Text Based Approaches for Content-Based Image Retrieval on Large Image Collections

Peter Wilkins, Paul Ferguson, Alan F. Smeaton and Cathal Gurrin
Centre for Digital Video Processing
Adaptive Information Cluster
Dublin City University
Glasnevin, Dublin 9, Ireland
{pwilkins, pferguson, asmeaton, cgurrin}@computing.dcu.ie

Keywords

Large-scale content-based image search

ABSTRACT

As the growth of digital image collections continues so does the need for efficient content based searching of images capable of providing quality results within a search time that is acceptable to users who have grown used text search engine performance. Some existing techniques, whilst being capable of providing relevant results to a user's query will not scale up to very large image collections, the order of which will be in the millions. In this paper we propose a technique that uses text based IR methods for indexing MPEG-7 visual features (from the MPEG-7 XM) to perform rapid subset selection within large image collections. Our test collection consists of 750,000 images crawled from the SPIRIT collection (discussed in section 3) and a separate set of 1000 query images also from the SPIRIT collection. An initial experiment is presented to measure the accuracy of the subset generated for each query image by taking the top 100 results of the subset, and comparing those to the top 100 results derived from a complete ranking of the collection for that query image. Ranking is performed via L2 Minkowsky distance measures for both sets.

1. INTRODUCTION

Almost all work on practical implementations of image retrieval have used either image metadata such as anchor text, date, time or location, or have used low and mid-level features such as colour and texture. In the case of metadata-based image retrieval, large scale implementations such as Google Images [2] can be realised, but in the case of image-based image retrieval collection size is usually much more limited.

In this paper we propose a methodology to use Content-Based Image Retrieval (CBIR) on very large collection sizes. We propose a technique of applying text based IR techniques to index MPEG-7 extracted features (using the MPEG-7 eXperimentation Model), to perform rapid subset selection on large image collections. On this subset we then compute similarity measures in order to obtain a ranking of images. Using a naive approach of building an inverted index and basing retrieval on this will yield poor discrimination among images because the feature space in such image retrieval is too densely compacted with images. Our approach increases the feature space by increasing the number of terms by which an image can be described.

In [12] Quack et al. they proposed an image retrieval system for a set of up to three million images, based on visual features and collateral text, by means of clustering features and then utilising relevance feedback to improve precision. We propose a method for searching large image collections based on image-image similarity, without the need for clustering or calculation of a similarity matrix. This is currently the method employed in most current CBIR systems and as we will discuss in section 3 and is also the major impediment to the scaling up of these CBIR systems.

We begin the paper with a review of document (text) retrieval and image retrieval, contrasting their respective environments, following with related work. We then introduce the SPIRIT image collection and describe our approach for feature extraction. Following that we describe our subset selection algorithm including the preprocessing and indexing stages in section 5 and retrieval in section 6. A preliminary experiment is included and we finish with issues, future work and conclusions.

2. BACKGROUND

2.1 Document Retrieval

When searching collections of text documents we normally index each document by the words or word stems which occur within the document. Sometimes we include positional information about words, namely the ordinal position or offset within the document of each word and this is done to allow phrase or word adjacency searching. Document retrieval is accomplished by matching the words or word stems from a user query against words or word stems in the document and then weighting those occurrences using term frequencies, document lengths and other parameters.

Because the vocabulary of words which can be used in natural language text documents is so large, and because documents themselves normally are not trivially small, at

EWIMT '05 London, U.K.

least not as small as user queries, the virtual "space" in which documents occur is not overly dense for distinguishing documents from each other. For example, in our work on text-based information retrieval we use the SPIRIT collection of 94,552,870 web pages [8] crawled directly from the internet in 2001, as described in section 3. As noted in [3] the size of the vocabulary for a collection of text documents follows Heaps law [6] and with an average document length of 456 terms, the number of index terms should be approximately 73,600,000. Although this is a huge number of terms, most of them correspond to numeric and mis-spellings and have very low frequencies of occurrence, and the actual number of content-bearing terms, or dimensions in the feature space, is much less.

During document retrieval, the issue of short queries, of the order of 2 or 3 query terms creates problems, in that while (longer) documents can be distinguished against each other a short query does not offer as much potential for discrimination between documents and so information retrieval research has developed techniques including relevance feedback, pseudo-relevance feedback, latent semantic indexing, and others. In addition, although there are semantic dependencies between the words used to index documents, a consequence of the dependencies between words in natural language, for the most part we ignore these dependencies in information retrieval and we treat the indexing terms as if they were independent of each other.

2.2 Motivation for Large-scale Image Retrieval

If we now look at the characteristics for retrieval of images we find a very different environment. We are concerned here with raw image matching, not with retrieval based on meta-data such as anchortexts or descriptions [2] but with retrieving an image from a collection based on the fact that the image somehow matches or looks like another image. Conventionally, image retrieval can be based on matching low-level features such as colour, texture or edges, or mid to high-level semantic features such as 'indoor/outdoor', 'person', 'vegetation', 'water', etc. For low-level features, the continuous spectrum of colours, textures and edge directions is normally quantised into a relatively small number of bins, of the order of some dozens, each of which forms a dimension in the feature space while each of the mid- and high-level features also forms another individual dimension in this space. So for example, if we quantise colour and texture into 80 bins each, and we quantise edge directions into 10 bins, and we have 10 mid-level semantic feature detectors then we have a total of $80 + 80 + 10 + 10 = 180$ dimensions in our feature space. Collectively the number of dimensions in the "space" occupied by images is relatively small, especially when contrasted with the number of features in text document retrieval, and image retrieval also generally ignores any dependencies that may exist between such features.

Let us now examine one common set of circumstances for image retrieval, namely retrieval from a collection of personal photos. In a related project we have collected nearly 10,000 personal photos of people's travel, holidays, family events, etc. [5]. These have been analysed using our implementation of the MPEG-7 eXperimentation Model (XM) [9] as described in section 4. In addition, all our photos have GPS location and local weather conditions information, and we have implementations of mid-level feature detectors including face(s), indoor/outdoor, buildings, etc. Using the MPEG-7 XM we can use these low-level features to create a multi-dimensional feature space in which each photo can be placed. We have augmented the colour/texture/edge space by up to 692 further category or feature values, the most frequent of which is *daylight* (5867 times) and the least frequent of which includes *photo taken in misty weather, photo taken in Schenley Heights Pittsburgh* and many others, each of which occurs only once. Because the number of photos is only of the order of some thousands, the feature space is still not overly dense and photos are distinct enough from each other to allow acceptable levels of image retrieval. Furthermore there are no issues of 'short' queries as with document retrieval since query images themselves are analysed using exactly the same feature analysis as other images in the collection and so have as many dimensions.

But what if the collection of images is more than several thousands, but several millions of images? This certainly would not be the case for an individual's personal photo collection but might be the case for the photos from a community of users, or from a commercial photo archive, or even from the web. In such a scenario of a much larger collection of images then the context for retrieval changes completely as the feature space then becomes more densely populated with images. Because the characteristics of the feature space are so different to document retrieval, although the number of images/documents may be comparable, we cannot re-use techniques used in large-scale document retrieval in retrieval from large numbers of images. As acknowledged by [7] current image search systems rely on high-dimensional visual features which are difficult to search efficiently, and resolving this problem is the key to scaling up existing CBIR systems to deal with realistic, web sized collection of images. In this paper we examine some options open to us for addressing this problem but before we do that in the next section we shall give an outline of the collection of images we are working with, and then we shall give a summary of the low-level features from the MPEG-7 XM that we are using.

2.3 Related Work

Related work resides in areas of image clustering, classification and text retrieval of image data. These approaches are concerned with adding further discrimination into the feature space at either a low or semantic level. Image clustering and classification has been explored in many systems, e.g. [13][15]. The major difference between our work and these techniques is that for the most part the clusters or classifications are created at indexing time. Our subset selection occurs at retrieval time and the subset generated is dependant on the low level features of the query image. All subsets generated are dynamic, however as these are all based on low-level features no semantic information could be derived.

Other systems e.g. [14] have taken advantage of text retrieval research, and make use of such concepts as inverted indexes, relevance feedback and ranking metrics incorporating frequency measures[16]. Our work takes advantage of inverted index structures and term identification techniques via variations on n-grams[16]. However we do not use any text based ranking metrics or relevance feedback.

3. SPIRIT COLLECTION

As previously mentioned, the SPIRIT collection is a col-

lection of over 94.5 million web pages. From this large collection of web pages we have identified over 125 million unique image URLs, of which we have crawled all available images (over 50 million) and stored these images locally. It is with indexing such a large collection of images that motivates this research and the SPIRIT collection provides us with an ideally large collection of readily available images.

Our previous work at generating similarity matrices for image-image similarity calculations relied on the fact that the collections we were working with were rather small, of the order of about 33,000 images. With a collection of this size it is feasible to store a whole $N \times N$ similarity matrix on a single desktop computer. Traditional matrix reduction techniques that rely on removing large numbers of zero values from a matrix will not be applicable in this case as virtually all similarity values between images will be non-zero. Alternatively, simply reducing the number of similar images similar to each image in the top 1,000 or the top 100 will result in an incomplete matrix. Rather the approach employed should attempt to minimise the effect of any matrix reduction.

Were one to take the traditional approach, we estimate that a triangular similarity matrix for the SPIRIT collection of 50 million images would require 9 petabytes (9 million gigabytes) of memory, or approximately 23,000 large hard drives. In addition, the processing time required to generate such a triangular matrix, assuming 50,000 similarity calculations a second (MPEG-7 XM on a fast workstation), would require almost 800 years of (Pentium 4) processor time to complete. Clearly this is not feasible and an alternative technique is required, such as is outlined in this paper in order to use the MPEG-7 XM to extract features from images and provide retrieval facilities using these features.

4. FEATURE EXTRACTION

The extraction of features from our image collection was performed by our feature extraction toolbox which we have used as part of our participation in TRECVid. TRECVid is an international exercise which benchmarks the effectiveness of video retrieval systems. As part of TRECVid the organisers run a shot boundary detection process on the video collection, for each shot within a video they extract a keyframe image which is representative of that shot. These keyframes become the standard image reference set for the video collection.

In order to support keyframe matching for our video shot retrieval application as part of TRECVid in 2004 [4], we developed a tool which processed all keyframes from the collection using the feature descriptors described below. These descriptors (based on the MPEG-7 XM) were developed within the context of the aceToolbox, a toolbox of low-level audio-visual analysis tools being developed as part of DCU's participation in the EU aceMedia project [1].

The following description of the toolbox and visual features is referenced from our earlier TRECVid work [4]:

In this first version of the toolbox, colour feature grouping is performed by Recursive Shortest Spanning Tree (RSST). The original RSST algorithm is a relatively simple and fast region-growing method. It starts from pixel level and iteratively merges regions (two regions per iteration) according to the distance calculated using colour features and region size. The process stops when the desired number of regions are obtained. For our experiments we can process images using any of the following four descriptors.

- An **Edge Histogram Descriptor (EHD)** is designed to capture the spatial distribution of edges by dividing the image into 4x4 subimages (16 non-overlapping blocks) and edges are then categorized into 5 types (0, 45, 90, 135 and nondirectional) in each block. The output is a 5 bin histogram for each block, giving a total of 5x16 = 80 histogram bins.
- A **Local Colour Descriptor (Colour Layout - CLD)** is a compact and resolution-invariant representation of colour in an image. The colour information of an image is partitioned in 64 (8x8) blocks; second, the representative colour of each block is determined by using the average colour in each block.
- A **Global Colour Descriptor (Scalable Colour - SCD)** measures colour distribution over an entire image. It is defined in the hue-saturation-value (HSV) colour space and produces a 256 bin colour histogram, normalised, non-linearly mapped into a four-bit integer value, and then encoded by a Haar transform to obtain a 32 bin histogram.
- A **Homogenous Texture Descriptor (HDT)** describes directionality, coarseness, and regularity of patterns in images. It is computed by first filtering the image with a bank of orientation and scale sensitive (Gabor) filters, and then computing the mean and standard deviation of the filtered outputs in the frequency domain. In this work we only use the mean values to compute the similarity between the images.

More details on these descriptors can be found in [10].

The overall outcome of employing these features would be to generate four separate triangular similarity matrices, with the similarity being estimated by using L2 Minkowsky distance. The TRECVid 2004 collection consisted of 33,367 images. However, scaling such a similarity matrix approach up to the 50 million images from our SPIRIT collection will not be possible (as outlined in section 3), and a new methodology is required, as proposed below.

5. INVERTED INDEX FEATURE REPRESENTATION

5.1 INDEXING OVERVIEW

In our work we explored two approaches to creating our inverted index representation, these being a frequency based approach and a positional based approach. To illustrate the fundamental differences between the two, let us take an example colour structure document:

```
56 233 5 56 56 255 0 0 0 37 94
```

The above document consists of 7 unique terms. In a frequency based index we record the occurrences of each term in the document. For instance, term '0' would have a frequency of 3. Conversely for a positional based index we are more interested in what position in the document does the term occur. If we start counting our positions from 0, we can say that the term '233' appears at position '1'.

Regardless of the type of index to be created, there are several steps that must be followed for the creation of an

inverted index. First there is a pre-processing step, which processes the candidate images through the MPEG-7 XM, then takes the resulting raw XML documents and converts them into a format for rapid indexing, such as the previous example. The second phase is the actual indexing itself which creates the inverted index file, and the lexicon which specifies what terms exist in the index and if they do at what location within the file. Finally there is retrieval which allows us to query the index and obtain our subset.

5.2 PRE-PROCESSING

The first step involved in our indexing process is to extract the features from the candidate images utilising the MPEG-7 XM. For each feature extracted from the set of images, an XML document is produced which contains for each image the extracted feature output. An example entry for the edge histogram feature for a single image would appear as:

```
<Descriptor xsi:type = "EdgeHistogramType">
    <BinCounts>5  1  1  0  1  4  4  2  1  3  3  6
    2  6  1  2  7...[till 80]
    </BinCounts>
</Descriptor>
```

The meaning of these 80 values was described earlier in section 4. However, given that each feature represents different representations of an image, the corresponding outputs are not overly similar. For instance the following examples are extracts of outputs for Colour Structure, Colour Layout and Homogenous Texture respectively:

```
<MultimediaContent xsi:type = "ImageType">
    <Image>
        <VisualDescriptor xsi:type =
            "ColorStructureType" colorQuant = "1">
            <Values>10  8  0  0  128  10  0  0
            255 166  [.. further elements]
            </Values>
        </VisualDescriptor>
    </Image>
</MultimediaContent>

<Descriptor xsi:type = "ColorLayoutType">
    <YDCCoeff>7</YDCCoeff>
    <CbDCCoeff>24</CbDCCoeff>
    <CrDCCoeff>36</CrDCCoeff>
    <YACCoeff5>14  24  12  15  11 </YACCoeff5>
    <CbACCoeff2>18  12 </CbACCoeff2>
    <CrACCoeff2>15  19 </CrACCoeff2>
</Descriptor>

<Descriptor xsi:type = "HomogeneousTextureType">
    <Average>51</Average>
    <StandardDeviation>52</StandardDeviation>
    <Energy>159  161  169  200  152  147  145
    [... further elements]</Energy>
</Descriptor>
```

At the end of this process we have one XML file per feature for the collection. These XML documents now provide a textual representation of the image content which we can now exploit.

Our final phase in this step is to process each of the XML files into a stripped down file, that contains one document per line, and aggregates all the integer values for a document to appear on that line. Once this is complete the data is ready to be indexed.

5.3 INDEXING

This indexing phase is primarily concerned with, for each document, taking it's integer array of values, identifying the terms within this array depending on the indexing strategy (frequency or positional) and recording this information to disk.

The first step in the creation of any inverted index is term identification for the current document being processed. If we look at data for the Colour Structure feature, we can observe that the possible value for any term will be within the range of 0-255, providing only 256 unique terms. For Edge Histogram data this problem is even worse with the possible value for a given term being within the range of 0-8 providing only 9 unique terms. This means that our lexicon of unique terms will now be severely limited if we take terms as being integer values delineated by white space. To compare this to text, we would expect a similarly sized text index to have a lexicon ranging into the hundreds of thousands of unique terms. Therefore we need to create a greater number of unique terms for text retrieval techniques to be effective. Our approach is to create what we are calling Term Grams (TGrams).

TGrams are similar to n-grams in that both are used to analyse the text to identify new terms. While n-grams work at the character level, TGrams work on the word or term level. A TGram is a new term that is the concatenation of adjacent terms to the current term being processed. The number of terms that will be concatenated is determined by the length of TGram we are trying to create. This is best illustrated by an example. If we take a snippet of Colour Structure data:

```
10 8 0 0 128 10 0 0 255 166
```

we can first extract our standard single terms, which would provide an identical list as above. Next we create our 2 term length TGrams, which would result in the following terms:

```
10_8, 8_0, 0_0, 0_128, 128_10, 10_0, 0_0, 0_255, 255_16
```

It should be noted that the TGram creation is overlapping, and that we insert an underscore character between the concatenated terms. The reason for this is that it's important to differentiate between the integer value '108' and the two length TGram '10_8'. For our term identification process we create TGrams of length 2 through to 6, meaning that for our earlier example we would produce the following output:

Length 3:

```
10_8_0, 8_0_0, 0_0_128, 0_128_10, 128_10_0, 10_0_0,
0_0_255, 0_255_166
```

Length 4:

```
10_8_0_0, 8_0_0_128, 0_0_128_10, 0_128_10_0, 128_10_0_0,
10_0_0_255, 0_0_255_166
```

Length 5:

```
10_8_0_0_128, 8_0_0_128_10, 0_0_128_10_0, 0_128_10_0_0,
128_10_0_0_255, 10_0_0_255_166
```

Length 6:

10_8_0_0_128_10, 8_0_0_128_10_0, 0_0_128_10_0_0, 0_128_10_0_0_255, 128_10_0_0_255_166

This process creates more complex and varied terms than what we began with, and adds a degree of size to the lexicon to aid with retrieval. However we can still generate another increase in size if we take into account either positional or frequency information depending on the index that we are creating.

If we are creating a frequency based index, we take for each document its unique terms and count the number of occurrences of that term within the document. Taking our initial example document, and examining the TGrams of length 2 we created, we obtain the frequency data shown in Table 1.

Table 1: Frequency Data

Term:	Frequency:
10_8	1
8_0	1
0_0	2
0_128	1

We record this data in our index for this document. However we can also record this data as unique terms to store in our lexicon to provide greater term differentiation again. To store this data in the lexicon we create a new term that is the aggregation of the TGram and the frequency data. We use a hyphen to delineated between the two types of data so that the term isn't confused as being a TGram of greater size (e.g. differentiate between the TGram 10_8_1 and the TGram 10_8-1 which occurs once). If we take the above example we would then have created the following new terms:

10_8-1, 8_0-1, 0_0-2, 0_128-1

A similar approach is taken with the positional index, except that instead of recording the frequency of the term, we record at what position it occurs within the document. Again taking the earlier TGram of length 2, we would get the data as shown in Table 2.

Table 2: Position Data

Term:	Position:
10_8	0
8_0	1
0_0	2
0_128	3

This would generate the following terms to be added to the lexicon:

10_8-0, 8_0-1, 0_0-2, 0_128-3

6. RETRIEVAL

Like any IR system, retrieval is initiated by a query and in this case the query is in the form of an image. The query image is processed by the MPEG-7 XM and its features extracted. This feature output is then processed through the same TGram creation process as described in the previous section. We can select the length of TGram's to generate, so that we can query with only 2 length terms, 3 length terms etc. The specific TGram generation steps will depend on what index is being queried (positional or frequency).

Once query TGrams are created, examining the index begins. For each TGram we first check if it appears in the lexicon, if it does not then there are no documents that contain that particular TGram, so we return nothing. Alternatively if there is a lexicon entry we retrieve the all matching documents from the index and add these into our result area.

Given that TGrams incorporate either positional or frequency information, the retrieval from the index will be quite selective. For instance if our query TGram is frequency based, then the TGram '10_8-1' will only retrieve from the index documents that contain a pair of integers '10 8' that appears exactly once in the document. Similarly for a positional based index the query TGram '10_8-1' will only retrieve a pair of integers '10 8' where the pair start at position '1' within that document.

Once these iterations are complete we are left with our subset of candidate documents which can be ranked using whichever ranking scheme has been implemented. Again it should be emphasized at this point that this system is not designed to create a new ranking methodology, only a subset selection process which prunes the amount of documents to be ranked.

The final addition we have made to the retrieval system is the capability for the index to be queried again with TGrams of a smaller length if the initial query did not generate enough query documents. We refer to this mechanism as a 'stepped' query, where the query will specify the initial length of TGram to query by, and if not enough documents are returned, to query the index again with a specified TGram of a shorter length. The number of documents that are required before the 'step down' is invoked is referred to as the 'stepping threshold' and is currently set to 100. For example, a query may specify that it wishes to query initially with TGrams of length '6', but if not enough documents are returned to query again with TGrams of length '3'. The initial query would lookup the index using TGrams of length '6'. If 100 or less documents are returned the system will re-initiate the query using TGrams of length '3'.

Therefore our system's main purpose is to utilise the inverted index structure to obtain a subset of candidate images from the collection which are most likely to be somewhat similar to the query image. At present we execute an L2 similarity measure between images in this subset and the query image to obtain our final ranked output for the query. This last step could be replaced with any ranking mechanism.

7. EXPERIMENT

To explore whether this technique of index creation and retrieval would scale up we ran an experiment to test the performance of our inverted index system versus that of a complete similarity matrix to evaluate how effective the inverted index system is in identifying the subset of most likely similar images. For this experiment the dataset used was 750,000 images from the SPIRIT collection, as discussed in Section 3 and the query images are comprised from a sepa-

rate set of 1000 images, also from the SPIRIT collection.

The resulting similarity matrix that was produced was therefore 750,000x1000 in size. For this experiment we constructed an index from the Edge Histogram and Scalable Colour features. The similarity measure employed for generating the matrices and ranking of the subset is L2.

The experiment therefore is that for each query image we query both the similarity matrix and the inverted index for the top 100 results. Taking the results of the similarity matrix as our ground truth (being a ranking of the entire collection), we compare the results to the returned subset. We measure the percentage of documents returned in the top 100 from the subset to the matrix top 100 for that query image, in other words we are calculating the precision, where the matrix results are considered correct.

We also record the size of the subset that is returned and express it as a percentage reduction of the overall set. For example if a query returns a subset of 75,000 from the 750,000 set, then this subset would be expressed as being a 90% reduction of the initial set.

The purpose of the experiment is to determine how accurate our inverted index system is in selecting an appropriate subset for similarity ranking and in doing so eliminating the problem of high-dimensionality as discussed in section 2.2. As both systems utilise L2 ranking, the score of an image appearing in either list will be identical, therefore we are only evaluating to what extent the two sets overlap.

The following series of experiments was executed on 750,000 set of image data for Edge Histogram and Scalable Colour data. Both frequency and positional indexes were created, using TGrams of lengths 1 - 6. The queries that were executed were using frequency and positional TGrams of lengths 1 - 6, as well as 'stepped' queries of combinations of 5 and 2, 5 and 3, 6 and 2, and 6 and 3.

7.1 INDEX TIME AND LEXICON SIZE

The creation of the two indexes of 750,000 for both Colour Structure and Edge Histogram, of both frequency and positional types took can be seen in Table 3

Table 3: Indexing Time

Feature:	Time (min):
Colour Structure Frequency	216
Edge Histogram Frequency	218
Colour Structure Positional	220
Edge Histogram Positional	234

Figures 1 and 2 illustrate the rise in lexicon growth per TGram used. We can observe from these graphs that as the length of the TGram increases so to does the number of unique terms that it generates. More generally we can see that positional indexes contain more terms than frequency indexes, and that Colour Structure generates more terms than Edge Histogram. This is not unexpected given that the raw terms for Edge Histogram have a range of 0-8 whereas Colour Structure has a range of 0-255. What is surprising though is the massive number of terms that long positional TGrams generated as opposed to frequency based.

7.2 QUERY RUN RESULTS

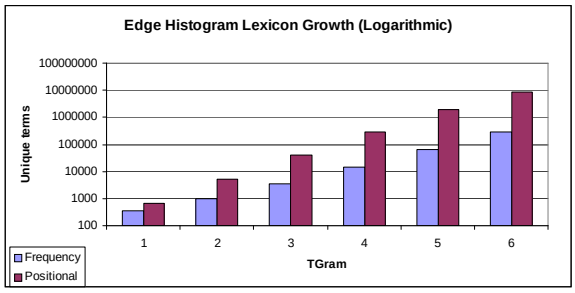

Figure 1: Edge Histogram Lexicon Growth

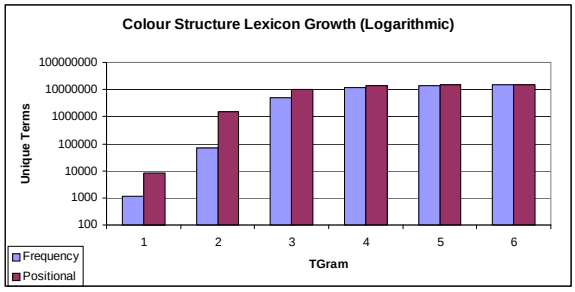

Figure 2: Colour Structure Lexicon Growth.

As mentioned earlier, the results we present are the percentage overlap our subset results have to that of a complete L2 ranking of the entire collection, and the percentage to which we have returned a reduced subset size. For both measures, the higher percentage indicates greater performance.

The first results to be presented here are the results of the queries to both Edge Histogram and Colour Structure frequency based indexes as seen in Figures 4 and 3. These results can be contrasted to those generated from the results for positional indexes, as seen in Figures 6 and 5.

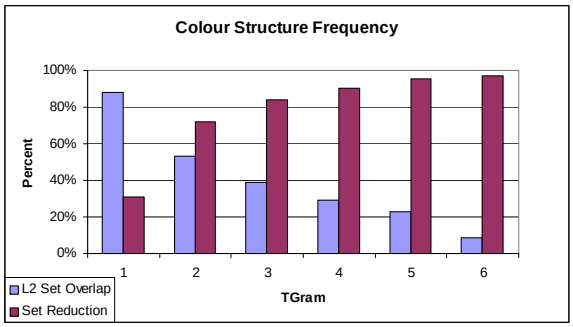

Figure 3: Colour Structure Frequency

The first major observation to make is that for some particular reason, the Edge Histogram Frequency results (Figure 4) are at odds with the trends established by the other indexes. Why this is the case isn't immediately clear but is assumed to be an artefact of the data, and may be because a frequency based index is not as selective in its subset reduction techniques.

On the whole positional indices achieve a greater level of performance than the frequency based indices, with the

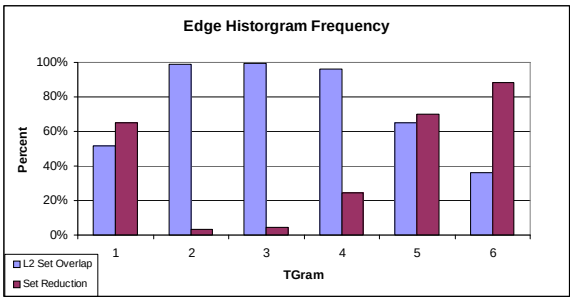

Figure 4: Edge Histogram Frequency

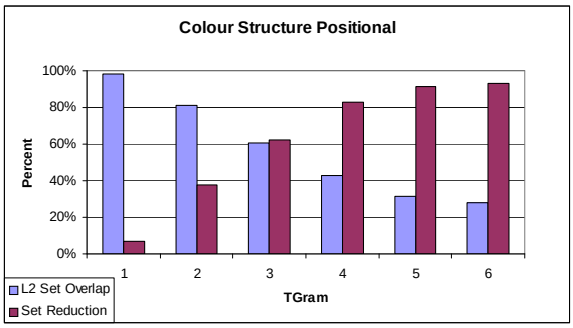

Figure 5: Colour Structure Positional

best level of trade-off occurring with TGrams of length 4. In these cases we are achieving a 42%-51% overlap but with a reduction of the set in the order of 82%-88% area. It is interesting to note that despite the two different types of data in Edge Histogram and Colour Structure, a comparable level of performance can be achieved.

The retrieval times for these results are presented in Tables 4 and 5 (where retrieval time is measured as the average time in seconds to generate the TGrams from the query, lookup the index and place the results into a set).

Table 4: Timings for frequency queries (seconds)

Frequency	1	2	3	4	5	6
Edge Histogram	0.87	12.12	14.68	4.43	1.26	0.65
Colour Structure	3.14	0.72	0.39	0.25	0.15	0.10

In an attempt to improve our retrieval time further, we attempted 'stepped' runs as described earlier. However based on earlier experiments, we found that the stepping threshold of 100 documents was not being reached for the Edge Histogram index, therefore for the Edge Histogram experiments, thresholds of 1000 and 5000 were used. The results for these runs appears in Figures 7 and 8, with timing data presented in Table 6.

This result turned out to be the most surprising as we were able to achieve further significant performance increases. In

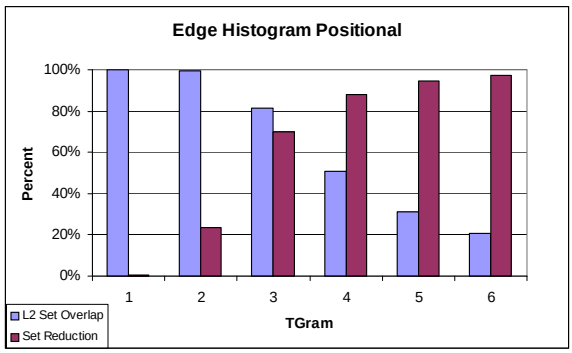

Figure 6: Edge Histogram Positional

Table 5: Timings for positional queries (seconds)

Positional	1	2	3	4	5	6
Edge Histogram	23.74	4.95	1.74	0.96	0.71	0.61
Colour Structure	9.03	3.69	1.87	0.99	0.67	0.54

particular the combination of TGrams of length 5 and 2 yielded high overlap results between 47% and 53% yet still achieved subset reduction in the order of 78% to 82%. What is also observable is that depending on the search scenario, different types of combinations may yield more suitable results. In particular if we are more interested in larger subset reduction, then a combination of 6 and 3 length TGrams would provide up to a 93% reduction in the set size whilst still achieving a 35% overlap.

8. ISSUES, FUTURE WORK AND CONCLUSIONS

We believe that the approach to reducing the search space for image retrieval described here has potential, as these results show that a fair degree of overlap can be achieved in a reduced subset that can be retrieved in a timely manner. As with any information retrieval task the effectiveness of the system will be determined by what the user is attempting to retrieve. A system that employs our aforementioned mechanisms for rapid subset selection would be most applicable to an ad hoc retrieval scenario where a user is looking for some general answers that match their query, and would not care about achieving 100% recall.

A major issue confronting this system is that by using a text based approach, we will retrieve documents that only match some part of the query document. However existing similarity techniques such as L2 will rank documents as being very similar even if they do not share any terms in common. In these instances our approach will fail as we require an overlap for the document to be retrieved. However as noted earlier, depending on the retrieval task and the size of the collection, the returned results may be adequate to fullfill the users tasks.

It would be interesting to compare our results to that of more contemporary ranking techniques (and fusion models) such as [11] to see how this approach compares. To do this though would require the current approach to be extended to incorporate the other MPEG-7 features that we regularly make use of, including Colour Layout and Homogenous Texture.

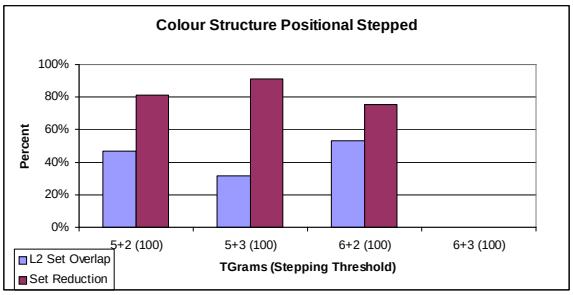

Figure 7: Colour Structure Stepped

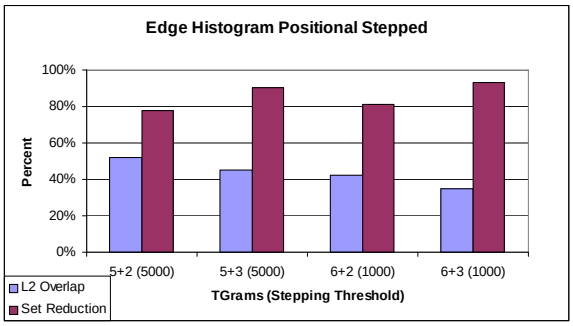

Figure 8: Edge Histogram Stepped

The final major issue, also acknowledged in [12] is at this stage there is no large scale image retrieval evaluation platforms that can be utilised to provide feedback on system performance.

Future work to be undertaken within this area is to determine if there is any advantage that can be achieved to aid ranking of query's results using this mechanism, and if in doing so we can get an improvement in precision over current ranking measures such as L2. However again as noted earlier, this work would be difficult to undertake as no mechanism for the evaluation of results from a large scale image search currently exists.

Acknowledgments

This work was supported by Science Foundation Ireland under grant 03/IN.3/I361. We are grateful for the support of our colleagues from the aceMedia project who provided us with output from the aceToolbox image analysis tooklit.

9. REFERENCES

[1] The AceMedia project, available at http://www.acemedia.org.

[2] The Google image search page, available at http://images.google.com.

[3] F. Cacheda, V. Plachouras, and I. Ounis. Performance analysis of distributed architectures to index one Terabyte of text. In *Lecture Notes in Computer Science*, volume 2997, pages 394 – 408, March 2004.

[4] E. Cooke, P. Ferguson, G. Gaughan, C. Gurrin, G. J. F. Jones, H. L. Borgne, H. Lee, S. Marlow, K. McDonald, M. McHugh, N. Murphy, N. E. O. Connor, N. O'Hare, S. Rothwell, A. F. Smeaton, and P. Wilkins. TRECVID 2004 experiments in Dublin City University. In *Proceedings of TRECVID 2004*, November 2004.

[5] C. Gurrin, G. Jones, H. Lee, N. O'Hare, A. F. Smeaton, and N. Murphy. Digital photos: Where and when. In *ACM Multimedia 2005, Singapore, 6-12 November 2005*, 2005.

[6] H. Heaps. *Information Retrieval: Computational and Theoretical Aspects*. Academic Press, 1978.

[7] P. Howarth and S. Rüger. Trading precision for speed: Localised similarity functions. In *Int'l Conf on Image and Video Retrieval (CIVR, Singapore, Jul 2005)*, pages 415–424. Springer LNCS 3568, 2004.

[8] H. Joho and M. Sanderson. The SPIRIT collection: An overview of a large web collection. *SIGIR Forum*, 38(2):57–61, December 2004.

[9] B. Manjunath, P. Salembier, and T. Sikora, editors. *Introduction to MPEG-7: Multimedia Content Description Language*. Wiley, 2002.

[10] B. S. Manjunath, J. R. Ohm, V. Vasudevan, and A. Yamada. Color and texture description. In *IEEE Trans. On Circuits and Systems for Video Technology*, June 2001.

[11] K. McDonald and A. F. Smeaton. A comparison of score, rank and probability-based fusion methods for video shot retrieval. In *Proceedings of CIVR 2005*, 2005.

[12] T. Quack, U. Mönich, L. Thiele, and B. Manjunath. Cortina: A system for large-scale, content-based web image retrieval. In *Proceedings of ACM Multimedia*, 2004.

[13] G. Sheikholeslami, W. Chang, and A. Zhang. Semantic clustering and querying on heterogeneous features for visual data. In *Proceedings of ACM Multimedia*, 1998.

[14] D. M. Squire, W. Müller, H. Müller, and J. Raki. Content-based query of image databases, inspirations from text retrieval: inverted files, frequency-based weights and relevance feedback. In *Technical Report Number 98.04, Computer Vision Group, University of Geneva*, 1998.

[15] A. Vailaya, M. A. T. Figueiredo, A. K. Jain, and H.-J. Zhang. Image classification for content-based indexing. In *IEEE Transactions on Image Processing, 10:1*, 2001.

[16] I. Witten, A. Moffat, and T. C. Bell. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann Publishers Inc., 1999.

Table 6: Timings for stepped queries

Stepped	5+2	5+3	6+2	6+3
Edge Histogram	1.36	0.91	1.25	0.79
Colour Structure	0.93	0.72	2.19	n/a