no explicit method for this, but it can be achieved using a combination of calls.

† Not fully evaluated.

Table 7.7.1.: Table showing which functional requirements have been met.

7.7.2. Non-functional Requirements

Table 7.7.2 shows acceptance criteria for non-functional requirements as stated in Section 3.2.2. These have all been demonstrated and met, with the exception of the item marked with an asterisk, which has not been demonstrated in this report. Informally however it has been shown to be passed, with queries always taking sub-second times.

Requirement	**Met**
The system should run on a standard, off-the-shelf personal computer.	✓
Query time is less than a second for an averaged sized (≈ 1000) image databases.	✓*
Query time is asymptotically invariant to the images pixel size.	✓
Query time is bounded by $O(n \log n)$, where n is the number of images.	✓ (see Section 7.6.3)
Pre-retrieval computation time < 1 minute per average size image ($\approx 1024 \times 768px$),	✓†
Adding an image or label does not require database regeneration.	✓
Asymptotic running time for building a database of n images is bounded by $\Omega(n)$ and $O(n^2)$.	✓
RAM usage for meta-data per image < 30 KB	✓ (see Section 7.2)
RAM usage upper bound is $O(n)$ for n images.	✓
The software should be platform and architecture independent.	✓

* This has been verified empirically but has not been formalised for the report.

† SIFT takes ≈ 30 seconds for an image of size 1024×768px, and clustering takes ≈ 0.5 seconds per image.

Table 7.7.2.: Table showing which non-functional requirements have been met.

8. Future Work

The work done thus far gives a glimpse of what may, ultimately, be a functioning system, but it also opens many areas that will require additional work. These areas fall into three major categories: Variations and extensions, evaluation and analysis, and software engineering.

8.1. Variations and Extensions

The pipeline design model allows various components to be replaced or augmented in the system. The SURF algorithm could be used instead of SIFT for feature extraction and description. Research shows that SURF is considerably faster whilst maintaining or exceeding descriptive power(Bauer, Sunderhauf and Protzel, 2007). There is also research showing that SIFT with key-points randomly generated over a certain distribution can produce better performance than searching for interest points (Nowak, Jurie and Triggs, 2006). According to the authors of the same paper, using a binary image signature is preferable to term-frequency or weighted term-frequency based signatures such as TF-IDF (Nowak, Jurie and Triggs, 2006). These ideas offer simplification to the process and are worth investigating.

Another area of future investigation is the descriptor similarity metric used by K-Means. Note that this is distinct from the signature similarity metric which has been investigated in Section 7.4. In this project the Squared Euclidean metric has been used as a descriptor metric, but this is not necessarily a good choice.

One problem in this system is with the computational performance of the retrieval operation. It is currently dependant on both the size of the database and the length of image signatures. It appears that there is a compromise to be had between descriptiveness and performance; as the signature length grows the time required for retrieval grows linearly. With database size the relationship also is linear. Unfortunately the descriptiveness of a signature is dependant on its length. Consequently, as the database grows, there will come a point when the signature length is not enough to correctly distinguish images; the vocabulary will become saturated. One solution would be to use Principle Component Analysis (PCA) to reduce the dimensionality of the signatures. An obvious problem with a naive application of PCA here is that the covariance matrix of the signatures is required. For signatures of length n the covariance matrix is of size n-by-n. For a typical signature length of over 10000 the memory requirements will be prohibitively large for a desktop computer. There do exist variants to PCA that can find the most significant eigenvectors without the covariance matrix, using a Hebbian Layer. The result could be a drastic reduction in signature dimensionality.

The whole system could be extended by adding a traditional classification module based on the image signatures. Although this is not in-line with the retrieval paradigm introduced in this report, it may be a way of improving results. In fact, the vocabulary tree method was originally proposed as a first pass technique to narrow down a search space

to some subset of the total documents, thus enabling slower but more accurate methods to become tractable. Candidates for classification systems include Neural Networks such as Multi-Layer Perceptrons, Radial Basis Functions, and Support Vector Machines. These technique, while representationally powerful, are weakly biased towards any one structure. If the regularities of the signatures could be better understood, this a priori knowledge could inform the choice classification algorithm. One problem with Neural Networks is that they are incredibly slow to train on high dimensional data, so additional pre-processing (such as PCA) would definitely be required.

Another possible improvement would be to use an inverted file structure at each node of the vocabulary tree, which references images that have descriptors quantised through that node (Nister and Stewenius, 2006). This technique is described in Section 5.2.3. For sparse signatures this could reduce the impact of very large image sets.

8.2. Evaluation and Analysis

There is clearly a need for further evaluation of the phenomena observed in the prototype implementation. It may even be premature to recommend extensions without a better understating of the current system. The evaluation in Chapter 7 demonstrates that the system works well under some conditions. However, each experiment seems to open up new questions. One of the most intriguing observations is in the signature similarly metric evaluation (Section 7.4). This showed that some metrics work well, while others don't work at all. It has been speculated that this is related to normalisation, but has no been confirmed. Further work is required.

All current evaluation has been done using a sub-set of Caltech-256 that has been shown to be easy (Griffin, Holub and Perona, 2007). While this is a reasonable starting point, it is hardly plausible in terms of an end-user application. Therefore some of the other data sets (discussed in Section 3.5) should be used. In particular the Flickr set which is likely to be very hard but most plausible.

Another approach to evaluation would be to analyse the sub-symbolic data used by the prototype program. In particular the use of a vocabulary tree raises a number of questions that may be answerable through careful inspection. For example, if this is a vocabulary tree then what does each vocabulary item actually represent? This is more in line with how Computational Linguist conduct their research and is appropriate considering that much of the technology has been borrowed from the field of Natural Language Processing (NLP).

There are some areas of this work that are novel, building on previous work but introducing new variations. It may be that some of it would be suitable for academic publication, but not before considerable additional effort has been spent on evaluation and analysis.

8.3. Software Engineering

There is some potential for a polished application in this work, and indeed that *is* the ultimate intention. One possible continuation would be to finalise a design for the system, and start engineering a stand-alone image organisation program. This would also be necessary for carrying out an end-user study, which would be a much more robust way of evaluating the solution in terms of real world performance.

There are a number of implementation alternatives besides writing an entire application. One attractive alternative is to write a plug-in for existing software. Most user orientated image organisation software provide an Application Programming Interface (API) to allow extension and modification of their basic functionality. It would be much easier to use such an API to augment an existing image management program with retrieval and assisted organisation capabilities. Further more, it would be preferable because users are already comfortable with the application.

9. Conclusion

Upon choosing the topic of Assisted Image Organisation for my third-year project, an initial investigation of the field gave rise to worry about the feasibility of the task. The project information website[1] emphasised the importance of producing a robust end-user application, as a software engineering exercise. Once it was made clear that this is not the only acceptable way to proceed, it was much easier to work towards a speculative implementation.

A literature review showed that most Computer Vision and Machine Learning techniques are totally inappropriate for this problem. Computer Vision systems are typically very task specific; effective within their constraints but with poor generalisation. The recent success in Machine Learning has focused on tractable data regularities such as cluster or area structure. Relational structure, of which images are a textbook example, has proved very difficult. The work of David Nistér, Josef Sivic (Nister and Stewenius, 2006; Sivic and Zisserman, 2003) borrows heavily from Natural Language Processing by viewing the image as a document. Each feature key-point is a visual word, and the surrounding patch is its context. From this we can construct a visual vocabulary by clustering based on context. The success of this project is down to the combined knowledge of these three fields, and is yet another example of the importance of interdisciplinary work.

The initial prototype proved to be unexpectedly proficient given the naivety of the approach. These early indications, however, where hard to justify because they where based on visual inspection. The construction of a suitable error metric (Section 5.5) enabled robust, quantitative evaluation. The experiments in Chapter 7 showed that within some constrains the prototype performed extremely well, considering the complexity of the task. Figure 7.3.2 showed that for large queries of 8 images or more, accuracy was approximately 75%. While this would be too low for unsupervised learning, it is perfectly acceptable in the supervised context of an organisation assistant. Further evaluation did not show a way to improve on this accuracy, but there are numerous extensions that are still to be investigated.

Over all the project has gone well. Given the opportunity to start over, there is little that could have been done differently. There where instances were too much time was spend on details that later turned out to be irrelevant, but that is the nature of research.

[1] The CSAI (Computer Science and Artificial Intelligence) project website is available at http://www.informatics.sussex.ac.uk/courses/csaiproj/

Bibliography

Bauer, J., Sunderhauf, N. and Protzel, P. (2007), Comparing several implementations of two recently published feature detectors. in: Proceedings of the International Conference on Intelligent and Autonomous Systems, IAV, Toulouse, France..

Bay, H. et al. (2008), Speeded-up robust features (SURF). Computer Vision and Image Understanding, 110:3, pp. 346–359.

Bay, H., Tuytelaars, T. and Van Gool, L. (2006), Surf: Speeded up robust features. Lecture notes in computer science, 3951, p. 404.

BCS (2004), The BCS Code of Good Practice. Version 1 edition. The British Computer Society, September 2004 ⟨URL: `http://www.bcs.org/upload/pdf/cop.pdf`⟩.

BCS (2006), Code of Conduct for BCS Members. The British Computer Society, November 2006 ⟨URL: `http://www.bcs.org/upload/pdf/conduct.pdf`⟩.

Bradshaw, B. (2000), Semantic based image retrieval: a probabilistic approach. in: Proceedings of the eighth ACM international conference on Multimedia. ACM New York, NY, USA, pp. 167–176.

Chatfield, C and Collins, A J (1980), Introduction to Multivariate Analysis. Chapman and Hall, London, UK.

Chum, O. et al. (2007), Scalable near identical image and shot detection. in: Proceedings of the 6th ACM international conference on Image and video retrieval. ACM, p. 556.

Chum, O., Philbin, J. and Zisserman, A. (2008), Near duplicate image detection: min-hash and tf-idf weighting. in: Proceedings of the British Machine Vision Conference. Volume 3,, p. 4.

Csurka, G. et al. (2004), Visual categorization with bags of keypoints. in: Workshop on Statistical Learning in Computer Vision, ECCV. Volume 1,, p. 22.

Dantzig, G.B. (1951), Application of the simplex method to a transportation problem. Activity analysis of production and allocation,, pp. 359–373.

Duda, R.O. and Hart, P.E. (1972), Use of the Hough transformation to detect lines and curves in pictures..

Fei-Fei, L., Fergus, R. and Perona, P. (2004), Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. IEEE. CVPR 2004, Workshop on Generative-Model Based Vision..

Flickr (2009a), Flickr API Services. flickr.com ⟨URL: `http://www.flickr.com/services/api/`⟩ – visited on 5th Decmber 2009.

Flickr (2009b), Flickr Most Popular Tags. flickr.com ⟨URL: `http://www.flickr.com/photos/tags/`⟩ – visited on 26th November 2009.

Forsyth, David A. and Ponce, Jean (2002), Computer Vision: A Modern Approach. Prentice Hall, ISBN 0130851981.

Fraundorfer, F., Stewenius, H. and Nister, D. (2007), A binning scheme for fast hard drive based image search. in: IEEE Conference on CVPR'07..

Gao, Y. and Fan, J. (2005), Semantic image classification with hierarchical feature subset selection. in: Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval. ACM, p. 142.

Gonzalez, Rafael C. and Woods, Richard E. (2007), Digital Image Processing (3rd Edition). Prentice Hall ⟨URL: `http://www.worldcat.org/isbn/013168728X`⟩, ISBN 013168728X.

Griffin, Griffin, Holub, Alex and Perona, Pietro (2007), Caltech-256 object category database. California Institute of Technology (7694). – Technical report ⟨URL: `http://authors.library.caltech.edu/7694`⟩.

Hawkins, Jeff and George, Dileep (2007), Hierarchical Temporal Memory: Concepts, Theory, and Terminology. Unpublished, ⟨URL: `http://www.numenta.com/Numenta_HTM_Concepts.pdf`⟩ – visited on 20th October 2009.

Jacob, E.K. (2004), Classification and categorization: A difference that makes a difference. Library trends, 52:3, pp. 515–540.

Jain, P., Kulis, B. and Grauman, K. (2008), Fast image search for learned metrics. in: IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008., pp. 1–8.

Jolliffe, I. T. (2002), Principal Component Analysis. 2nd edition. Springer ⟨URL: `http://www.worldcat.org/isbn/0387954422`⟩, ISBN 0387954422.

Kise, K., Noguchi, K. and Iwamura, M. (2007), Simple representation and approximate search of feature vectors for large-scale object recognition. Proc. BMVC2007,, pp. 182–191.

Kovesi, P. D. (2005), MATLAB and Octave Functions for Computer Vision and Image Processing. School of Computer Science & Software Engineering, The University of Western Australia ⟨URL: `http://www.csse.uwa.edu.au/~pk/research/matlabfns/`⟩ – visited on 5th Decmber 2009.

Lazebnik, S., Schmid, C. and Ponce, J. (2006), Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories..

Ling, Haibin and Okada, Kazunori (2007), An Efficient Earth Mover's Distance Algorithm for Robust Histogram Comparison. Pattern Analysis and Machine Intelligence, IEEE Transactions on;, 29:5, pp. 840–853.

Lowe, David G. (1999), Object Recognition from Local Scale-Invariant Features. Computer Vision, IEEE International Conference on, 2, pp. 1150–1157 vol.2 ⟨URL: `http://dx.doi.org/10.1109/ICCV.1999.790410`⟩, ISBN 0–7695–0164–8.

Lowe, D.G. (2004), Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60:2, pp. 91–110.

Marr, D. (1982), Vision: A computational investigation into the human representation and processing of visual information. Henry Holt and Co., Inc. New York, NY, USA.

Mikolajczyk, K. and Matas, J. (2007), Improving descriptors for fast tree matching by optimal linear projection. in: Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on., pp. 1–8.

Moosmann, F., Nowak, E. and Jurie, F. (2008), Randomized clustering forests for image classification. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30:9, pp. 1632–1646.

Moosmann, F., Triggs, B. and Jurie, F. (2007), Fast discriminative visual codebooks using randomized clustering forests. Advances in neural information processing systems, 19, p. 985.

Muja, M. and Lowe, D.G. (2008), Fast approximate nearest neighbors with automatic algorithm configuration. Preprint.

Nister, D. and Stewenius, H. (2006), Scalable Recognition with a Vocabulary Tree. ⟨URL: `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1641018`⟩, pp. 2161–2168.

Nowak, E., Jurie, F. and Triggs, B. (2006), Sampling strategies for bag-of-features image classification. Lecture Notes in Computer Science, 3954, p. 490.

OED (1989), The Oxford English Dictionary. 2nd edition. OED Online. Oxford University Press.

Omercevic, D., Drbohlav, O. and Leonardis, A. (2007), High-dimensional feature matching: employing the concept of meaningful nearest neighbors. in: Proc. IEEE Intl. Conf. Computer Vision, Rio de Janeiro, Brazil..

Ozuysal, M., Fua, P. and Lepetit, V. (2007), Fast keypoint recognition in ten lines of code. in: Conference on Computer Vision and Pattern Recognition. Volume 1,, pp. 1–8.

Pelleg, D. and Moore, A. (1999), Accelerating exact k-means algorithms with geometric reasoning. in: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM New York, NY, USA, pp. 277–281.

Pelleg, D. and Moore, A. (2000), X-means: Extending K-means with efficient estimation of the number of clusters. in: Proceedings of the Seventeenth International Conference on Machine Learning. San Francisco, pp. 727–734.

Philip, Brad, Updike, Paul and Weber, Markus (2001), Caltech-6 category object database. California Institute of Technology – Technical report ⟨URL: `http://www.vision.caltech.edu/Image_Datasets/Caltech6/`⟩.

Rodden, K. and Wood, K.R. (2003), How do people manage their digital photographs? in: Proceedings of the SIGCHI conference on Human factors in computing systems. ACM New York, NY, USA, pp. 409–416.

Ross, Sheldon (2005), A First Course in Probability. Prentice Hall ⟨URL: `http://www.worldcat.org/isbn/0132018179`⟩, ISBN 0132018179.

Rubner, Y., Tomasi, C. and Guibas, LJ (1998), A metric for distributions with applications to image databases. in: Computer Vision, 1998. Sixth International Conference on., pp. 59–66.

Russell, B.C. et al. (2008), LabelMe: a database and web-based tool for image annotation. International Journal of Computer Vision, 77:1, pp. 157–173.

Sivic, J. and Zisserman, A. (2008), Efficient visual search for objects in videos. PROCEEDINGS-IEEE, 96:4, p. 548.

Sivic, Josef and Zisserman, Andrew (2003), Video Google: A Text Retrieval Approach to Object Matching in Videos. in: ICCV '03: Proceedings of the Ninth IEEE

International Conference on Computer Vision. Washington, DC, USA: IEEE Computer Society ⟨URL: http://portal.acm.org/ft_gateway.cfm?id=946751&type=external&coll=Portal&dl=GUIDE&CFID=64967756&CFTOKEN=28271009⟩, ISBN 0–7695–1950–4, p. 1470.

Stockman, George and Shapiro, Linda G. (2001), Computer Vision. Upper Saddle River, NJ, USA: Prentice Hall PTR ⟨URL: http://portal.acm.org/citation.cfm?id=558008⟩, ISBN 0130307963.

Vedaldi, Andrea (2009), Personal website of Andrea Vedaldi. No address in ⟨URL: http://www.vlfeat.org/~vedaldi/⟩.

Wolpert, D.H. and Macready, W.G. (1995), No free lunch theorems for search. IEEE Transactions on Evolutionary Computation, 1:1, pp. 67–82.

Yang, J. et al. (2007), Evaluating bag-of-visual-words representations in scene classification. in: Proceedings of the international workshop on Workshop on multimedia information retrieval. ACM, p. 206.

A. Project Log

2^{nd} *October* 2009 Initial meeting with David Young. Discussed feasibility of the project and requested David Young for supervisor.

6^{th} *October* 2009 Meeting with David Young. Talked about project structure in terms outcomes and software used.

7^{th} *October* 2009 Studied chapter 22 from Forsyth and Ponce (2002), "Finding Templates Using Classifiers.".

8^{th} *October* 2009 Refreshed mathematics to understand better algorithms used in classification.

14^{th} *October* 2009 Read Hawkins and George (2007) to see if Hierarchical Temporal Memories (HTMs) where worth further investigation. The main problem is the closed nature of this research. HTMs are developed by a private company (Numenta Inc.) and some of their algorithms are proprietary. The repeated derogatory use of the term "prior art" in their paper is telling.

15^{th} *October* 2009 Read relevant chapters (1 - 23) from Forsyth and Ponce (2002) to recap a general overview of computer vision.

16^{th} *October* 2009 Worked through a simple plan on how one would go about making a general classifier.

20^{th} *October* 2009 First draft of the Project Proposal document.

21^{st} *October* 2009 Meeting with David Young. Discussed concerns about feasibility of the project.
Read Ross (2005) to refresh my knowledge of probability theory, which will be required for the implementation of statistical classifiers such as Naive Bayes.

22^{nd} *October* 2009 Researched SIFT by reading Lowe (1999) and other material.
Final version of the Project Proposal document. Update to re-enforce the speculative and research based nature of the project, as apposed to a software engineering project.

23^{rd} *October* 2009 Reading on principle component analysis (Jolliffe, 2002) and other multivariate analysis techniques C Chatfield and A J Collins (1980), *Introduction to Multivariate Analysis*. Chapman and Hall, London, UK. Need to do more background reading matrix algebra and elementary inference before I can get a handle on this.

25^{th} *October* 2009 Started planning out this Interim Report document.

$8^{th} - 9^{th}$ *November* 2009 Read Sivic and Zisserman (2003) and Nister and Stewenius (2006). These papers introduced the idea of classification as document retrieval problem.

12^{th} *Nov* 2009 Meeting with David Young. Proposed hierarchical clustering of SIFT descriptors to employ document retrieval as a classification tool.
Started doing extensive reading on papers relating to Nister and Stewenius (2006).

16^{th} *November* 2009 More reading.
Wrote a description of the ideas proposed on the 12^{th}.

$22^{nd} - 24^{th}$ *November* 2009 Build and tested a prototype.

25^{th} *November* 2009 Meeting with David Young. Demonstrated the prototype software.
Did some work the the Interim Report document.

28^{th} *November* 2009 Work on Interim Report.

29^{th} *November* 2009 Work on Interim Report.

30^{th} *November* 2009 Attempted to resolve problems with signature distance calculation.

1^{st} *December* 2009 Trying various forms of signature difference calculation: Euclidean, various p-norm distance, normalised euclidean distance, and Mahalanobis distance. All of them where worse than the highly naive and conceptually unintuitive calculation originally used.
Wrote up some of the Interim Report.

2^{nd} *December* 2009 Tried binary signature difference.
Work on the Interim Report.

$3^{rd} - 5^{th}$ *December* 2009 Work on the Interim Report.

20^{th} *January* 2010 Meeting with David Young. Discussed the Interim Report. Major areas of work are evaluation and similarity metrics.

20^{th} *January* 2010 Investigated Earth Movers Distance (EMD).

3^{rd} *February* 2010 Meeting with David Young.

3^{rd} *March* 2010 Meeting with David Young.

19^{th} *March* $- 11^{th}$ 2010 Work on Draft Report.

12^{th} *April* 2010 Submit Draft Report.

$13^{th} - 28^{th}$ *April* 2010 Work on Final Report.

29^{th} *April* 2010 Submit Final Report.