Multimedia Tools and Applications 4, 333–356 (1997)

Benchmarking Multimedia Databases

A. DESAI NARASIMHALU desai@iss.nus.sg
MOHAN S. KANKANHALLI mohan@iss.nus.sg
JIANKANG WU jiankang@iss.nus.sg
Institute of Systems Science, National University of Singapore, Singapore 119597

Abstract. Multimedia technologies are being adopted both in the professional and commercial world with great enthusiasm. This has led to a significant interest in the research and development of multimedia databases. However, none of these efforts have really addressed the issues related to the benchmarking of multimedia databases. We analyze the problem of benchmarking multimedia databases in this paper and suggest a methodology.

Keywords: benchmarking, multimedia database, information retrieval, hypermedia, context-based retrieval

1. Introduction

With the exponential growth of multimedia data comes the challenge of managing this information in a controlled, organized and efficient manner. The database researchers have set their sights on addressing the fresh challenges offered by the management of multimedia data [14]. While there are well known metrics for benchmarking traditional databases either in terms of performance of access methods & query optimization [2] or in terms of transaction throughput power [1], there has been no similar attempt for multimedia databases.

We define the problem in Section 2. Section 3 presents the benchmarking measures for the different retrieval techniques used by a multimedia database system. Section 4 provides a methodology for the overall benchmarking of a multimedia database system. Section 5 provides a brief summary and the future work. Some examples used in developing this methodology are presented in the appendix.

2. Definition of the problem

Multimedia database engines can be complex systems. In this section we will first define the types of retrieval mechanisms available to us and then go on to formulate the definition of the problem of benchmarking multimedia database systems.

2.1. Retrieval as a function of data and query

Multimedia databases can contain attribute data, content data and structure data. Let us discuss this using a multimedia document on *people of different nations*, as an example. The name of a country, its population, area and GNP are examples of attribute data. The

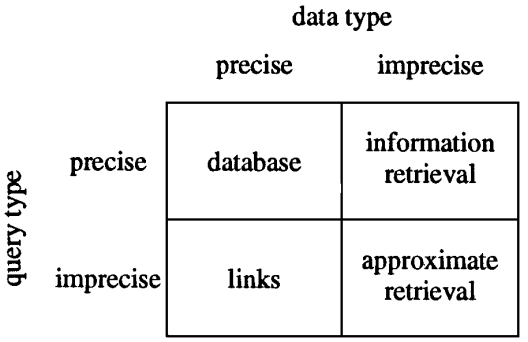


Figure 1. Relationship among data, query and types of retrieval.

types of terrain, the face images of peoples living in that country, the flag of the country and the trademarks of companies in that country are examples of content data.

Content data are normally expressed in terms of features such as shape, color and texture. Certain types of content data such as a video shot of the terrain may be represented using spatio-temporal features. Structures are used to represent the relationship such as those among the country, its states, their counties, their towns and villages. Structures are also used to define navigation among objects or an object and the information about it.

Queries can be classified as well-formulated or *precise queries* and ill-formulated or *imprecise queries*. Queries expressed in SQL and similar tools are examples of well formulated queries. Queries that fall under this category are normally those addressing the attribute data or those using the attribute data to get the content or structure data if such associations exist.

Imprecise queries consist of two types. The first type of imprecise queries arise mostly because of a user's lack of focus and is prevalent in browsing applications using multimedia data. The second type of imprecise queries arise due to either the user's inability to express the content data accurately or user's preference to refer to content data using everyday language. Subjective descriptors such as 'beautiful' and 'thin', which are both abstract and relative abound in everyday language.

We suggest that the retrieval techniques practiced today can be related to the nature of query and data. We present this in figure 1.

All the traditional retrieval techniques used for *on-line transaction processing* (OLTP) applications based on their attribute data have collectively been called *database retrieval technology*. This technology handles precise queries on precise data sets. Thus, in answer to a query on the names of employees earning 10,500 dollars per month would be null (or a zero set) if no such employee exists. The retrieval techniques used for handling structure data of the first type, i.e., representing relationships such as *part-of* and *is-a*, fall under this category. *Information Retrieval* (IR) techniques, on the other hand, have been addressing collections of text document databases [8]. In such databases, the queries are very precise, they are made up of words and the semantics of the words in a query is very clear. Documents are also made of definitive words. However, the different roles played by the same word

in different documents makes the collection of data in the document space less precise. Link based retrieval techniques such as *hypertext* and *hypermedia* [11] have been used to handle the imprecise queries of the first type defined above, to look for precise data. There is as yet no "label" attached to the retrieval techniques that handle imprecise query of the second kind on a collection of data that in itself is imprecise. The popular term in usage is *content-based retrieval*.

However, content based retrieval will also include information retrieval. Content based retrieval covers the right half of the table or the second column in total and not the bottom right quadrant. We like to suggest the phrase *approximate retrieval* (AR), to refer to those retrieval techniques that belong to the bottom right quadrant. We will use the above taxonomy in the following discussions on benchmarking multimedia databases.

2.2. Definition of benchmarking of multimedia databases

A multimedia database application could use any permutation and combination of the above four classes of retrieval techniques and hence our statement at the beginning of Section 2 that multimedia database engines can be complex systems. Clearly, speed of response is an important aspect of benchmarking any database system. This should be assiduously pursued and studied with respect to different levels of scaling up of data collections.

A second component that has been often used as a measure is the expressive power of a query language. Expressive power, in effect, determines the range of queries that an application can use. This is primarily a metric for the query language and not for a retrieval engine. Thus, even when retrieval engines are capable of handling any type of query, a query language may inhibit certain types of queries due to limitations in either the underlying model (such as relational algebra or calculus), or the syntax and semantics of the query language. Hence, we will not consider the range of queries as a measure given that there is now a trend towards using customized 'application programming interfaces' (API) for queries as compared to using structured or other query languages. This assumption can only affect the range of ad hoc queries and not API driven queries.

Multimedia database systems introduce a new concept called the *quality of result* to a query. This is due to the necessity of approximate retrieval for multimedia data. This is especially important in the context of the queries on imprecise data collections and assumes an even more important role when the queries are also imprecise. Quality was never an issue in database engines developed for OLTP applications. They simply returned an answer that satisfied the conditionals in the query or returned a null. Hence, the quality dimension was completely ignored. Given the prevalence of imprecise nature of both data and queries in multimedia databases, quality takes on a greater role than speed given that incorrect response however fast it is, will be of little value.

Hence, the benchmark of a multimedia retrieval engine can be thought of as a quality of service metric:

$$\text{QOS} = f(\text{Speed, Quality of answer}). \tag{1}$$

In the rest of the paper, we will formulate a measure for QOS and suggest a methodology to evaluate this metric.

3. Benchmarkings for different retrieval methods

There are standard metrics for conventional retrieval methods. We will mention them here. But we will certainly devote more discussion on metrics for links and approximate retrieval.

3.1. Benchmarking for database retrieval methods

There are examples of *DebitCredit* [1] and other test standards. These emerged primarily as measures of speed of response for scaled up collections and were typically measured in terms of number of transactions per unit time. The results here will depend on optimization of resources and tools such as buffer management and evaluation of join. The metric for OLTP type of applications T_{DB} is *Transaction per Second*, where transactions are generally speaking, queries. In *DebitCredit*, it is quantified by measuring the elapsed time for two standard batch transactions and throughput for an interactive transaction. A plot of T_{DB} vs. database size will provide the performance of the database engine with respect to database size. Although the database community has considered the *utilization* issue which counts the overhead for generating a useful query result, there has been no benchmark for the quality of retrieval.

3.2. Benchmarking of information retrieval methods

An *Information Retrieval* (IR) system returns a set of relevant objects (usually text documents) for any query. The objects not returned are considered irrelevant. Hence an IR system can also be called as a *Boolean* system since any object can either be relevant or not relevant for a particular query. Assume that we have a database D having n objects (documents):

$$D = \{o_i \mid i = 1, 2, \ldots, n\}$$

where o_i is a database object or a document. For a given query q we have the following:

1. *Ideal system response*: This assumes that an ideal IR system I exists. The response of this ideal IR system is the *ideal* or the desired response. The ideal response can be considered to be two sets which can be considered as a partition on the database D induced by query q:

$$R_q^I = \{o_j \mid (o_j \in D) \wedge (o_j \text{ is relevant})\}$$
$$N_q^I = \{o_k \mid (o_k \in D) \wedge (o_k \text{ is not relevant})\}$$

such that

$$R_q^I \cap N_q^I = \phi$$
$$R_q^I \cup N_q^I = D$$

Note that the ideal response can be considered as the *ground truth*. This can be a corpus which is a collection of known objects with known ideal responses to some known queries.

2. *Test system response*: This is the response actually provided by an IR system E being benchmarked. This system also partitions D for the same query q:

$$R_q^E = \{o_u \,|(o_u \in D) \wedge (o_u \text{ is relevant according to } E)\}$$
$$N_q^E = \{o_v \,|(o_v \in D) \wedge (o_v \text{ is not relevant according to } E)\}$$

such that

$$R_q^E \cap N_q^E = \phi$$
$$R_q^E \cup N_q^E = D$$

Given R_q^I, N_q^I, R_q^E and N_q^E, the IR community has the following two well known measures for benchmarking:

- **Recall:** The *recall* for query q is defined as:

$$\frac{\left\| R_q^E \cap R_q^I \right\|}{\left\| R_q^I \right\|}$$

where $\| \cdot \|$ indicates the cardinality of the set.

- **Precision:** The *precision* for query q is defined as:

$$\frac{\left\| R_q^E \cap R_q^I \right\|}{\left\| R_q^E \right\|}$$

Relevance of retrieved documents with respect to a query measured in terms of precision and recall have continued to be the commonly accepted metrics. There is, however, a growing disillusionment among the IR community that these two measure somehow do not tell the whole story but there is yet to emerge alternatives to these two measures. Speed of retrieval T_{IR} for IR has to be measured differently from that for database technology. This is primarily because the application environment does not have the same volume of transactions and the retrieval process is more interactive due to the imprecise nature of the data collection.

$$T_{\text{IR}} = \frac{\text{No. of query terms}}{\text{Response time}} \tag{2}$$

Quality of output from IR systems can be measured as follows.

$$Q_{\text{IR}} = W_{\text{PR}} * \text{precision} + W_{\text{RE}} * \text{recall} \tag{3}$$

where, W_{PR} is the weightage for precision and W_{RE} is the weightage for recall. Then, the overall quality of service can be defined as:

$$\text{QOS}_{\text{IR}} = T_{\text{IR}} * Q_{\text{IR}} \tag{4}$$

TREC, the annual *Text Retrieval Conference* organized by the *National Institute of Standards and Technology* (NIST) in the United States continues to be the key forum for benchmarking both commercial and academic IR systems [3]. This conference aims to bring IR groups together to discuss their work on a new large test collection. The participants of TREC employ a variety of retrieval techniques, including methods using automatic thesaurii, sophisticated term weighting, natural language techniques, relevance feedback, and advanced pattern matching. These methods are run through a common evaluation package in order for the groups to compare the effectiveness of different techniques and to discuss how differences between the systems affected performance.

Thus, the emphasis of this exercise is to benchmark the different IR methods on increasingly large and common collections. It is important to note that these two metrics relate to the quality of the answer to a query. Although TREC participants may record the time taken to get a result, quality measures have assumed a more important role in this benchmarking exercise.

TREC is now considering corpuses in languages other than English. A complementary forum called the *Message Understanding Conference* (MUC) is likely to produce results that may impact information retrieval methods.

3.3. Benchmarking links

We have not come across any attempts to benchmark significant sized collections of hyperlinks. We list below the measures that could be used for retrieval methods for links.

3.3.1. Speed as a measure. The time, T_{LI}, to fetch immediate (or next level) information associated with a link is definitely an immediate measure to consider. This measure will be affected by the quality of schemes such as buffer management and cluster management on the disk.

In addition, the time taken to reach distant information, T_{LD}, can also be a measure. This measure will be affected by the design of the information structures. If there are too many levels of nesting before reaching a desired information, then this measure will suffer. It will also be poor when retrieval on structures is not complemented by other retrieval methods. The speed of response, T_L, is defined as:

$$T_L = \frac{W_{\text{LI}} * N_{\text{LI}} * T_{\text{LI}} + W_{\text{LD}} * N_{\text{LD}} * T_{\text{LD}}}{W_{\text{LI}} * N_{\text{LI}} + W_{\text{LD}} * N_{\text{LD}}} \tag{5}$$

where W_{LD} is the relative importance of distant transactions, N_{LD} is the number of distant transactions, W_{LI} is the relative importance of immediate transactions and N_{LI} is the number of immediate transactions.

3.3.2. Quality as a measure. The ability to find a piece of information is one measure of quality. This can be represented by the ratio Q_L, between the number of successful retrievals of information to the total number of queries.

$$Q_L = \frac{\text{No. of successful retrievals}}{\text{Total No. of queries}} \tag{6}$$

No retrieval method employing links can afford to keep all possible links all the time. Hence, it is desirable to generate links when they are not predefined. This is a difficult task and is dependent on the extent of supporting information available to a link retrieval method.

3.3.3. A wholistic measure for link based system. A total quality measure for link based system can be defined to be,

$$\text{QOS}_L = T_L * Q_L \tag{7}$$

3.4. Benchmarks for approximate retrieval methods

Benchmarks for the approximate retrieval methods are the least understood or established. Hence we will devote more discussion to this topic. The imprecise representation of data will have to be converted into some objective measures represented in terms of numbers or alphanumeric characters. This transformation may employ concepts such as fuzzy mapping, fuzzy set theory, classification using traditional clustering techniques and more recent methods using neural nets, and classification trees.

3.4.1. Speed as a measure. AR methods will in general use data from more than one dimension for retrieval. The features that identify a country from its flag may use color and the spatial location of the different colors as the two features or dimensions for retrieval. The difference between AR and other retrieval methods, more often, lies in the fact that most features take on continuous values. For example, while there are only a finite number of words in a corpus of text, the number of face images in a face database can be very large and the shape feature like "moment invariants" (say for the nose, eyes, mouth and chin) can assume continuous values. Speed of response will continue to be an important measure. However, AR methods may need to be evaluated with greater emphasis on the quality of the response rather than the speed. The speed, T_{AR}, can be measured by:

$$T_{\text{AR}} = \frac{\text{No. of features in query}}{\text{Response time}} \tag{8}$$

3.4.2. Quality as a measure. Quality of response has been a subject of discussion even as early as 1968 [5]. The measures presented here will apply to the qualitative aspects of an AR system. The ranked list of objects in response to a query (based on some features) is referred to as the *response set*. The objects that are useful and relevant to the query are called *relevant objects*.

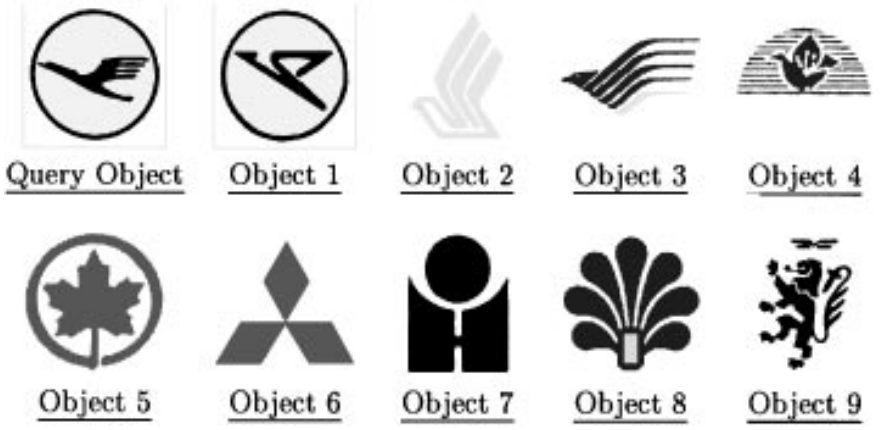


Figure 2. Some trademark image objects used in the example.

To illustrate the importance of quality of result for AR techniques, we will present an example now. This example will also serve as a vehicle to present certain important aspects of the quality of a retrieval result. Assume that we have a database of different trademark shape images. A typical query to this system would be to obtain all images in the database which are *similar in shape* to a query trademark shape image. So it uses an AR technique which utilizes shape similarity computation between two images. It answers any shape-similarity query by computing the similarity between the query trademark shape image and all the trademark shape images in the database and presents them in a ranked list in the decreasing order of similarity to the query image. We also refer to an image (i.e., a trademark shape) as an object. Object i is represented as o_i in the following discussion and in general, it refers to a multimedia object. Assume that the query object o_q is a bird. Let the four objects o_1, o_2, o_3 and o_4 (all of them are stylized versions of birds) be the relevant objects in the database for the query object o_q. Let us also assume that for the four relevant objects, the ith object is expected to appear in ith rank in a response set. So the o_1 is the most similar object to the query bird image, the o_2 is the next-most similar object followed by o_3 and finally o_4 is the least similar relevant object. This also implies that all other objects in the response sets constitute noise and are irrelevant. Figure 2 shows the query object o_q, all the relevant objects (o_1 through o_4) and some of the irrelevant objects (o_5 through o_9) in the database. We will use two response sets for our discussions below. The use of two response sets will help illustrate the concepts discussed better than the use of a single response set.

The following observations on relevant objects in ranked response sets obtained for the same query from two different retrieval engines will be used to discuss the new performance metrics (or measures).

- **Observation 1:** The same set of objects may appear in different orders in the two response sets (Order).

Example 1.

Response set 1: $o_1, o_2, o_3, o_4, \ldots$
Response set 2: $o_2, o_4, o_3, o_1, \ldots$

In this example, the first system correctly retrieves the four relevant objects in descending order of similarity. The second system has retrieved the four relevant objects at the top of the list but they have not been retrieved in a descending order of similarity. Hence the quality of the second system is not good as that of the first system.

- **Observation 2:** The same set of objects while appearing in the same order may have different ranks in the two response sets (Rank).

Example 2.

Response set 1: $o_1, o_2, o_3, o_4, \ldots$
Response set 2: $o_7, o_1, o_2, o_3, o_4, \ldots$

In this example, the first system has performed ideal retrieval. The second system has retrieved an irrelevant shape as the most similar shape and the next four objects are the same as in the ideal response. Hence, even though the second system obtains the correct order, the ranks of the individual objects are not correct. Hence the second system response is of an inferior quality.

- **Observation 3:** The same set of objects while appearing in the same order may have different spreads (Spread).

Example 3.

Response set 1: $o_1, o_7, o_2, o_3, o_4, \ldots$
Response set 2: $o_1, o_2, o_8, o_9, o_5, o_3, o_4, \ldots$

This example shows while the order can be preserved, the number of objects to be considered can vary for all relevant abjects to appear in the response. In this case, both systems have retrieved the relevant objects in a correct order, but the second system response has more irrelevant objects. Hence the spread of the ranked list which encompasses the relevant objects is larger for the second system which is less desirable.

- **Observation 4:** The same set of objects while having the same Spread may have different displacements of individual objects in the two response sets (Displace).

Example 4.

Response set 1: $o_1, o_7, o_2, o_6, o_3, o_4, \ldots$
Response set 2: $o_1, o_7, o_6, o_2, o_3, o_4, \ldots$

Here, the number of objects appearing between the first relevant object and the last relevant object on the ranked list is the same. However, the displacement (from the ideal position) of the individual relevant objects are different. For the first system o_2 is displaced by one position while o_3 and o_4 are displaced by two positions. For the second case o_2, o_3 and o_4 are displaced by two positions each. Hence the response of the second system is of a lesser quality than that of the first system.

Taking these observations in account, we will now classify all types of AR systems into two categories. We will define the measures for a response set and then define the measures for calibrating an AR system. The preliminary version of these ideas were first presented in [7].

3.4.3. Types of AR systems. Depending on the type of the response set produced, we can classify AR systems into two categories:

1. **Continuous:** A continuous AR system returns a ranked list of object as the response to any query. Essentially, for any query q, it computes a relevance measure (often a *similarity measure* based on content of multimedia objects) for all the objects in the database and returns the set of objects in a ranked manner where the rank indicates the relative relevance to the query. The first object in the ranked list is the most relevant object for the query (most *similar* object for similarity retrieval) and the last object in the list is the least relevant object. More formally, assume that we have a database D having n objects:

   $$D = \{o_i \mid i = 1, 2, \ldots, n\}$$

   where o_i is a multimedia object such as text, image, audio or video. For the continuous AR system, we also have a *Relevance Function* $\mathcal{S}$, which for a given query q assigns a relevance measure s_i^q to every every object o_i in the database i.e., it is a function $\mathcal{S} : D \times Q \rightarrow [0, 1]$ such that $\mathcal{S}(o_i, q)$ is the relevance measure of object o_i for query q. Q is the set of all queries. Note that [0, 1] is a closed subset of $\Re$, the set of real numbers. Also note that:

   $$\mathcal{S}(o_i, q) = 0.0 \Rightarrow o_i \text{ is totally irrelevant for query } q$$
   $$\mathcal{S}(o_i, q) = 1.0 \Rightarrow o_i \text{ is totally relevant for query } q$$

   For a given query q we have the following:

   (a) *Ideal system response*: This assumes that an ideal continuous AR system I which has the associated relevance function $\mathcal{S}^I$ exists. The ideal response to a query q can be considered to be a *ranked* list:

   $$L_q^I = \left(o_1^I, o_2^I, o_3^I, \ldots, o_n^I\right)$$

   where L_q^I is a ranked list such that:

   $$\mathcal{S}^I\left(o_1^I, q\right) \geq \mathcal{S}^I\left(o_2^I, q\right) \geq \mathcal{S}^I\left(o_3^I, q\right) \cdots \geq \mathcal{S}^I\left(o_n^I, q\right)$$

   Therefore the first ranked element o_1^I is the most relevant object in the database for query q and the last element o_n^I is the least relevant. The ranking is done using the relevance function $\mathcal{S}^I$. Note that $\mathcal{S}^I$ induces a partial order on the objects of D. Most often, this relevance function is a similarity measure based upon some feature of the objects. It is clear that this is the "ideal" or the desired ranking of the objects in the database.

(b) *Test system response:* This is the response actually provided by an AR system E, with the associated relevance function $\mathcal{S}^E$, being benchmarked. What this system returns is also a ranked list:

$$L_q^E = \left(o_1^E, o_2^E, o_3^E, \ldots, o_n^E\right)$$

which is ranked in the descending order of relevance by the relevance function $\mathcal{S}^E$. Therefore, $\mathcal{S}^E(o_1^E, q) \geq \mathcal{S}^E(o_2^E, q) \geq \mathcal{S}^E(o_3^E, q) \cdots \geq \mathcal{S}^E(o_n^E, q)$.

Precision and recall are not adequate to capture the performance of an AR system since they cannot account for the *order*, *rank*, *spread* and the *displacement* aspects of response sets. We therefore need new measures for benchmarking AR systems performance. In order to come up with new measures, we adopt the following two principles:

(a) *Principle 1*: If an object is displaced from its "ideal rank" for an AR system, then the penalty is symmetric with respect to moving up or down the list. That is to say that the penalty for displacement around the ideal position is the same whether the displacement increases or decreases the rank.
(b) *Principle 2*: The penalty of an object displaced from its "ideal rank" is directly proportional to its relevancy measure. This is intuitive because given equal displacements, the displacement of an object with a higher relevancy measure is less desirable than for an object with a lower relevancy measure. Also, this is reasonable since the relevancy measure is the *basis* for the ranking.

Based on these two principles and the four observations noted in Section 3.4.2, we propose the following two measures:

- **Order:** The *order* of a response set is defined as the size of the longest consecutive subsequence of objects appearing in their ideal order (not the ideal rank). This basically measures one aspect of the quality of the relevance function of the AR system being benchmarked. For a database of size n, the ideal order is n. This measure captures the Order aspect of response sets described in Section 3.4.2.
- **Weighted displacement:** For an object o_i^E, the displacement $d_q(o_i^E)$ for a query q is defined as:

$$d_q\left(o_i^E\right) = \left|\left(\text{Rank}\left(o_i^E\right) \text{ in } L_q^I\right) - \left(\text{Rank}\left(o_i^E\right) \text{ in } L_q^E\right)\right|$$

The displacement indicates the amount an object has moved from its ideal rank. We define the performance measure *weighted displacement*, ω_q^E, for a query q by weighting the displacement with the ideal relevance measure:

$$\omega_q^E = \sum_{i=1}^{n} \mathcal{S}^I\left(o_i^E, q\right) d_q\left(o_i^E\right)$$

Note that ω_q^E is 0 if the ideal response is returned by AR system E. Also, this measure incorporates the two principles described above. Moreover, it captures the Rank (through $\mathcal{S}^I$) and Displace (through $d_q(o_i^E)$) of Section 3.4.2. Note that Spread is not relevant in case of an continuous AR system since the spread will be n for all responses.

2. **Mixed:** A mixed AR system has characteristics of both Boolean IR systems and continuous AR engines. The mixed AR system returns a response set consisting of two subsets—a set of relevant objects and a set of irrelevant objects for any query. Moreover, for the set of relevant objects, it also returns a ranked list of objects based on the relevance function for that query. Again, assume that we have a database D having n multimedia objects:

$$D = \{o_i \mid i = 1, 2, \ldots, n\}.$$

Like in the case of the continuous AR system, we also have a *Relevance Function* $\mathcal{S}$, which for a given query q assigns a relevance measure s_i^q to every every object o_i in the database. It is a function $\mathcal{S} : D \times Q \rightarrow [0, 1]$ such that $\mathcal{S}(o_i, q)$ is the relevance measure of object o_i for query q. Again, Q is the set of all queries. For a given query q we have the following:

(a) *Ideal system response*: This assumes that an ideal AR system I exists. The ideal response can be considered to consist of two sets which is as a partition on the database induced by query q:

$$R_q^I = \{o_j \mid (o_j \in D) \wedge (o_j \text{ is relevant})\}$$
$$N_q^I = \{o_k \mid (o_k \in D) \wedge (o_k \text{ is not relevant})\}$$

such that

$$R_q^I \cap N_q^I = \phi$$
$$R_q^I \cup N_q^I = D$$

In addition, the set of relevant objects, R_q^I, has an associated ranked list L_q^I:

$$L_q^I = \left(o_1^I, o_2^I, o_3^I, \ldots, o_x^I\right) \quad \text{for some } x \leq n,$$

such that:

$$\mathcal{S}^I\left(o_1^I, q\right) \geq \mathcal{S}^I\left(o_2^I, q\right) \geq \mathcal{S}^I\left(o_3^I, q\right) \cdots \geq \mathcal{S}^I\left(o_x^I, q\right)$$

This means that there are x relevant objects for the query q ranked in a descending order of relevance.

(b) *Test system response*: This is the response actually provided by a mixed AR system E being benchmarked. What this system returns is a ranked list:

$$L_q^E = \left(o_1^E, o_2^E, o_3^E, \ldots, o_n^E\right)$$

which is ranked in the descending order of relevance by the relevance function $\mathcal{S}^E$. Therefore, $\mathcal{S}^E(o_1^E, q) \geq \mathcal{S}^E(o_2^E, q) \geq \mathcal{S}^E(o_3^E, q) \cdots \geq \mathcal{S}^E(o_n^E, q)$. Now, as far as the relevant and non-relevant sets go, there are two scenarios for the mixed AR system response:

i. **Size thresholding:** In this case, the relevant number of objects for query q from the response set L_q^E is considered to be ℓ. This means that the top ℓ objects of

the ranked list L_q^E are relevant for the query. Formally, the relevant set of objects R_q^E is:

$$R_q^E = \left\{ o_j^E \mid \left(o_j^E \in L_q^E\right) \wedge \left(j \leq \ell\right) \right\} \quad \text{where } \ell \leq n$$

Note that it is reasonable to have $\ell \leq x$ because the ideal system I returns x relevant objects, but this may not always be the case. The set of irrelevant objects N_q^E is defined as:

$$N_q^E = \left\{ o_k^E \mid \left(o_k^E \in L_q^E\right) \wedge \left(o_k^E \notin R_q^E\right) \right\}$$

For the case of size thresholding, we propose three performance measures which can be used:

- **Order:** The *order* of a response set is defined as the size of the longest consecutive subsequence of *relevant objects* (according to I) appearing in their ideal order (but not necessarily possessing the ideal rank). This basically measures one aspect of the quality of the relevance function of the AR system being benchmarked. The ideal order is ℓ if $\ell < x$ and is x if $\ell \geq x$. This measure captures the Order aspect of response sets described in Section 3.4.2.
- **Rank:** The *rank* of a response set is defined as the sum of the ranks (positions) of the relevant objects appearing in the system response relevant set. If an object appears in the ideal response set R_q^I but does not appear in the system relevant set R_q^E, it is assumed to have a rank of $\ell + 1$. The ideal rank will be $\frac{\ell(\ell+1)}{2}$ if $\ell < x$. If $\ell \geq x$, then the ideal rank is $\frac{x(x+1)}{2}$.
- **Fill ratio:** The *fill ratio* is defined to be the ratio of the number of actually relevant objects appearing in R_q^E to the number of objects in R_q^E which is ℓ. Therefore,

$$\text{Fill ratio} = \begin{cases} \frac{\|\chi\|}{\ell} & \text{if } \ell < x \\ \frac{\|\chi\|}{x} & \text{if } \ell \geq x \end{cases}$$

where

$$\chi = \left\{ o_j \mid o_j \in \left(R_q^I \cap R_q^E\right) \right\}$$

ii. **Object thresholding:** If the last relevant object o_x^I of L_q^I appears at the $\bar{n}$th position of L_q^E, then we threshold L_q^E by $\bar{n}$. So the response set is thresholded such that the last (ideally) relevant object appears in the set. In other words, the relevant number of objects is $\bar{n}$ such that the *last* object o_x^I of the ideal response ranked list L_q^I appears in the $\bar{n}$th position of system E's response. So, formally the relevant set of objects R_q^E is:

$$R_q^E = \left\{ o_j^E \mid \left(o_j^E \in L_q^E\right) \wedge (j \leq \bar{n}) \right\}$$

where

$$\bar{n} = \text{Rank of } \left(o_x^I \in L_q^I\right) \text{ in } L_q^E$$

The set of irrelevant objects N_q^E is defined as:

$$N_q^E = \left\{ o_k^E \mid \left(o_k^E \in L_q^E\right) \wedge \left(o_k^E \notin R_q^E\right) \right\}$$

For the case of object thresholding, we propose three performance measures which can be used:

- **Order:** The *order* of a response set is defined as the size of the longest consecutive subsequence of relevant objects appearing in their ideal order. This again measures one aspect of the quality of the relevance function of the AR system being benchmarked. The ideal order is x. This measure captures the Order aspect of response sets described in Section 3.4.2.
- **Spread:** The *spread* of a response set is the threshold at which all the objects in R_q^I appear in R_q^E. In other words, the spread is exactly equal to $\bar{h}$. This measure captures the Spread aspect described in Section 3.4.2.
- **Weighted displacement:** For an object o_i^E, the displacement $d_q(o_i^E)$ for a query q is defined as:

$$d_q\left(o_i^E\right) = \left|\left(\text{Rank}\left(o_i^E\right) \text{ in } L_q^I\right) - \left(\text{Rank}\left(o_i^E\right) \text{ in } L_q^E\right)\right|$$

The displacement indicates the amount an object has moved from its ideal rank. We define the performance measure *weighted displacement*, ω_q^E, by weighting the displacement with the ideal relevance measure:

$$\omega_q^E = \sum_{i=1}^{n} \mathcal{S}^I\left(o_i^E, q\right) d_q\left(o_i^E\right)$$

Note that ω_q^E is 0 if the ideal response is returned by RLIR engine E. This measure incorporates the two principles described earlier. It also captures the Rank and Displace of Section 3.4.2.

3.4.4. Quality measures for calibrating an AR system. Measures for calibrating an AR system will assume the existence of an ideal AR system that produces the Expected Response Set (ERS). ERS is the ideal answer to a query. Hence ERS will become the reference result against which results from all other AR systems will be evaluated. Such evaluations can then be used to calibrate the AR system under test. We can define relative measures which show how well a system E performs with respect to a particular measure against the ideal system I. We define the relative measures in the range [0, 1]. A relative measure closer to unity indicates a better (more "ideal") response and hence is of better quality. We have defined the new measures in the last section. We will now show how to compute the relative measure for each of the type of AR system:

1. **Continuous AR system:**

- **Order:** Since the ideal order for a database of size n is equal to n (the largest possible), the relative order for a query q can be defined as:

$$\text{Relative order}_q = \frac{\text{Order of engine } E \text{ for query } q}{n}$$

- **Weighted displacement:** Since the ideal weighted displacement is 0, the relative weighted displacement for a query q can be defined as:

$$\text{Relative weighted displacement}_q = \frac{\omega_q^E}{\Omega}$$

where

$$\Omega = \begin{cases} \frac{n^2}{2} & \text{if } n \text{ is even} \\ \frac{n^2-1}{2} & \text{if } n \text{ is odd} \end{cases}$$

for a database of size n. Note that it can be proved that Ω is the largest weighted displacement possible assuming all objects have the highest weight of unity [10].

2. **Size thresholded mixed AR system:**

- **Order:** Since the ideal order for a database of size n is equal to ℓ, the relative order for a query q can be defined as:

$$\text{Relative order}_q = \begin{cases} \frac{\text{order of } E \text{ for query } q}{\ell} & \text{if } \ell < x \\ \frac{\text{order of } E \text{ for query } q}{x} & \text{if } \ell \geq x \end{cases}$$

- **Rank:** This can be computed as the ratio of the ideal rank to the engine rank:

$$\text{Relative rank} = \frac{\text{Rank}\left(R_q^I\right)}{\text{Rank}\left(R_q^E\right)}$$

- **Fill ratio:** Since the fill ratio of the ideal system is 1 and that of the system response is less than or equal to 1, the relative fill ratio is equal to the fill ratio.

3. **Object thresholded mixed AR system:**

- **Order:** Since the ideal order for a database of size n is equal to x, the relative order can be defined as:

$$\text{Relative order}_q = \frac{\text{Order of } E \text{ for query } q}{x}$$

- **Spread:** The relative spread is the ratio of the spread of the ideal response to that of the system response:

$$\text{Relative spread}_q = \frac{\left\| L_q^I \right|}{\bar{h}}$$

- **weighted displacement:** Since the ideal weighted displacement is 0, the relative weighted displacement can be defined as:

$$\text{Relative weighted displacement}_q = \frac{\omega_q^E}{\Omega}$$

where

$$\Omega = \begin{cases} \frac{n^2}{2} & \text{if } n \text{ is even} \\ \frac{n^2-1}{2} & \text{if } n \text{ is odd} \end{cases}$$

for a database of size n.

3.4.5. Overall measure for AR method. The measures defined earlier can be combined to derive an overall figure of merit by using a weighted combination of the individual measures. The quality AR of an AR system can be computed as:

$$Q_{\mathrm{AR}} = \frac{\sum(W_i * M_i)}{\sum W_i} \quad \text{for } i = 1 \ldots N \tag{9}$$

where M_i is the appropriate relative measure i (e.g., for an object-thresholded mixed AR system, it will be relative order, relative spread and relative weighted displacement), W_i is the relative importance of the measure i and N is the total number of measures appropriate for that AR system. Then the overall quality of service can be defined as:

$$\mathrm{QOS}_{\mathrm{AR}} = T_{\mathrm{AR}} * Q_{\mathrm{AR}} \tag{10}$$

3.5. Complete benchmark for multimedia databases

We have described the benchmarks for each of the different retrieval methods in Sections 3.1. to 3.4. Different multimedia database applications may use one or more of the retrieval methods in some permutation. The combination of retrieval methods may be used either independently or in some interdependent manner. The independent combination will use the different retrieval method to get the answers to different parts of a complex query. The results are usually collated into different parts of the response to a query. The overall benchmark for such an application will be,

$$\mathrm{QOS}_{\mathrm{MMDB}} = \frac{W_Q * \prod(Q_i)}{W_T * \max(T_i)} \tag{11}$$

where W_T is the importance of time of response, W_Q is the importance of quality, Q_i is the quality measure of the ith retrieval technique, T_i is the speed measure of the ith retrieval technique. The simplest form of interdependent usage may be to use the results from one retrieval method as the input to another retrieval method. In such a case, the overall benchmark may be represented as follows.

$$\mathrm{QOS}_{\mathrm{MMDB}} = \frac{W_Q * \prod(Q_i)}{W_T * \sum(T_i)} \tag{12}$$

The graph in figure 3 shows the desired region into which we would like to see the systems fall.

4. A methodology for benchmarking of multimedia databases

The benchmarking of multimedia databases use the time and quality measures that were discussed in Section 3 on what we call a multimedia database engine. Such an engine may be just one of the four types of retrieval methods or some permutation and combination of the four types.

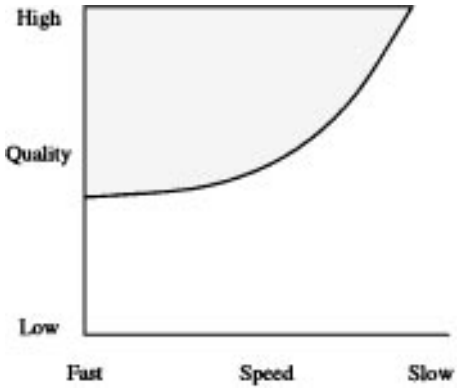


Figure 3. Desired region in quality-speed plane.

4.1. Scalability with respect to time

This is a fairly straightforward method. One can start with certain number of records in the database and then scale up using multiples of same quantity. Then a plot of time (or speed) of response versus the database size for the same query will give us a good idea of scalability of the engine with respect to time.

4.2. Scalability with respect to quality

The measures for quality for each of the retrieval methods has been presented in Section 3. We will discuss how these can be used for the different types of retrieval methods.

4.2.1. Database technology. Since quality has not been a factor for consideration in database technologies used in OLTP applications, we do not suggest any methodology.

4.2.2. Information retrieval. TREC has been attempting to set up testing standards and hence we will keep our discussion on this topic to a minimum. The following quality related tests may be desirable in addition to what TREC has been carrying out. It would be desirable to construct corpuses containing different roles of words capable of playing multiple roles, and to set up the queries corresponding to each of these roles. The different parts of speech will be the set of roles that words can assume.

The corpuses, roles and queries will be determined by the type of applications. For example, applications using corpuses containing name databases would be interested only in those words that play the role of noun phrase. Applications using corpuses containing sports information would be interested in noun phrases and verbs.

The method for such testing can be designed using an iterative two step process. We will first define the two step process.

The first step will be to choose a word that plays the most types of roles and to generate documents and queries corresponding to each of these roles. This will give the different IR algorithms and methodologies an opportunity to refine themselves against this test data.

The second step would be to choose additional words that play the same set of roles and corresponding documents and queries to test the IR methods against this "scaled up corpus."

Later iterations of this two step process should increasingly bring in other words, documents and queries that correspond to other types of roles as dictated by the type of application. The testing can be said to be completed when sufficient corpus and queries have been generated and tested corresponding to all the common (and necessary) roles.

4.2.3. Links. Benchmarking the links for quality can be set up as a test set of information to be retrieved against queries on links and then computing the quality measure mentioned in Section 3.3.

4.2.4. Approximate retrieval methods. The benchmarking of quality of retrieval from AR methods can be set up in two steps: evaluation of individual features, and overall testing. Evaluation of approximate retrieval is application-type dependent. It must be done against a given set of objects—the *test data set*. The sample data set specifies a sequence of objects which should be retrieved for every given sample. To define a test data set for approximate retrieval, let us define the ideal retrieval. It must be clearly understood that for any AR query, there is a "context" associated with it. This context can vary from person to person and perhaps it can vary with time for the same person. The response to a query can then vary depending on the context. So for AR, the 'right answer' to a query can vary depending on the context. This makes the task of evaluating an AR system very difficult. Therefore, in order to benchmark various AR systems, one needs to fix a context for every query and then obtain a corresponding *ideal retrieval result*. For any given sample, the ideal retrieval result should exactly follow designer's expectation of similarity—all similar data items must be retrieved and be ordered from the most similar to the least similar. In order to perform a fair evaluation, it is necessary to have a large set of queries, contexts and the associated ideal retrieval results. These are in turn driven by the specific application area which the AR system purports to address. Moreover, for ideal approximate retrieval, the test data set should consist test data (and associated queries) for both individual features and the overall retrieval. The system should be tuned using individual features and then tested for overall retrieval. The test data should be used to evaluate the measures presented in Section 3.5. We give a brief description of the use of this methodology for real applications in the appendix.

4.2.5. Combined retrieval techniques. The individual retrieval techniques should be first tested to obtain their individual performance measures. The results can be used to tune the different retrieval techniques. Once the individual retrieval techniques are tested, then they will remain unaltered unless the range of values of the individual features change. The next step is to change the weights for combining the results obtained from each of the component retrieval techniques. The combined result is the overall benchmark and can be used to modify the weights until the desired performance levels are obtained.

5. Future work

We have provided a taxonomy of retrieval methods in this paper and defined the measures for benchmarking each of the methods and combinations of these methods. We have also outlined a methodology for benchmarking multimedia database applications.

Our further research is on refining the quantitative measures and defining the test environments for benchmarks. We would like to propose that some international body work towards MREC (Multimedia Retrieval and Evaluation Conference) along the lines of TREC, which would essentially try to establish benchmarking data and standards for multimedia databases. A good starting point may be the MIRA effort being undertaken in the EC which works on evaluation frameworks for interactive multimedia information retrieval applications [4]. We strongly feel that the multimedia database community should have standard test data sets, standard queries and the *ideal* retrieval output for each of these queries. Perhaps one database will not be enough, several would be needed for the different types/features of multimedia data and also for overall combined retrieval in various applications. In the long run, standard data sets, queries and known ground truth will be the only way to judge the accuracy and utility of different retrieval engines. Moreover, common collections for each type of multimedia data type should be built up. Reporting research results based on massive data sets should also be made mandatory. Fortunately this is already happening in certain niche areas like the NIST standard databases for human face images and handwriting images.

Acknowledgments

We thank our colleagues at ISS for critiquing the ideas presented. In particular, we thank Ang Yew Hock, Lam Chian Prong, Ngair Teow Hin and Limsoon Wong for their help. Mohan S. Kankanhalli and Jiankang Wu's work has been supported by the *Real World Computing Partnership* of Japan.

Appendix: Applications

The benchmarking procedure defined in Section 4 using the measures defined in Section 3 are being employed for testing applications built around multimedia database engines developed at the Institute of Systems Science.

FACE application

The FACE application was reported in [12]. It used image features and text descriptions as the two dimensions. The image features were defined both as crisp sets and fuzzy sets on six facial features: chin, eyes, eyebrow, nose, mouth, and hair. A set of artificially composed facial images are created to form a test data set. In the test data set, there are faces with one, two, or more different facial features. The test is conducted in two steps: Firstly, quality of retrieval is tested against individual facial features. For example, to test retrieval on eyes, a subset of test image data set with different eyes are used. Samples for the test is arbitrarily chosen from the set. Retrieval results are evaluated subjectively by several (10 in our test) people.

The second step is the overall test. It is conducted against all facial features. Figure 4 gives an example of a set of faces with change in one or more features. On the bottom-right side, there is a "similarity retrieval" panel to control the weightage among facial features.

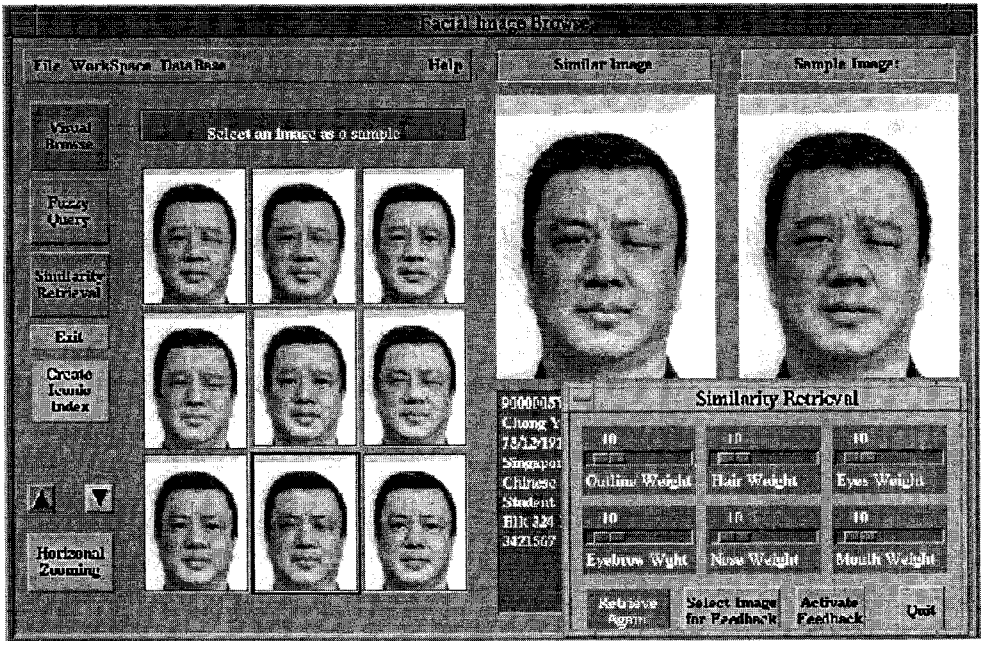


Figure 4.

Here, all weightage are equally set to 10. The nine images on the left are retrieval result with the first being the sample. Comparing with the sample, the differences of other eight images are listed in the table, where "small" "medium" stand for "small difference" and "medium difference":

	Eyes	Eyebrow	Nose	Mouth	Chin	Hair
1	Same	Same	Same	Small	Same	Same
2	Medium	Same	Same	Small	Same	Same
3	Same	Same	Small	Small	Same	Same
4	Medium	Same	Small	Same	Same	Same
5	Same	Medium	Same	Small	Same	Same
6	Same	Medium	Small	Small	Same	Same
7	Medium	Medium	Same	Same	Same	Same
8	Medium	Medium	Small	Same	Same	Same

Such test sets were used for refining the image database engine used in the system. The fuzzy retrieval engine also used similar tests except that the membership of a feature was not limited to one set only. These engines are of the *AR type*. The text retrieval engine used in the system was tested along the quality measures reported in Section 3.2. The text retrieval engine uses the *IR method*. The iconic browsing uses the *link method*.

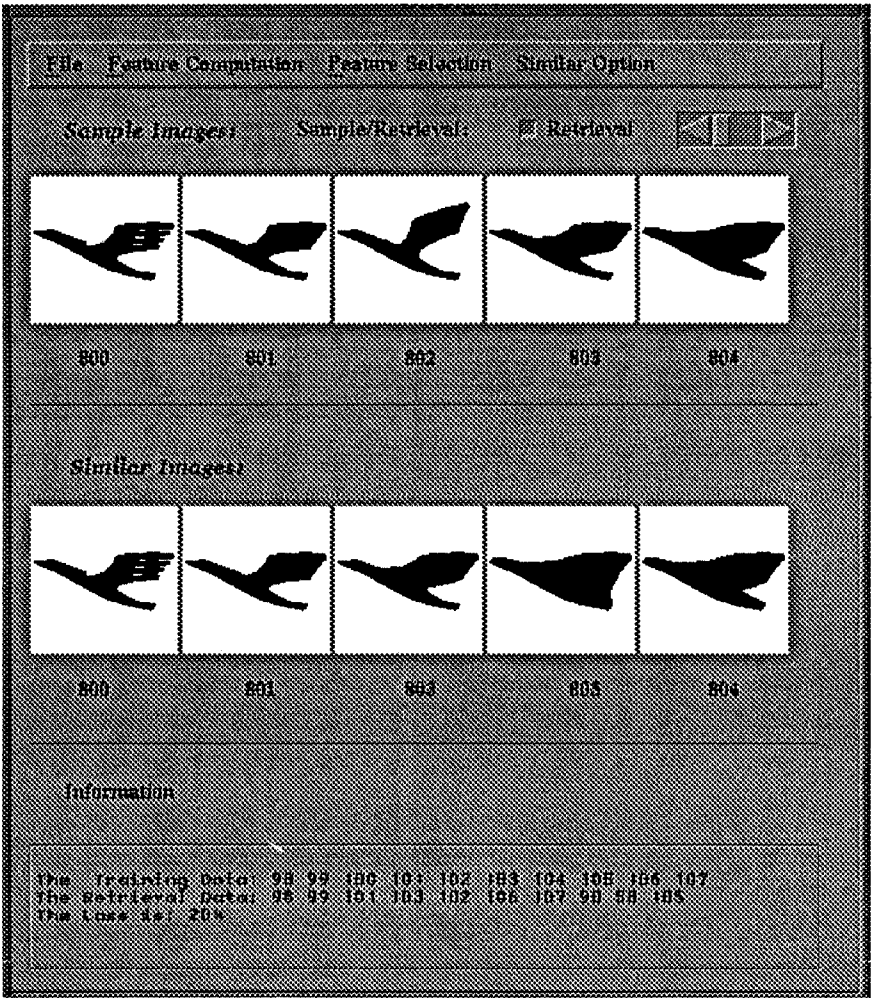


Figure 5.

Trademark application

The trademark application consisted of device mark (local and global shape measures) retrieval, phonetic retrieval, word-in mark retrieval and retrieval by meaning. The retrieval engines used in this application are reported in [13, 14]. While we have different engines for using color as a feature [6], we have not yet integrated this into the present trademark application. The device mark uses the *AR method*. The phonetics retrieval, word-in mark retrieval and the retrieval by meaning are all based on the *IR method*.

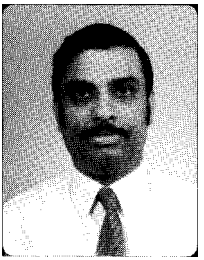
To test the quality of retrieval for device mark shape, three levels of test data sets have been created: mathematically created geometric patterns, actual device marks and their

variations, complicated device marks. In the first class of test patterns, the variations can be exactly controlled by the program used to create these patterns. For example, one series of geometric patterns consists of a circle, an octagon, a hexagon, a pentagon, a square, and a triangle. Figure 5 shows the test results using the second class of test patterns. The images shown in the first row belong to one test series with the first as the sample for retrieval. Neighboring images are more similar than distant ones. The second row shows retrieval result. By comparing these two rows, different quality measures are computed.

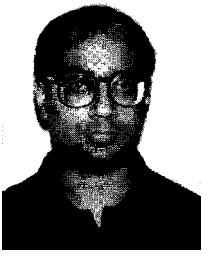
We can see, from figure 5, that in the test data set displayed in the first row, the bird in the third image is indeed more similar to the bird in the sample image (the first one) than that in the fourth and fifth. This test data set is selected from the view point of perception and by consideration of user requirements. The system uses shape similarity for retrieval. The bird in the third test image has its wings open wider, so its shape is surely different from the sample. The quantitative measures of *order* and *weighted displacement* provide a tool for the fine tuning of the system. The third class of test data sets consists of actual trademarks selected by trademark officers. This test data set is for the final test stage to see if the system can bring out potentially conflicting trademarks from the databases with respect to all possible aspects.

References

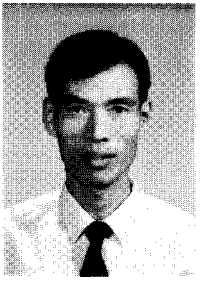
1. Anon et. al., "A measure of transaction processing power," Readings in Database Systems, M. Stonebraker (Ed.), Morgan Kaufmann Publishers, Inc.: San Mateo, CA, pp. 300–312, 1988.
2. D. Bitton and C. Turbyfill, "A Retrospective on the wisconsin benchmark," Readings in Database Systems, M. Stonebraker (Ed.), Morgan Kaufmann Publishers, Inc.: San Mateo, CA, pp. 280–299, 1988.
3. http://potomac.ncsl.nist.gov:80/TREC/
4. http://www.dcs.gla.ac.uk/mira/
5. F.W. Lancaster, Information Retrieval Systems, John Wiley, 1968.
6. B.M. Mehtre, M.S. Kankanhalli, A.D. Narasimhalu, and C.M. Guo, "Color matching for image retrieval," Pattern Recognition Letters, Vol. 16, pp. 325–331, 1995.
7. A.D. Narasimhalu, "Beyond recall and precision: Additional measures for multimedia applications and services," MIRO Workshop, Glasgow, Sept. 1995.
8. G. Salton and M.J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill Advanced Computer Science Series, Auckland, 1983.
9. G. Salton, The State of Retrieval System Evaluation, Information Processing and Management, Vol. 28, No. 4, pp. 441–449, 1992.
10. L. Wong, Personal Communication, 1995.
11. P. Wright, "Cognitive overheads and prostheses: Some issues in evaluating hypertexts," Proceedings of Third ACM Conference on Hypertext, San Antonio, Texas, Dec. 1991, pp. 1–12.
12. J.K. Wu, Y.H. Ang, C.P. Lam, H.H. Loh, and A.D. Narasimhalu, "Inference and retrieval of facial images," ACM Multimedia Journal, Vol. 2, No. 1, pp. 1–14, 1994.
13. J.K. Wu, B.M. Mehtre, Y.J. Gao, P.C. Lam, and A.D. Narasimhalu, "STAR—A multimedia database system for trademark registration," Applications of Databases, Lecture Notes in Computer Science, Vol. 819, pp. 109–122, W. Litwin and T. Risch (Eds.), Proc. First International Conference on Applications of Databases—ADB 94', Springer-Verlag, 1994.
14. J.K. Wu, A.D. Narasimhalu, B.M. Mehtre, C.P. Lam, and Y.J. Gao, "CORE: A content-based retrieval engine for multimedia databases," ACM Multimedia Systems, Vol. 3, pp. 3–25, 1995.



A. Desai Narasimhalu is an associate director of the Institute of Systems Science (ISS), National University of Singapore. He is responsible for the strategic planning and partnership for the research division of ISS. He also manages the *New Initiatives* program which is presently focussing on several research areas—Content-based Retrieval in Multimedia Databases, Information Security, Bio-Informatics, Healthcare, and Map Understanding. He is on the editorial board of several international journals and is a main editor for the ACM/Springer journal on Multimedia Systems. Desai is also a visiting professor with the Department of Computing Science, University of Glasgow.



Mohan S. Kankanhalli received the B.Tech. degree in Electrical Engineering from the Indian Institute of Technology, Kharagpur in 1986. He obtained the M.S. and Ph.D. degrees, both in Computer & Systems Engineering, in 1988 & 1990 from the Rensselaer Polytechnic Institute. Since then, he has been a researcher with ISS, initially working on Medical Imaging. He is now with the Multimedia group, working on Content-based Multimedia Retrieval and Spatial Information Systems. His current research interests include content-based retrieval, image processing, pattern recognition, computer graphics and information security.



Jian-Kang Wu received his bachelor's degree from the University of Science and Technology of China, and his Ph.D. degree from the University of Tokyo. Prior to joining ISS, he worked in the University of Science

and Technology of China as a full professor. He also worked as an Alexander-Humboldt research fellow in the University of Erlangen-Nunberg in 1989, a visiting associate professor in the University of Pittsburgh in 1986, a foreign research fellow in the University of Tokyo in 1982, and as an academic visitor in Imperial College of Science and Technology, London, in 1980. He is now heading a group doing research on active multimedia and spatial information. His research interests include: multimedia and spatial data modeling; flexible storage, retrieval and intelligent utilization of multimedia and spatial data; computer vision and neural networks. He has authored 4 books and more than 50 journal papers.