データベースシステム 113－28
（１９９７．７．１５）

A Comprehensive Image Similarity Retrieval System that utilizes Multiple Feature Vectors in High Dimensional Space

K. Curtis, N. Taniguchi, J. Nakagawa and M. Yamamuro
{kate, nota, junichi, masashi}@dq.isl.ntt.co.jp

Database Systems Laboratory
NTT Information and Communication Systems Laboratories
1-1 Hikarinooka, Yokosuka-shi, Kanagawa 239 Japan

HyperMatch is a general purpose image retrieval engine made up of three components; a conventional database, a set of high dimensional image feature indices and a coordinator. Various new and improved high dimensional indexing techniques are discussed for their applicability to different features and distance measurements. The role of the coordinator is to combine the search results from individual indices into an overall measure of similarity. Experiments reveals the coordinator must choose search parameters based on the number of features to be combined, data set size and the number of similar images requested. Important future development work includes index improvement and system evaluation.

多次元特徴ベクトルを用いた画像類似検索エンジン

キャサリン　カーティス　谷口　展郎　中川　純一　山室　雅司
ＮＴＴ情報通信研究所

大量画像をデータベースから画像の特徴によって検索する類似画像検索のためのデータアクセス法とそれを実現した検索エンジン*HyperMatch*について述べる．*HyperMatch*は一般データ検索部，特徴量による類似検索を行う多次元データ検索部，及びこれらの検索部の調整をして最終検索結果を返すコーディネータからなる．コーディータは，特徴量の種類，データサイズ，結果要求数などのパラメタを考慮するべきことを実験結果から示す．

1 Introduction

Rapid advances in technology have led to a surge in the popularity of multimedia for a wide variety of uses. In particular, this paper is concerned with the growth in demand for very large image data repositories. Possible repositories vary from a commercial collection of stock photography to a representation of the collection of an art gallery to a time series of satellite images.

The content and purpose of each repository can be very different, but one of the uniting factors of all large image databases is that in order for such systems to be useful, it must be possible to easily retrieve images from the repository according to a variety of search criteria. It is imperative that the search facilities be simple to use, have a fast response time and provide accurate results.

This paper describes a system named *HyperMatch* that has been specifically designed to address the issue of image retrieval in an integrated manner for diverse image repositories. It relies heavily on the intelligent combination of existing database technology with a selection of indexing structures that provide fast search capabilities over image data. The system is designed for deployment in a client/server environment.

2 Background

Image database systems have generated significant interest in recent years. In particular, Stonebraker [1] and Jain [2] have presented well formed proposals for the general requirements for the storage architecture and user interface of such systems. They have highlighted the difficulty of translating user requests into representations that can be used to query the image database and the advantages of an object-oriented approach to data storage and query language development. Wu [3] described a content based retrieval engine that recognizes the need for a variety of index structure types. The design of *HyperMatch* represents an attempt to specify more fully how such indexes should be combined to achieve the fastest possible search without compromising accuracy.

Broadly speaking there are two approaches to image searching; keyword retrieval and similarity retrieval. Key word retrieval relies on the mostly human input of textual tags that are stored in a database together with the image. Queries are then used to try and locate images via the content of the tags. Keyword retrieval alone is not practical for large databases. Relying on human input introduces inconsistencies and errors due to the differences in judgment

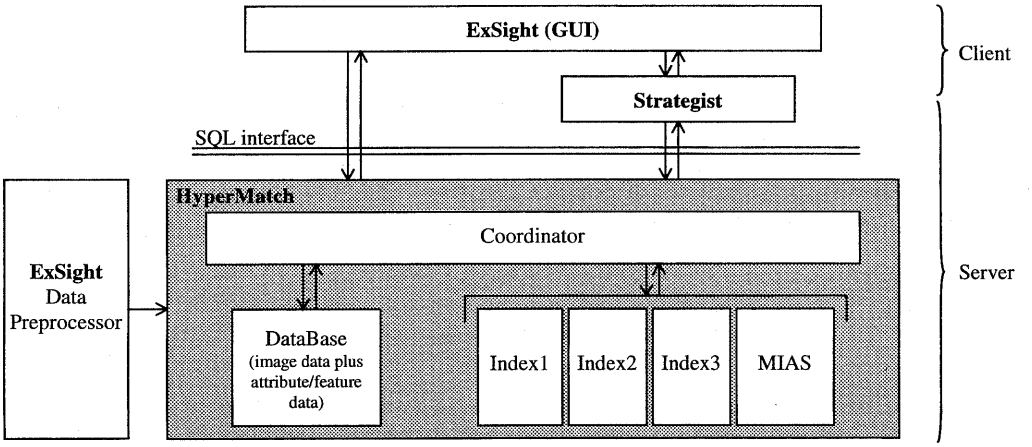

Figure 1. The *HyperMatch* Engine

that exist between any two humans. Similarity searching has been proposed as an alternative. Images can be represented by a set of feature vectors and advances in image processing have meant that these vectors can be extracted automatically from raw image data and stored together with the original image in a database system. For example, red, green, blue, hue, intensity, saturation, shape and position are possible features for describing an image or subsection thereof. Similarity retrieval is the process of comparing images by measuring the similarity of their feature vectors, usually using some type of vector distance measurement such as Euclidean or Manhattan distance.

In either type of retrieval it is essential to use a data indexing method to speed up the process. It is not feasible to conduct an exhaustive search on the entire contents as we expect the database to contain hundred of thousands of images, each with as many as ten or more feature vectors associated with it. Keyword retrieval generally relies on well established indexing techniques for one dimensional data. However, feature vectors tend to be of high dimension (typical ranges are from 16 to 256 dimensions) and so special high dimensional indexing methods are required. The most promising advances in high dimensional indexing have been achieved through modification of the k-d tree and R-tree structures [4].

3 The *HyperMatch* Engine

The *HyperMatch* engine, shown in Figure 1, is an image data retrieval server that has been designed for use with a variety of client applications. In particular it is used by the *ExSight* [5], an innovative graphical interface for the specification and manipulation of retrieval requests and result sets. The basic function of *HyperMatch* is to provide the client with a set of images ranked in order of their similarity to the key image provided by the client.

During the initialization phase of an installation of *HyperMatch* all image data is subject to preprocessing for the extraction of component image objects from the original images. Each object is further preprocessed to extract a set of feature vectors.

There are three basic components found in the *HyperMatch* engine. First, a conventional database system that can be used to store the image with its associated attribute and feature vector data. Second, a set of data indexes that can be used to achieve high performance searching in high dimensional feature vector space. Third, a coordinator that is responsible for executing searches on individual indices and optimizing, for speed and accuracy, the combination of individual index results to form one result set ranked by overall similarity. All communication with the *HyperMatch* engine takes place though an SQL like interface.

3.1 Conventional Database Component

The database look-up component of the system is used to locate the image data and other data derived from the image data such as feature vectors, key words and image identifiers. All communication with a database is carried out using a standard database protocol (e.g. ODBC) so that there is significant flexibility in accommodating existing image databases or a particular user's requirements. In fact, there is no requirement that the data be stored in only one database since the coordinator will be provided with enough information to retrieve needed data from the correct information source. The database can also provide standard indexing and search procedures for the more traditional data elements such as image identifiers and key words.

3.2 High Dimensional Indices

Conventional database indexing methods cannot be applied in high dimensions and so special high dimensional indexing must be provided in order to search the feature vector spaces. Our research has focused on two distinct types of

index; variants of the R-tree data structure and the multiple inverted array system (MIAS).

The R-tree Structure

An R-tree [6] is an extension of the B-tree indexing method for multidimensional objects. A complex data element is represented by its minimum bounding hyper-rectangle (MBR) and a tree is constructed by recursively grouping MBRs which can in turn be represented by an enclosing MBR.

Currently, one of the most successful implementation of the R-tree for indexing high dimensional data appears to be the VAMSplit R-tree proposed by White[7]. This tree is created by recursively choosing splits of the data set using the maximum variance dimension and then choosing a split that is approximately the median. The tree has a fast construction time and does take advantage of examining the entire data set before choosing the first split.

Once the index tree has been built, a search algorithm must be implemented to find the k-nearest neighbors to a given data element. The branch and bound search algorithm for the nearest neighbor searching technique in [8] gives the basis for the search process for almost all the R-tree variations. They detail the concept of using minimum and maximum distance metrics between the key data element and the minimum bounding hyper rectangles of a set of non-leaf nodes to establish a prioritized search order among the nodes.

Memory vs. Disk Based Implementation

Given the size of the image database and the number of different indices that will be made available, it is unrealistic to assume that the indices will all fit in memory. Therefore we developed a version of the VAMSplit R-tree index that resides partially in memory and partially on disk. This is achieved by allowing the system to choose the number of levels of the tree that are read into memory. If a node at a deeper level is required during the search process then it is read from disk.

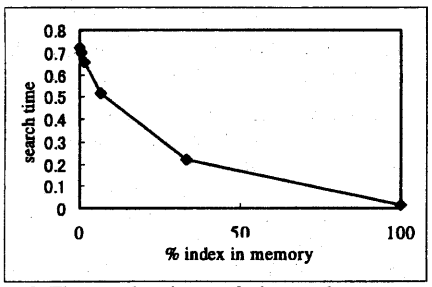

Figure 2. The speed up in search time as the percentage of index stored in memory increases

The proportion of the index stored on disk can be altered dynamically by the coordinator in response to prevailing patterns of usage. If one or more indices is used heavily and there is room in memory, then it makes sense for those indices to reside wholly in memory. Allowing the maximum amount of index in memory is obviously the most ideal solution and so the index is capable of immediate performance improvement if more memory is able to be made available to it. In general it is hoped that the flexible design will allow for more intelligent divisions between the in memory and on disk portions of an index to be developed. Figure 2 shows how the performance of an index can be improved as a greater percentage of the nodes of the tree are stored in memory.

C-tree

The C-tree [9] was developed in order to take advantage of situations where the construction speed of the index can be sacrificed in order to speed up retrieval performance. It uses a non-hierarchical clustering method to attempt to optimize an R-tree initially constructed using one of the standard construction methods. It is created as an unbalanced tree in an attempt to keep the cluster radii minimized. However, this unbalance can lead to difficulties in the disk based implementation of the index and initial tests did not show significant improvement over the VAM Split R-tree in these situations. However, we believe that the C-tree may be used advantageously with less traditional distance measures such as a multi-resolution distance metric.

Multi-resolution R-tree

We believe that it is possible to take the type, construction method and likely distance measure of a particular feature vector into account in order to design an efficient tree index for that feature. For example, the construction of a 256 dimension color feature vector involves recording the number of pixels in the image for each level of the color scale. Images with close, but not exactly the same color levels, will therefore be found on adjacent dimensions and a dimension reduction procedure that simply groups adjacent dimensions together will preserve similarity.

This observation led to the development of a multi-resolution R-tree where each level of the tree represents the feature vector space at a different dimensional granularity. By taking advantage of calculating distances to nodes in lower dimensions in order to reduce the search space, significant time savings can be achieved. Construction of the tree is straightforward, with the VAMSplit R-tree algorithm used. The only change is that dimension reduction is carried out on the data set at each level before the maximum split dimension and MBRs are calculated. The search algorithm carries out equivalent dimensional reduction on the key vector so that the most likely node for continued searching can be identified at each level.

MIAS

The Multiple Inverted Array Structure (MIAS) [10] is a data structure which has one inverted array for each dimensional axis of the feature space. The advantage of the data structure is that it is completely independent of any kind of similarity measure. Thus the query response can be improved by selecting the optimal search method for a given query. Moreover since each inverted array can be handled independently it is suitable for parallel/distributed processing. The structure is trivial to construct and dynamic

insertion and deletion is also much easier than for a tree structure. The search algorithm takes advantage of a type of dimension reduction to achieve efficiency.

3.3 Coordinator Functionality

The coordinator is responsible for ensuring the optimal execution of the search requested from the client or strategist. It is aware of the existence of the database system and the external index structures. As part of the initialization phase of the system the coordinator is given information concerning the database schema, the types of index available and the relationships between the index information and the data stored in the database. For example, the coordinator needs to be able to match the results of an index search to actual image data contained in the database.

HyperMatch expects requests to find similar images based on a particular set of features. For example, a request to find images similar to a key image based on the red, green and blue feature vectors. The role of the coordinator is to combine the results from each individual index into a coherent single set of results. The overall similarity of an image is determined by a function of its similarity distance to each feature. Typically this function is a linear combination of the individual feature distances with each distance assigned a weight to describe its importance in the overall distance calculation.

The result set returned by each individual feature vector will be different and the ranking of images in the individual feature space is not the same as its ranking using the overall distance measurement. For example, an image can be ranked below the top twenty in all individual vector spaces but be ranked inside the top twenty for the total distance. Therefore it is obvious that simply retrieving and combining the top k ranked images from individual feature spaces in order to achieve the top k images overall is not satisfactory and will lead to inaccurate results.

However there is no currently available method for deciding what is the correct number of image rankings that should be retrieved in each feature space in order to achieve an overall accurate result. Therefore we conducted several experiments in order to show that the number of rankings required from individual indices depends on the number of images required in the final result set, the size of the data set and the number of feature vectors that are to be combined. It will also be dependent on the weight assigned to the feature and probably to a lesser extent on the nature of the data set.

The data set used to conduct the experiments consisted of real image data. The number of images requested by the user is denoted by k. The number of images retrieved from each individual vector space is denoted as r. It is assumed that r is constant over all feature spaces in the same search, even though this may eventually not be the most optimal solution. N refers to the size of the data set. The percentage of correct results featured in each of the graphs refers to the percentage of the k ranked images found by the system that are correct when compared to an exhaustive search of the images, i.e. $r = N$. Finally N_f refers to the number of feature spaces/indices considered in the search. The feature spaces considered in the experiment were red, green, blue, hue, saturation and intensity.

The speed at which an index search is completed is proportional to the value of r, the number of most similar images that should be found in order to guarantee a particular level of correctness in the final combined result set. Therefore the coordinator always attempts to minimise the value of r for all searches.

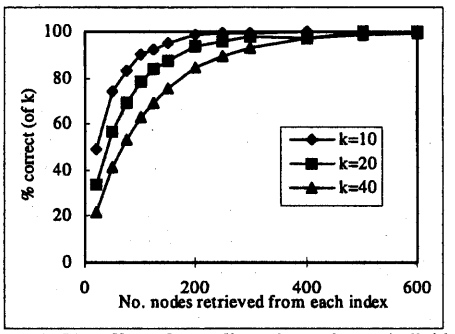

Figure 3. The effect of overall result set size on individual index retrieval set size and accuracy

Figure 3 shows the effect of increasing the value of k while N=11620 and N_f =3 remain constant. It is clear that as k increases, r must also be increased in order to maintain the same percentage correctness in the result.

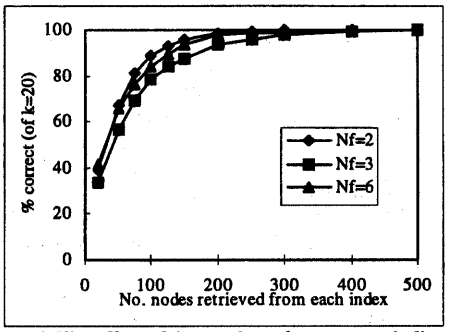

Figure 4. The effect of the number of component indices on individual index retrieval set size and accuracy

Figure 4 shows the effect of increasing N_f. when N=11620 and k=20 remain fixed. As N_f increases from 2 to 3, r must be increased to maintain accuracy. This is as expected as increasing the number of feature spaces increases the likelihood that an image close to the key in one feature space will be discarded because it is far from the key image in all other spaces. However, as N_f increase to 6, a decrease in the required value of r is observed. This can be explained by considering that as the number of feature spaces is increased and r is kept constant, the total number of images returned by the feature spaces increase as a percentage of the total image set size. Therefor it becomes more probable that the overall closest images are included in the result set.

Figures 5 and 6 examine the case for constant k=20 and N_f =2 but N increasing from 5,000 to 75,000. In Figure 6 we

examine the value of r needed to produce 90% and 100% correctness. For guaranteed correctness r must be increased in proportion with the data set size. However, if some inaccuracy is tolerable to the user, significant savings can be obtained by reducing r to the level of 90% correctness.

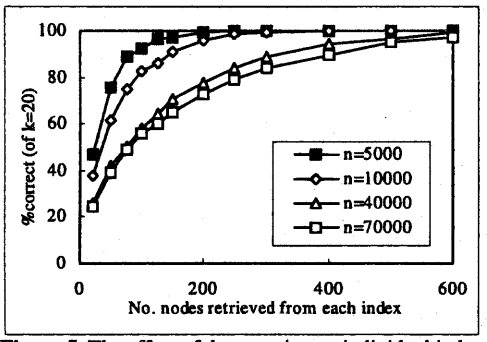

Figure 5. The effect of data set size on individual index retrieval set size and accuracy

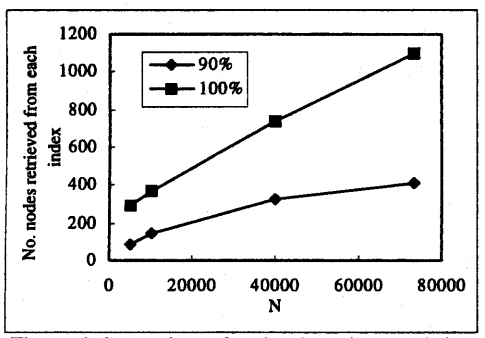

Figure 6. Comparison of retrieval set sizes needed to achieve certain accuracy in overall result

In addition to being responsible for combining search results from different indices, the coordinator is also responsible for combining the results with conventional searches carried out on data attributes in the database.

3.4 General Implementation Issues

HyperMatch is implemented as a server that responds to requests from incoming client programs. Current plans call for a multi-threaded implementation that will allow for multiple users at multiple locations to query the system simultaneously. This also has the important advantage that the client interface can be a small program that can be run successfully with significantly less computational resources than are needed by the *HyperMatch* engine.

An extremely important issue which is not directly addressed in this paper is how a choice can be made as to which indices to use in order to best satisfy a user request. For example, if a user is primarily concerned with finding objects of a similar color, then the search procedure must place emphasis on the results obtained from search the color feature vector space. The position or shape feature vectors spaces can probably be safely ignored. This decision is made prior to the use of the *HyperMatch* server, either directly by the client application or by a special module known as the strategist which may be accessed by the client. These special strategist modules are intended to provide domain specific assistance in achieving good retrieval results.

The external interface presented by *HyperMatch* is designed to allow the connected user interface maximum flexibility in its design without propagating any design changes to the *HyperMatch* retrieval engine. A version of extended SQL was decided upon for two reasons. The object-oriented nature of the language allows the query language to be easily extended in future versions of *HyperMatch* via the use of object functions without rendering previous clients obsolete. Also, a familiar SQL interface is expected to be easy for the implementers of client programs to understand and use.

The nature of the index search method also makes it an ideal candidate for parallel implementation over several processors. Each index can be searched simultaneously and the results combined. Preliminary tests have shown this leads to faster search times although there is significant overhead involved in assigning searches and reassembling results from the various sources. The advantages of parallelism are expected to increase significantly as we hope show how the accuracy of the result can be significantly improved by increasing the number of indices searched.

4 Future Directions

We have identified three principle areas for concentrating on future research with the *HyperMatch* system. These are further development of the retrieval optimizing capabilities of the coordinator, continued development and improvement of high dimensional index structures and a comprehensive evaluation of the system performance.

4.1 Coordinator Development

We have identified three areas in which we will concentrate on the development of the coordinator. These are:

Index Choice

Preliminary research shows that the accuracy of retrieval performance can be greatly improved by the consideration of different types of index or distance metric in different situations. For example, although the user may think that a color match is best found by consulting the red, green, and blue feature vector spaces, experiments have shown that the additional inclusion of the hue vector space can significantly improve retrieval accuracy. It is intended that the client and strategist modules will be primarily responsible for determining index choice. However, the coordinator, being directly connected with the database and index structures is best aware of what index choices exist, i.e. which feature spaces and distance metrics are provided for. Provision should be made for this information to be communicated to the client or strategist on request.

Dynamic Evaluation of Results

If an appropriate distance metric is employed it can be possible to discern a clear cut-off point between images that may be close to the key image and those that are definitely not. Since the user is usually more interested in obtaining matching images rather than a particular number of images, significant performance improvement may be gained by dismissing these non matching candidate images when they are first encountered, even at the possible sacrifice of returning less images than the user has requested. User interaction in the setting of this type of cut-off is anticipated.

If the candidate set of possibly similar images can be narrowed down sufficiently using fewer than the total number of suggested indices then it may become more efficient to evaluate the candidate set exhaustively rather than continuing to use preformatted indices. The coordinator needs to be able to identify these situations and respond accordingly.

Feedback

An important feature for any image retrieval system is the ability to improve the accuracy of a retrieval given some additional information from the end user about the quality of its initial retrieval result. The most common scenario is where the user selects a number of images from the result set as closely matching their requirements and asks the system to return images that are more similar to this new group of images.

In order to be able to fulfill this request, the coordinator must be able to supply information about properties the images have in common, and how these commonalities are expressed using a combination of particular index measurements with appropriate weighting adjustments applied. The search for common features need not be limited to the original feature vectors searched but can encompass the full range of features available for those images.

4.2 Index Performance

The overall speed performance of the system relies on both the speed at which an individual index can be searched and the speed with which individual results can be combined. Therefore it is worthwhile to continue investigating possible new index structures in addition to refining the behavior of the coordinator.

The speed of an index depends on its fundamental type, the number and dimension of the data and the distance metric used. As part of the system evaluation proposed in the next section we emphasize accuracy which is greatly affected by the distance metrics employed. As new distance metrics are proposed it is very important to consider creating new index methods that are optimized for this particular distance measurement. We will also continue to emphasize the need to provide for disk based indices as even with the rapidly increasing machine capacity, it is unlikely that for such large scale image database systems, all the indices required by the system will be able to reside in memory, given the large scale dimensionally of the problem. Reducing the number of indices available in the system is not seen as a valid option as this has a large negative impact on retrieval accuracy.

4.3 Evaluation

One of the most important aspect of large scale image database development has been largely ignored, is the extensive evaluation of the database performance under a variety of conditions and using a variety of performance metrics.

For even the most preliminary evaluation there are two basic categories under which the system should be tested - speed and accuracy. For the first, speed of query retrieval is obviously most important, but this itself is made up of at least two components, speed of query execution and speed of data transmission from storage to user display. The second metric, accuracy, is significantly harder to quantify, since ultimately accuracy is compared to the user's perception of the correct result and so varies considerably from user to user and is extremely dependent on the user's environment and the type of image data under consideration.

Preliminary comparisons between human and computer generated result sets have shown wide discrepancies, but also that being able to combine information from a wide variety of image indices based on a selection of distance metrics can lead to a vast improvement in query performance. It is this advantage that we are trying to exploit with *HyperMatch*.

5 Conclusion

This paper has outlined a proposal for a large image database similarity retrieval engine to be used with a variety of client applications. Thus far, the research has focused on the need to efficiently combine the results from multiple feature space and attribute indices into one coherent result set ranked according to overall similarity. In the future, there remain many more issues concerning optimization, index performance and evaluation to be addressed.

References

[1] Stonebraker M., Object-relational DBMSs, Morgan Kaufmann 1996.

[2] Gupta A. and Jain R., "Visual Information Retrieval," Communications of the ACM pp. 71-79, May 1997.

[3] Wu J.K. et al., "CORE: a content-based retrieval engine for multimedia information systems," Multimedia Systems, Vol. 3, pp. 25-41, 1995.

[4] Petrakis E. and Faloutsos C., "Similarity Searching in Large Image Databases" Technical Report CS-TR-3388, Univ. of Maryland, 1994.

[5] 赤間，三井，紺谷，串間：画像内オブジェクトの自動抽出をった画像検索システムExSight，信学会第8回データ工学ワークショップＤＥＷＳ'９７（1997.3）

[6] Guttman A., "R-trees: a Dynamic Index Structure for Spatial Searching," Proc. ACM SIGMOD, 1984.

[7] White D.A. and Jain R., "Algorithms and Strategies for Similarity Retrieval," Technical Report VCL-96-101, UCSD, 1996.

[8] Roussopoulos N., Kelley S. and Vincent F., "Nearest Neighbor Queries," Proc. ACM SIGMOD, 1995.

[9] カーティス，中川，谷口，山室：Indexing in High Dimensional Image Space, 情処学会マルチメディア通信と分散処理 82-18（1997.4）

[10] 谷口，カーティス，中川，山室，寺中：Multiple Inverted Array Structureを用いた類似画像検索，情処学会マルチメディア・分散・協調とモーバイルワークショップDiCoMo'97（1997.7）