Trading Precision for Speed: Localised Similarity Functions

Peter Howarth and Stefan Rüger

Department of Computing, South Kensington Campus,
Imperial College London, London SW7 2AZ, UK
{peter.howarth, s.rueger}@imperial.ac.uk

Abstract. We have generalised a class of similarity measures that are designed to address the problems associated with indexing high-dimensional feature space. The features are stored and indexed component wise. For each dimension we retrieve only those objects close the query point and then apply a local distance function to this subset. Thus we can dramatically reduce the amount of data looked at. We have evaluated these distance measures within a content-based image retrieval (CBIR) framework to determine the trade-off between the percentage of the data retrieved and the precision. Our results show that up to 90% of the data can be ignored whilst maintaining, and in some cases improving, retrieval performance.

1 Introduction

CBIR methods aim to provide a way for users to search and browse large image and video collections. The quantity of multimedia data available through digital libraries and on the web is huge and and is rapidly expanding. To ensure the scalability of search and browsing systems it is essential that fast and efficient indexing methods are available.

Current image search systems rely on high-dimensional visual features extracted from images to quantify a facet of the image and then a similarity measure to rank the images in relation to a query. It is the high dimensionality that makes feature space difficult to search efficiently. Tackling this problem is the key to scaling up existing CBIR systems to realistic and useful amounts of data.

In this paper we have generalised a class of similarity measures that address this problem by drastically reducing the amount of data examined and thus retrieved from disk. The core of this paper is to assess the trade-off between the amount of data looked at and retrieval performance.

The problems associated with indexing high-dimensional features are discussed in Section 2, together with approaches for tackling these issues. Section 3 describes our similarity measure. Sections 4 and 5 describe the experimental set-up and results. Finally, conclusions and future work are in Section 6.

2 Background

2.1 The Effect of High Dimensionality

The term *curse of dimensionality* was first used by Bellman [1]. It refers to the way that the behaviour we understand and use in 2 or 3 dimensions breaks down as the dimensionality of a space increases. This section considers the effect it has on similarity measures and indexing.

Let us say that our feature space is described by a p-dimensional unit hypercube containing uniformly distributed data points. Given a query point, how much of the range of each dimension must we consider to capture a proportion of the data q. To enclose a fraction q of the unit volume the length will be $r = q^{1/p}$. If we are trying to enclose 1% of the data then in 10 dimensions this means we must consider 63% of the range of each dimension, for 100 dimensions this increases to 95% and for 500 dimensions it is 99%.

Of course real-world data, such as image features, are unlikely to be uniformly distributed and may exist on a lower dimensional manifold. This will alleviate some of the symptoms of the curse, however significant effects for nearest neighbour searching and indexing remain. High-dimensional feature space is very sparsely populated, so it becomes hard to partition the data effectively. This is significant for tree based structures. Secondly, the notion of nearest neighbour has less meaning. As dimensionality increases it was shown by Beyer el al. [2], with similar arguments to those above, that all points will tend to the same distance from a query point. This has the ultimate effect of making the nearest neighbour problem ill defined.

2.2 Indexing Approaches

A significant bottleneck when searching any large database is the amount of data that needs to be loaded from disk. This is because disk access is slow. Linear indexes have been optimised with the B-Tree. In addition for high-dimensional features there is also the time to compute the similarity measure. Practical indexing approaches have to address both of these. This section gives an overview of some methods used.

It is likely that a real feature space may have an intrinsic dimensionality lower than the apparent data space. Dimensionality reduction methods aim to extract significant information into lower dimensions. Principal component analysis is the commonest technique and it is often used in combination with other methods. PCA works well but has drawbacks for indexing. Its complexity can make it impractical for very large datasets with high dimensionality and there are difficulties with incrementally adding data.

A significant class of methods partition the feature space or data points into tree structures. The first of these for multidimensional space was the R-tree developed by Guttman [3]. There have been many variants of this and they have proved successful in certain circumstances. However, Weber et al. [4] showed that above a certain dimensionality all tree structures would collapse to a linear scan.

This led them to develop a vector approximation technique called the VA-file. This accepts the fact that the linear scan is inevitable and attempts to optimise it using compression. They achieve times of 12.5–25% of a linear scan. This performance level is often used as a benchmark for other systems.

Approximate nearest neighbour approaches relax the constraint of finding exact results to speed up search. Nene and Nayar's [5] method recovers the best neighbour if it is within ϵ of the query point. Beis and Lowe [6] developed a variant of the k-d tree using a best-bin-first algorithm. They used this to efficiently retrieve the nearest or a very close neighbour in a shape indexing context.

Aggarwal et al. investigated the behaviour of L_p-norm distance measures in high dimensional space [7], specifically looking at the impact on nearest-neighbour search. They found that the lower norms gave more meaningful results and extended the idea to fractional values of p. This improved performance further. They work by increasing the significance of points local to the query and reducing the noise from distant points. We extended the analysis of fractional distance measures to visual features [8] and found them to significantly improve retrieval performance.

Then, there are approaches that vertically decompose the feature space. Each dimension of the feature is held and searched separately. This gives a very flexible approach as dimensions can be treated differently depending on their significance. For instance the inverted VA-file of Müller and Henrich [9] stores each dimension at different quantisation levels and only retrieves at the accuracy needed dependent on the query. The BOND system developed by de Vries et al. [10] uses a branch-and-bound algorithm so that data in later dimensions can be discarded.

Finally, the Aggarwal and Yu's iGrid [11] and the bitmap index of Cha [12] work with vertically decomposed features and use only the part of each dimension close to the query point to generate a similarity value. It is this idea that we have built upon for our similarity function.

3 Localised Similarity Functions

The aim of this similarity function is twofold: to give an effective similarity measure for high-dimensional features and to only examine a small percentage of the data when doing so. It is not aiming to be an approximation to another similarity measure, but a meaningful measure in its own right.

We have already found from our own work [8] that fractional distance measures give more meaningful nearest neighbour search when applied to high-dimensional visual features. The premise for this improved performance is that they emphasise dimensions that are close whilst reducing the noise from distant dimensions. This idea can be extended to similarity measures that consider only the locality of the query point in each dimension. They are effectively giving a weight of zero to those distant points and selecting a sample of the least noisy information for the measure.

We have defined the following similarity measure between a test vector X and a query point Q,

$$\mathrm{Sim}(X,Q) = \sum_{i \in K(X,Q,k)} \left(1 - \frac{d(x_i, q_i)}{z}\right), \tag{1}$$

where the set $K(X,Q,k)$ is the set of objects local to the query point. The function $d(x,q)$ is the local distance measure used and z is a normalising factor.

This is a generalisation of similarity measures used in [11, 12] as it allows the use of any distance measure. Defining the similarity measure in this way highlights the flexibility available. There are options available to trade speed for precision by varying the local distance function or the selection of the local neighbourhood. Previous work used a fixed number of equally populated partitions to define the localities. We allowed the selection of the exact neighbourhood and to be able to vary this at query time. This gives more flexibility.

The local distance measure $d(x,q)$ will effect the local topology around the query point and therefore the search results. In addition, the complexity of the function will have an impact on the computational time. We are interested in the trade-off between these. Other work has used the Manhattan distance measure in evaluations. We are interested in varying the functions and have used L$_p$-norms, fractional measures, ranking and a voting function.

A diagram showing these functions is in Fig. 1. The voting function allocates a value of 1 to all object within the local set and 0 to others. The obvious advantage of this over other measures is simplicity. It is the outermost function on the figure. The ranking function is not shown, the distance for this is based purely on the ranking of objects from the query point.

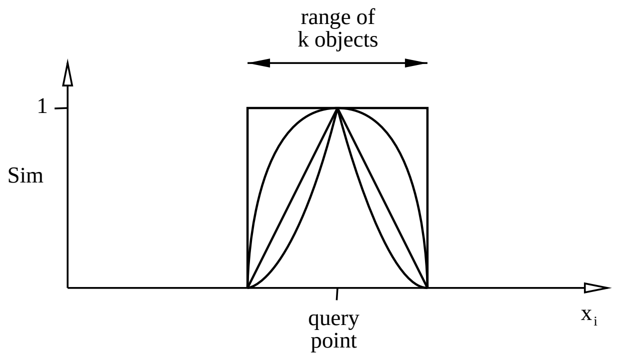

Fig. 1. Similarity functions

The second parameter in Eq. (1) is the function $K(X,Q,k)$ selecting the local set of objects from each dimension. This can be a selection of the nearest k points, or all the points within a certain distance. We chose the first option as it enables the selection of a fixed percentage of the data.

If we consider the voting function described above, each dimension will have k votes and there will be a total of kp votes cast across all dimensions. An object receives a vote from a dimension if it is in the set of k nearest objects to the query point in that dimension. If v is the total number of votes an object receives across all dimension it will be the upper bound on the similarity for that object.

The distribution of v across the data set will affect the discriminatory power of the similarity measure. Consider a data set with N objects. With $k = 1$ there will be few objects occurring in more than one dimension. As k increases the objects with higher v's will increase up until $k = N$, where all objects will be found in every dimension. If we are using the voting distance measure we can see that discriminatory power will increase with k to a maximum and then decrease to random at $k = N$. The other distance functions will have added discriminatory power. However, it is likely their maximum overall performance will be strongly correlated with the maximum performance of the voting measure.

One interesting issue with this function is where certain dimensions in a feature have a value that occurs frequently throughout the set of feature vectors. An example is for histogram features where particular bins can be empty most of the time, meaning that zero is a typical value. If the query point also had the same value then the resulting similarity has little meaning. We decided to exclude dimensions where this occurred in a similar way to the Jaccard measure, discarding information where the query point has the "ordinary" value. Initial tests showed that this outperformed other options.

4 Experiments

Experiments were set up in a CBIR framework to investigate the following:

- The trade-off between the fraction of the dataset looked at and retrieval performance
- The effect of the local distance function on retrieval performance
- The specific performance characteristics of the voting measure with increasing proportion of data.

4.1 Experimental Set-up

We use mean average precision (m.a.p.) as a measure of performance of similarity measures. Whilst m.a.p. can be criticised for not being related to a specific user task it does give a good overall measure of performance that trades off between precision and recall. M.a.p. is widely adopted for information retrieval and we therefore feel justified in its use.

It is recognised with image retrieval that the data set used can have a large influence on results of any experiments and the resultant conclusions. To ensure that our results were not just a feature of the data set used we ran experiments using two different collections, Corel and TRECVID, described below. This enabled us to validate our results and draw conclusions about the general applicability across two very different collections.

We used a subset of Corel that was created by Pickering and Rüger [13] to evaluate visual features. 6,192 Corel images were carefully selected to give 63 categories that were visually similar internally, but different from each other. This was then split into two sets. The first, a set of 1,548 images, was used to query the remaining 4,644 images. From the query collection we generated single and multiple image queries across all categories. The number of images per query was varied from 1 to 6; for each number we created 630 queries. This made 3,780 in total. The results shown in Section 5 are the mean average precision across these queries.

The TRECVID 2003 collection is widely used. It comprises of 32,318 key-frames from the TRECVID 2003 video collection [14]. These were taken from ABC and CNN news broadcasts. The search task specified for TRECVID consists of 25 topics. For each topic several example images were given as a query. The published relevance judgements for these topics can be used to evaluate the retrieval performance across experiments. The collection is much larger than Corel but has drawbacks mainly due to the limited number of queries. This can make results from this collection sensitive to minor changes. However, it is a realistic task and provides a good contrast to Corel.

For multiple image queries we used the k-nearest neighbour (k-nn) retrieval approach. Previous work in our group [13] has demonstrated that this outperforms the vector space model. k-nn is based on the idea that, given positive and negative example images, the test images can be classified according to their proximity to these examples. A version of the distance weighted k-nn approach was used [15]. Positive examples (P) are supplied as the query and negative examples (N) randomly selected from the collection. To rank an image i in the collection we identify those images in P and N that are amongst the k-nearest neighbours of i. Using these neighbours we determine the dissimilarity

$$D(i) = \frac{\sum\limits_{n \in N} (\mathrm{dist}(i,n) + \epsilon)^{-1}}{\sum\limits_{p \in P} (\mathrm{dist}(i,p) + \epsilon)^{-1} + \epsilon} \quad , \tag{2}$$

where ϵ is a small positive constant to prevent division by zero. A value of $k = 40$ was used for our experiments.

4.2 Visual Features

We used a range of visual features with dimensionality from 512 to 30. Full details of the features are available in [13]. In brief they are: **RGB**, a joint colour histogram defined in RGB colour-space with 512 bins; **HSV**, a joint colour histogram with 205 dimensions defined in the hue, saturation and value colour-space; **HDS**, the MPEG-7 colour structure descriptor, which has 184 non uniformly quantised bins; **MarginalRGB**, a colour histogram with 10 bins allocated to each of the 3 colour channels. It has 30 dimensions and was included as a lower dimensional vector.

5 Results and Analysis

We carried out an extensive range of experiments. This Section contains a representative sample of results. These are shown as pairs of graphs, Corel and TRECVID, of mean average precision retrieval against the fraction of data selected by the similarity function. This is the proportion of the entire dataset that would need to be loaded from disk and can be taken as an indicator of the speed-up in query time[1]. We are most interested in the region of the graphs covering less than 40% of the data as these will give a significant speed-up.

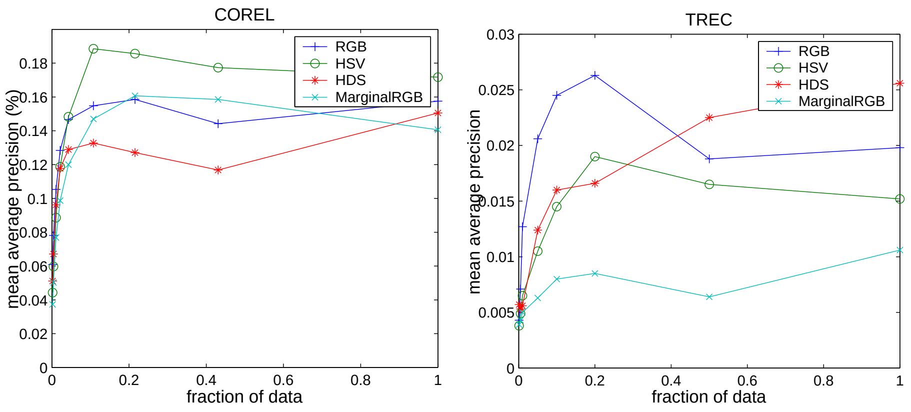

Fig. 2. Performance using L_1 as the local distance measure

Figure 2 shows the performance of the similarity function, across the different features, using a Manhattan local distance function. From the graphs we can see that — although there is some variation — the retrieval performance remains the same down to 10–20% of the data. After this point m.a.p starts to drop off. For several of the features performance actually increases as less data is examined. Overall this is an exciting result as it means that we can discard up to 90% of the database while maintaining retrieval performance.

The graphs in Figure 3 show the retrieval performance of the voting distance measure across all the features. They show the underlying performance impact of the proportion of each dimension used (equivalent to the number of votes cast). As expected from 50–100% the performance degrades rapidly as each dimension votes for all the objects.

If we compare these graphs with Figure 2 it is clear that each feature has the same characteristic shape between 0–40% of the data. This supports the notion that it is the number of votes cast for each object that dominates the performance of this similarity measure. The specific local distance measure will have an impact on top of this.

[1] In our implementation each sorted dimension is stored contiguously on disk. This enables us to retrieve the minimum numbers of disk blocks required

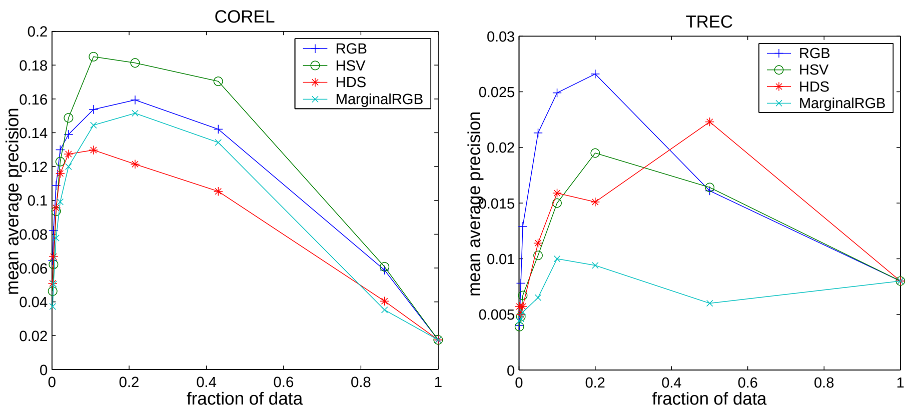

Fig. 3. Performance using voting distance measure

The final pair of graphs in Figure 4 shows the performance of the different local distance measures with the RGB feature. This was chosen as it has the highest dimensionality. It shows the same characteristic behaviour of all the other features and as such is a good representative.

Examining the region of interest, around 10–40% of the data, the first observation is that all the distance measures perform similarly. With Corel the performance curves are tightly grouped, only the rank function stands out. TRECVID shows a slightly wider spread. The other features, not shown, had similar graphs although the ranking of the distance measures varied.

Overall, the local distance measure does have an impact on performance that varies from feature to feature. This would be exploitable in a retrieval system. The voting distance measure shows surprisingly good performance: it is always within or close to the top group. It is a very simple function; using it would enable the further speed-up of search as there is no need to retrieve and process accurate distances. It therefore appears a good candidate for the fastest search method.

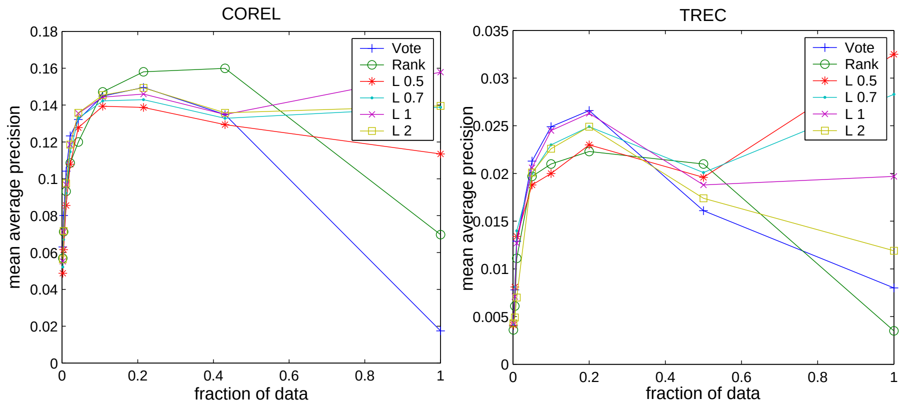

Fig. 4. RGB performance for all distance functions

6 Conclusions and Future Work

Our work has shown that the trade-off between the amount of data examined and the retrieval performance is exploitable for speeding up high-dimensional indexing. Experimental results indicate that we can ignore up to 90% of the data and maintain or improve retrieval performance based on a m.a.p. measure. We have demonstrated this across two very different datasets.

When comparing with the VA-file, the ability of localised similarity functions to use around 10% of the dataset puts them above the top end of VA-file performance. Some additional storage may be required with this method depending on the local distance function. This optimum level of 10% contrasts with the findings of Aggarwal et al. in [11]. They were not using visual features and found that a proportion of $1/p$ of each dimension gave acceptable results. This would be only 1% of the data with a 100 dimensional feature. Our experimental results in the area of CBIR show a significant loss in performance by this level.

Our experiments revealed that the overriding factor within this similarity measure is the number of votes that an object receives, i.e. the number of dimensions that an object occurs in the set local to the query point. This can be found using a voting local distance function which is simple and efficient. Varying the local distance functions can improve retrieval performance for specific features. This could be useful to squeeze the last bit of performance for a query, if more time is available.

In addition to reducing the amount of data examined, this style of distance function has many advantages for practical indexing. The decomposed feature values are stored independently of any distance measure. It lends itself to parallelisation and being distributed across a number of processors and disks. Furthermore at query time it is possible to vary the both the local distance function and the locality itself. This can be done dimension by dimension, dependent on the relevance to the query. Entire dimensions can be discarded. This has the potential of enabling further significant pruning of the data and dramatic speed up of search times. Our future work will investigate opportunities for this using feature selection techniques and exploiting information in multiple image queries and relevance feedback.

References

1. R Bellman. *Adaptive Control Processes.* Princeton University Press, 1961.
2. K Beyer, J Goldstein, R Ramakrishnan, and U Shaft. When is "nearest neighbor" meaningful?". In *Int'l Conf on Database Theory (ICDT 99), Jerusalem, Israel*, pages 217–235. Springer LNCS 1540, 1999.
3. A Guttman. R-trees: a dynamic index structure for spatial searching. In *Proc of ACM SIGMOD Int'l Conf on Management of Data*, pages 47–57, 1984.
4. R Weber, H-J Stock, and S Blott. A quantative analysis and performance study for similarity search methods in high-dimensional space. In *VLDB Conf Proc*, pages 194–205, 1998.
5. S Nene and S Nayar. A simple algorithm for nearest neighbor search in high dimensions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(9):989–1003, 1997.

6. J Beis and D Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *CVPR '97: Proc of the 1997 Conf on Computer Vision and Pattern Recognition (CVPR '97)*, page 1000. IEEE Computer Society, 1997.
7. C Aggarwal, A Hinneburg, and D Keim. On the surprising behavior of distance metrics in high dimensional space. In *Int'l Conf on Database Theory 2001*, pages 420–434. Springer LNCS 1973, 2001.
8. P Howarth and S Rüger. Fractional distance measures for content-based image retrieval. In *European Conf on Information Retrieval (ECIR, Santiago de Compostela, Spain, Mar 2005)*, pages 447–456. Springer LNCS 3408, 2005.
9. W Müller and A Henrich. Faster exact histogram intersection on large data collections using inverted VA-files. In *Int'l Conf on Image and Video Retrieval, CIVR 2004, Dublin, Ireland*, pages 455–463. Springer LNCS 3115, 2004.
10. A de Vries, N Mamoulis, N Nes, and M Kersten. Efficient k-nn search on vertically decomposed data. In *Proc of the 2002 ACM SIGMOD Int'l Conf on Management of Data*, pages 322–333. ACM Press, 2002.
11. C Aggarwal and P Yu. The IGrid index: reversing the dimensionality curse for similarity indexing in high dimensional space. In *Knowledge Discovery and Data Mining*, pages 119–129, 2000.
12. G-H Cha. Bitmap indexing method for complex similarity queries with relevance feedback. In *MMDB '03: Proc of ACM Int'l Workshop on Multimedia Databases*, pages 55–62. ACM Press, 2003.
13. M Pickering and S Rüger. Evaluation of key-frame based retrieval techniques for video. *Computer Vision and Image Understanding*, 92(1):217–235, 2003.
14. A Smeaton, W Kraaij, and P Over. TRECVID 2003 — An introduction. In *TRECVID 2003 Workshop*, pages 1–10, 2003.
15. T Mitchell. *Machine Learning*. McGraw Hill, 1997.

Acknowledgements: This work was partially supported by the EPSRC, UK.