FISH: A Practical System for Fast Interactive Image Search in Huge Databases

Pradhee Tandon, Piyush Nigam, Vikram Pudi, C. V. Jawahar
International Institute of Information Technology, Hyderabad, India
{ pradhee@research., piyushnigam@students., vikram@, jawahar@ } iiit.ac.in

ABSTRACT

The problem of search and retrieval of images using relevance feedback has attracted tremendous attention in recent years from the research community. A real-world-deployable interactive image retrieval system must (1) be accurate, (2) require minimal user-interaction, (3) be efficient, (4) be scalable to large collections (millions) of images, and (5) support multi-user sessions. For good accuracy, we need effective methods for learning the relevance of image features based on user feedback, both within a user-session and across sessions. Efficiency and scalability require a good index structure for retrieving results. The index structure must allow for the relevance of image features to continually change with fresh queries and user-feedback. The state-of-the-art methods available today each address only a subset of these issues. In this paper, we build a complete system FISH – Fast Image Search in Huge databases. In FISH, we integrate selected techniques available in the literature, while adding a few of our own. We perform extensive experiments on real datasets to demonstrate the accuracy, efficiency and scalability of FISH. Our results show that the system can easily scale to millions of images while maintaining *interactive* response time.

General Terms

Algorithms, Design, Experimentation, Human Factors, Performance, Verification

Keywords

Content Extraction, Dynamic Indexing, Image Retrieval, Long Term Learning, Relevance Feedback, Scalability

1. INTRODUCTION

Since the last few decades, huge amounts of multimedia data are being generated and stored digitally. The reasons for this are the widespread use of good multimedia capture devices and the availability of virtually infinite storage capacity. Even personal collections have become manually unmanageable. Content-aware automatic multimedia data management systems are needed to address this problem. One approach is to annotate multimedia data with textual tags representing the semantics of the data. However, the sheer volume makes annotation impractical for most scenarios. Subjectivity of interpretation also renders this approach inadequate.

This necessitates the use of machine-centric data features which can be automatically extracted, instead of tags. These can then be used for indexing, search, retrieval and comparison of multimedia data. All these features are not equally relevant for all the data objects. Depending on the semantics captured in the data, different features must be given different *weights*.

In interactive multimedia retrieval systems, a typical session starts with the user presenting the system with a query object (say an image or a tune) and the system retrieves the k most *similar* objects from a database. Similarity is computed based on the features and weights stored for each object. The user is then given an opportunity to provide *feedback* regarding whether each retrieved object is indeed similar or not. This feedback is used to modify the feature weights appropriately and the new weights are then used to retrieve fresh results for the similarity query.

Thus, multimedia object retrieval sessions consist of iterative feedback loops, where the similarity measure and the retrieved results are refined continuously. The refinement of feature weights within a user-session (known as *short term learning*) has been extensively studied in the literature [28].

However, a *practical* multimedia object retrieval system has various requirements that are not well-supported by the current state-of-the-art systems. Although the current systems incorporate novel and elegant ideas, most of them are built mainly as a proof-of-concept, and not to be deployed on a large scale. Practical systems have the following broad requirements:

1. **Long-term learning:** Practical systems must support multi-user sessions and the possibility of learning across sessions (known as *long-term learning*). Few studies exist in this domain and most primarily rely on tedious information processing of feedback logs [8, 14]. Such approaches do not scale up well with the dataset size in terms of efficiency as well as accuracy.
2. **Dynamic data collections:** Multimedia data collections are almost always dynamic in nature. Hence practical systems must seamlessly handle new objects inserted into the collection.

CIVR'08, July 7–9, 2008, Niagara Falls, Ontario, Canada.

3. **Dynamic indexing:** A good index structure is required to efficiently retrieve multimedia objects that are similar to a query object [2, 7, 20, 25]. While many schemes exist in the literature, most of them are designed to work with a *fixed* similarity metric. These do not suit our environment since the similarity metric depends on the weights of the features, which continually change with fresh queries and user-feedbacks.

4. **Scalability:** Typical multimedia collections are not small by any standards. Web-based systems for search and retrieval are huge – spanning to *millions* of objects [27]. In contrast, most of the existing techniques have been demonstrated only for much smaller collections involving thousands of objects.

5. **Interactive response:** Practical systems must be optimized for efficiency to the point where multimedia retrieval sessions become interactive for a user. The response time must be suitable for web-based search engines, where it would be frustrating for users to wait for more than a few seconds at most.

6. **Modular extensible design:** In addition to incorporating the above requirements, the resulting system must be easily extensible to new unforeseen requirements. This leaves us in favour of systems that are *simple* to understand, design, implement and modify. In contrast, most existing techniques, while being mathematically robust, are complex and hard to "break into pieces" – a quality essential for extensibility.

In this paper, we build a complete system, **FISH** – **F**ast **I**mage **S**earch in **H**uge databases. We focus on image collections, although the techniques we discuss are also applicable to general multimedia data collections. We perform an extensive experimental study to validate these claims on both real and synthetic datasets. Our results show that the developed system *easily scales to millions of images while maintaining interactive response time.*

The remainder of the paper is organized as follows: In Section 2, we review the state-of-the-art image retrieval methods. We then present FISH, our proposed system and describe its implementation aspects in Section 3. Our chosen indexing scheme in described in Section 4. In Section 5, we discuss the feedback-based learning mechanism (both short and long term) in FISH. Section 6 presents a series of experiments and discusses their results validating our claims of efficiency and accuracy. Finally, in Section 7, we conclude with a few comments on future directions.

2. RELATED WORK

Real-world image retrieval systems must effectively address *all* of the criteria enumerated in Section 1, including scalability, interactive response time, etc. Most of the listed criteria have, in isolation, been addressed in the past. However, the combination of all the listed criteria has so far not been achieved in a single system. In this section we briefly review the research efforts closest to our method in various aspects of the system pipeline including representation, learning, indexing, and user-interface.

Representation of visual content of the images has received considerable attention throughout the history of image retrieval [15, 16, 17]. Methods have explored features ranging from simple global color histograms to color layouts and structures, from point based shape matching to domain centric structural information and from moment based textures to wavelet transforms, all the way to robust highly descriptive point-based features such as SIFT. The choice of representation is effected by multiple factors like the special characteristics of the domain of deployment [1], sensitivity to computation overheads (like SIFT), seamless merger of learning, etc.

The retrieval of images in response to a query should match the user's intent. The machine centric representations, though efficient for extraction, comparison and indexing, are no match for human perception. This requires the systems to incorporate methods to absorb and use semantic input from some external guide, optimally the human user, and improve retrieval. This human input is primarily acquired in the form of his feedback on the relevance and irrelevance of the images in the set returned to him in response to his query. Various aspects of this human input have been extensively analyzed in literature [28]. They can be broadly split into two types. One comprises of direct feature importance modification based methods which either try to modify the similarity metric by biasing it towards the more relevant features or else modify the representation itself to tune the retrieval [10]. The other category prefers to improve the feature based classification of images [19]. These approaches primarily improve the performance for the present query only, better known as Intra-query or Short Term Learning approaches.

Though most of the approaches discard the expensive and invaluable feedback from the user after every query session, some researchers have explored learning from one query to benefit the subsequent ones. This category of techniques is known as Inter-query or Long Term Learning methods. Most of these approaches rely immensely on information processing of logs of user feedback for a considerable history of use. Some try to use these behavior logs for estimating the importance of features and try to predict those for subsequent queries [19]. Others use techniques like SVD, LSI, Neural Networks etc. [8, 14]. Some of these transform to a space where the data elements are related conceptually rather than on features and perform retrieval there. Still others improve using the logs for improving the classification of samples into similarly relevant clusters [26].

The over dependence of some of these approaches on huge amounts of user logs and while the iterative computational expense of others, overshadows their effectiveness as real time online systems and reduces the usability. Most of them also generally scale up poorly with the volume of data and users, making them further inapt for the real world.

Scalability to huge datasets is a key in real world systems. The problem of retrieving the exact k nearest neighbors from a large dataset is prohibitively time consuming. Thus there is a need to best approximate the actual k neighbors. A large body of work explores index structures for supporting similarity search in large datasets. These involve approaches like k-d trees [2], R-tree [7], SS-trees [25] and their variations. An important aspect to keep in mind when choosing an indexing scheme is that most of the popular learning driven retrieval approaches effect the comparison metric in some way. This necessitates that the indexing scheme should be adaptive to changing similarity metrics while the above solutions assume a fixed metric at the time of indexing. They tune the indexing according to the metric to approximate the

k nearest neighbors and are thus unsuitable in our scheme of things. There has been some recent work in the area of efficient search with changing metrics [5, 13]. These approaches primarily use a branch and bound methodology for their purpose, which can degrade to exhaustive retrieval over the entire dataset. Some recent proposals explore the use of B+ trees [11, 27] for efficient indexing. These schemes though effective fail to take into account the inherent nature of the data to form clusters based on a few out of the entire set of features. The data is generally clustered tightly over this small subset of dominant dimensions. This pattern of relative dominance corresponds to a concept in the image, so there can be many of them in an image. This multiplicity can be handled with some modifications to the scheme proposed in this work.

User interaction with the system being the key element in terms of his feedback on the retrieved images, the usability of the interface becomes a very important consideration from the perspective of developing a complete solution. The interaction stretches from the querying mechanism through the display of retrieved results to the method of feedback collection. For keeping the querying interface simple, Query by Example (QBE) and sketching methods should be preferred over methods requiring low level specification in terms of features and their weights [3, 21, 24]. Feedback collection should also be simplified to minimalistic user efforts. The ease of feedback improves learning by increasing the number of feedback iterations the user may provide.

In this paper, we propose some new techniques and adapt some proposed in literature seamlessly into our proposed FISH framework. Our proposal uses a set of inexpensive yet expressive visual descriptors with a highly adaptive indexing scheme which supports the basic notions of data organization into similarity clusters. We also adopt a set of popular relevance feedback approaches from literature. We propose a highly suited scheme for inexpensive inter-query learning. We validate all our claims on a huge real world dataset using a complete system FISH, developed keeping in mind the ease and effectiveness of the user interface to our system.

3. THE FISH SYSTEM OVERVIEW

In this section we describe FISH, our system for Fast Image Search in Huge databases. This system effectively addresses the major research issues plaguing content-based image retrieval systems like accuracy and efficiency and their trade-off on huge real-life data collections.

In Section 3.1, we describe the overall architecture of the system, followed in later subsections by details of the modules. We describe the our representation in Section 3.2, query input and retrieval in Section 3.3 followed by feedback interface and it's absorption in Section 3.4.

3.1 Architecture

The overall architecture of the FISH system is shown in Figure 1. Through the user-interface, which is a web-front end, users (shown on the left in Figure 1) provide query images to the system.

The system processes each query image into an internal representation and searches for similar images in a large database. The search for similar images is made faster by the use of an appropriate index structure. The retrieved similar images are shown back to the user.

The user then has a chance to give feedback to the system

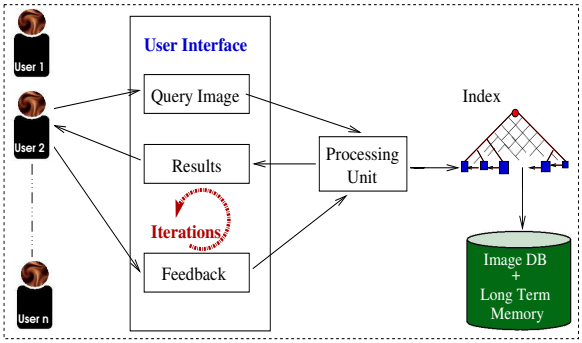

Figure 1: FISH System Architecture

as to whether the retrieved images are indeed similar to the query image. The system uses this feedback to provide better results in the next iteration. These iterations are continued on user's interest.

The ability of the system to provide better results by learning the intent of the user within a session is referred to as short-term learning. In the FISH system, this learned knowledge is represented succinctly and stored in the "long-term memory" for providing better results in later queries.

3.2 Features used to Represent Images

Each image in the system is represented as a vector of numeric feature values $X_1, X_2, \ldots, X_D$. The space of possible vectors constitutes a multi-dimensional space in which each image is a point.

The general features used in any image retrieval system are color, texture and shape descriptors. Color descriptors though weak in description, allow flexibility in use through variations ranging from the global histogram to the color layout descriptors. Texture is generally highly dependent on the homogeneity and regularity of the patterns in pixels. Shape is difficult to extract and represent.

In FISH we have predominantly used color descriptors for our features. We have experimentally selected a weighted combination of third order color moments and some selected MPEG-7 descriptors [17]. We have chosen to use mean, variance and skew color moments which capture the orders of variations of colors in the image. To incorporate texture information we have included three components from the *Texture Browsing Descriptor(TBD)* in MPEG-7 Standard. We have also included the *Color Layout Descriptor* and the *Color Structure Descriptor* from MPEG-7.

The *Color Layout Descriptor(CLD)* [12] captures the layout of the colors in the image by incorporating a DCT transform based representation. The color layout descriptor operates in the YCbCr space by breaking up the image into 8×8 or 64 blocks. Then it computes the dominant color for each of these blocks. To gain computational efficiency, we use the average color for this. This forms a 8×8 pixel representative image. Next, we compute the DCT transform of this representative image and compute the 8×8 DCT coefficients for each of the three channels separately. We then perform a quantization of these three matrices one for every channel and select a few of most representative coefficients from the three channels in a zig-zag scan order to make our feature vector. Based on MPEG-7 standards we use the top 6 coefficients from the Y channel and 3 from the other two.

We use a weighted combination of these coefficients based on scan order.

The *Color Structure Descriptor(CSD)* [18] is a 32-bin quantized representation. It slides an 8×8 window over the entire image and computes an occurrence histogram for the colors in the image. This captures the spatial layout of the colors in the image. The result is a spatially augmented color histogram of the image.

We use a combination these descriptors for our experiments. The dataset of images is indexed on these features. Feature extraction and indexing on dataset images is done offline while user-queries are processed online in real-time.

3.3 Obtaining and Processing the Query

The front end is a web-interface designed with a strong emphasis on ease of use. The design minimizes the efforts the user has to make while interacting with the system. The interface includes all the user-friendly features present in most commercial image search engines. We have minimized the amount of customization parameters the user has to set to use the system.

The user may provide the input query image either by selecting one of the displayed images or by providing a URL to an image. We have strictly avoided low-level methods of querying present in similar systems described in the literature, like the user specifying features and weights, etc. [3].

Once the query image is given, it is processed to extract features (described in Section 3.2) to allow comparison with images in the database. The system then searches the indexed database of images and returns a set of similar images, ranked by similarity. We use a weighted Mahalanobis metric to measure the (dis)similarity between data points. We estimate the covariance matrix using the initial set of database images. It serves as a good approximation of the actual, even in the presence of dynamic dataset modifications.

The image results are displayed through the web interface to the user in the order of decreasing similarity. The results are displayed in sets of images over multiple pages of results. Each of these displayed images has the provision of being queried for if the user wishes so.

3.4 Obtaining and Processing User Feedback

After the result set of similar images is shown to the user, the web-interface is used to retrieve relevance feedback from the user. The user is only required to click the images out of those displayed on the current page to mark them relevant. There are no complications such as the user having to rank the results based on similarity. This makes it non-cumbersome for a lay user to interact with the system.

Once the feedback is given, the system absorbs this feedback by learning, the mechanism of which is discussed in more detail in Section 5. Briefly, the system modifies the weights of features and returns a re-searched set of ranked images based on the semantics estimated from the current feedback iteration. Thus the feedback iteratively tunes the retrieval to match the semantic intent of the user. The learned weights are also used to update stored parameters to obtain more accurate results for future queries.

4. INDEXING

In this section we discuss the use of index structures to quicken the retrieval of similar images. In Section 4.1 we discuss the selection of an index structure for the FISH system. Next, in Section 4.2 we briefly describe its details and then improve upon it in Section 4.3.

4.1 Index Structure Selection

Image retrieval systems need to retrieve images from a dataset one by one and compare them with the query image for similarity. In principle this process can be quickened with the help of a nearest-neighbor index structure that can retrieve images in the neighborhood of a query image.

However, most existing indexing schemes require a fixed similarity metric with which the index is built [2, 7, 25]. Such schemes do not suit our environment since the similarity metric depends on the weights of data features, which continually change with fresh queries and user-feedbacks.

A few indexing schemes are available for changing similarity metrics [5, 13]. However, most of these enumerate a large number of candidate images from the dataset before determining the most similar ones. The reason for this is that they treat all features (dimensions) uniformly. Real-life datasets have inherent clusters in them which result in a highly skewed bias towards specific features.

The lack of a suitable indexing scheme is in fact the major problem faced by most of the current image retrieval methods, due to which, they have slow response times.

In FISH, we use an indexing scheme proposed earlier by us in [11]. It takes advantage of the inherent characteristic of multimedia data to form clusters, shown as a highly skewed bias towards specific features. While other changing metric schemes fail to accommodate this skewed variance the indexing scheme in [11] flourishes in the environment.

4.2 Details of the Index Structure

The index structure in [11] utilizes a B+ tree for each feature/dimension to store the corresponding feature values. Assume a query feature vector q is given and we are required to compute its k nearest neighbors. The indexing scheme then works as follows:

Each B+ tree is used to find the position where q would be inserted in it. The data points in the neighborhood of q are then retrieved. The data points retrieved from all the B+ trees are merged into a single set and the nearest t data points ($t \geq k$) from q are chosen based on the weighted distance metric. In this procedure, the B+ trees are enumerated in the order of decreasing relevance of the corresponding features. By doing so, the most similar data points would likely get enumerated early.

This simple approach reduces the search space drastically by using approximate t-NN samples. [11] shows that the desired k-NN samples lie among the t retrieved samples with very high probability. Therefore, the approximation trade-off is favored. The experiments on input weight vectors by the authors show commendable performance improvement over varying dataset sizes and feature lengths in comparison to many commonly used indexing schemes such as SS-trees, R-trees, k-d trees, SQL and flat files. In fact, they are shown to be an order of magnitude better in response time.

4.3 Further Optimization

In this paper, we further propose that even an exhaustive retrieval from all the feature dimensions is not required. We propose and experimentally validate that using only the most important few dimensions, a retrieval accuracy at par with the use of all of them can be achieved. The number of

dimensions to be used for a particular query iteration is not fixed. We propose an adaptive scheme for dimensionality reduction which provides tremendous gain in efficiency.

Our formulation uses the feature weights for the present query to estimate the optimal number of dimensions. This problem in our case is posed in a pretty simple manner. Unlike scenarios where inherent manifolds need to be estimated using computationally expensive techniques, our learning scheme provides us a weight vector which favors the more relevant features. This allows us to estimate the reduced number of dimensions to be used with a simple change monitoring method.

In our proposed dimensionality reduction approach, we traverse the dimensions in non-increasing order of weights and merge the samples suggested by each dimension into a master-list. For every new dimension we first check if merging the samples suggested by this dimension results in any change in the master list. If not, then we stop the traversal and return the current list as the final result set.

This simple scheme saves a lot of computations especially in later iterations when the learning guided metric is heavily biased towards a few good features. Experimental results in favor of this optimization are presented in Section 7.

5. LEARNING

Even the most complex features which can be extracted from images are far from satisfactory when it comes to semantic retrieval capabilities. This necessitates external input of semantics into the retrieval loop. As the human user is the best interpreter of visual content, Huang [23] proposed the use of feedback from the user on the relevance of images displayed to him in response to his query. They proposed a series of optimized methods for tuning the retrieval to the user's intent iteratively. Such approaches are generally studied under the class of intra-query or Short Term Learning(STL).

In this section we discuss the approaches we have incorporated for learning in FISH. We discuss intra-query or short term learning in Section 5.1 and then our proposed approach for inter-query or long term learning in Section 5.2.

5.1 Short Term Learning

Researchers have extensively explored the possibilities of effectively utilizing the semantic input from the user in the form of relevance of the result images to the query [28]. They try to infer the semantic intent of the user from the combination of the relevant and irrelevant images. They then use this learned information to tune the the retrieval iteratively to the user's intent.

Numerous techniques exist for absorbing this feedback each with it's own flavor [10]. In the FISH system, we have implemented a representative subset of these techniques, from which the user can choose one during operation. These include: (1) Delta Mean, (2) Inverse Sigma, and (3) Discriminative Variance Most of these techniques operate by biasing the distance metric in favor of relevant dimensions. We have selected these techniques from amongst many others primarily keeping in mind the feasibility of real time performance.

We shall now briefly mention the inherent characteristics of these methods. We, in general, use the following notation: $\mu_{j,p}$ and $\mu_{j,n}$ represent the means of the positive and the negative samples for the j^{th} feature and $\sigma_{j,p}$ and $\sigma_{j,n}$ their variances. The x_j^is denote the corresponding features for the i_{th} sample. And n_{p} and n_{n} denote the relevant and the irrelevant subsets of R, the retrieved set of images.

STL Methods.

Delta Mean: If the positive (p) and negative (n) sets are well-separated it returns a high score for the feature. It fails if the *unimodal* constraint on the distribution in the p and n sets is violated as generally happens over the n set.
$s_j = \frac{|\mu_{j,p} - \mu_{j,n}|}{\sigma_{j,p} + \sigma_{j,n}}$

Inverse Sigma: Here only the p set is constrained and the peaking of the p distribution is valued. It fails to utilize the n set. Here
$s_j = \frac{1}{\sigma_{j,p}}$

Discriminative Variance: This also constrains both the p and n sets but adds a discriminative edge to $Inverse\,Sigma$ above. Here
$s_j = \frac{\sigma_{j,n}}{\sigma_{j,p}}$

Any choice of the feature importance based approaches estimates the relative importance of the features in representing the query semantics and uses it to bias the dissimilarity metric. The score s_j is used for incrementally augmenting the weight, w_j, for the corresponding feature and thus tune the retrieval to the user's intent as,

$$w_j^t = \gamma\, w_j^{t-1} + \beta\, s_j \tag{1}$$

where w_js represent the weights for the j^{th} feature after the $(t-1)^{th}$ and the t^{th} iterations. The parameters γ and β control the learning rate. Here s_j can be estimated using any of the above methods.

Processing the relevance feedback from the user with these returns a vector of weights which characterize the relative importance of the features. These weights can now be used for tuning the retrieval by ordering the dimensions for retrieval so as to better reflect the user's intent on the query.

This learning acquired based on the user feedback is an invaluable user validated semantic interpretation of the visual content in those images. However, till date only a handful of approaches exist which do not discard the learning after every query. Most of the learning systems use the intra-query learning for benefiting the retrieval for the present query and start from scratch for the next one and thus fail to capitalize on the expensive and invaluable semantic input from the user.

5.2 Long Term Learning

While short term learning or intra-query learning from feedback has been extensively researched yet its inter-query counterpart in "Long Term Learning(LTL)" has for some reason received little attention.

Initially people talked about query memorization [22]. Literature mainly discusses the use of huge feedback logs and information processing techniques to infer semantics and improve retrieval with passing queries [4, 6, 9, 26]. Most of these approaches are critically dependent on aspects that make them practically infeasible for large online systems. Some are overly dependent on huge amounts of feedback logs [9], while some are concerned with memorization-based iteratively-improving groupings among images by extending the feedback to images beyond the marked ones [26]. Still others use techniques which become infeasible owing to their

computational expense [4, 6, 8, 14]. Few have tried to infer incrementally improving feature weights for improving retrieval but most of them have had limited success [19].

We propose a novel incremental framework for long term learning which seamlessly merges with the rest of the FISH system. In our approach we effectively capitalize on the semantic information about the relevant images acquired at the end of every query session following iterative refinement of weights. We use this semantic interpretation for incrementally updating the semantics of visual content for all the relevant samples in the database. So the image semantics are estimated as relative importance weights for the features similar to the weights for the current query as learned in short term learning iterations.

$$c_{ij} = c_{ij} + w_j \, \rho \, \frac{1}{d_i} \tag{2}$$

Here i and j denote the image and feature respectively. c_{ij} represents the relative importance of the j^{th} feature in the i^{th} image. It reflects the long term learning acquired by the system over all previous queries. The parameter ρ controls the learning rate. $\frac{1}{d_i}$ denotes a biasing and convergence factor which makes the change for the i^{th} sample happen in inverse proportion of this sample's dissimilarity with the query.

This formulation is a weighted increment to the learning accumulated till the end of the present query. The weighing factor stabilizes the learning by using the amount of time (number of query sessions) spent in acquiring it. This incremental modification of c_i for the i^{th} image converges over sessions to the concept in that image.

These image centric weights when incorporated in the similarity metric further tune the retrieval towards images which are closer semantic matches to the query. The formulation for the dissimilarity gets modified to

$$d_i = f(x_i, \, q, \, w, \, c_i) \tag{3}$$

A crucial aspect of real world systems is their seamless performance on dynamic datasets, primarily data additions. The critical issue in such a dynamic scenario is the manner in which learning can be transferred to the new samples and the number of sessions they need before they start showing up as relevant in the top few images. For a query image, which is not in the database, the weights obtained through STL are used as the initialization of LTL.

This proposed approach for long term learning seamlessly merges into the FISH retrieval framework with a minimal overhead of parallel on-line computation, independent of the retrieval. The idea of relative feature relevance based tuning of the retrieval to the query semantics makes it optimal for scalability. We showcase the benefits of using this long term learning across queries in Section 7.

6. PERFORMANCE STUDY

In this section we present our performance model for validating our claims of the FISH system. Our experiments fall into four categories, namely, (1) System, (2) Relevance feedback, (3) Long Term Learning, and (4) Optimizations.

The *system* related experiments validate the performance claims in terms of scalability and response times. Next we discuss a series of experiments which showcase the *accuracy* of our system as a result of effective and efficient use of relevance feedback from the human user. We show our improvements in performance using *precision* as the main performance metric. We also use a relatively new approach of *rank reduction* to express the fine improvement in system response across feedback iterations.

Following it we present a series of results showcasing the effectiveness of the proposed framework for inter-query or long term learning through precision gain. In the last part, we present some results on the optimizations we have incorporated in the system to improve performance. We experimentally validate our claim on dimensionality reduction by restricting the comparisons only to the top few dimensions rather than an exhaustive retrieval across all the feature dimensions.

We have used two types of datasets for our experiments. For the first of these datasets, we collected around 12000 real images by mixing the Corel dataset, images crawled from Flickr with research permissions, Caltech datasets and some other freely available small collections. This set is manually annotated. The data set is a mixed set of 58 concepts like flowers, trains on tracks, horses in the fields, cars, buildings, mountains, surfers in sea, ships at sea, meadows, aeroplanes, guns, cycles, buses and animals.

The selection of samples does not showcase any inherent visual discriminability among classes. We used this annotated set for all our accuracy validation experiments. For our system related experiments we populated a huge dataset of *1 Million* feature vectors. Approximately $400,000$ are above features extracted from real images collected from the above sources while the rest were synthesized.

The experiments involving response times and scalability were performed on a machine powered by Intel Xeon 1.6 Ghz processor and 8 GB RAM, running on Fedora Core 5 64-bit operating platform.

7. EXPERIMENTAL RESULTS

Performance - Scalability and Speed.

Scaling of the database in terms of the number of samples, feature dimensions and user sessions all add a computational cost. This expectedly leads to slow response for the query. Through our experiments, we show that our index structure efficiently adapts to the scale-up without much change in response time characteristics. The change in response to an increase in the number of samples in the database is very gradual and low (as shown in Figure 2). The response time in this graph has been averaged over a sets of 1000 queries picked randomly from the dataset images ensuring they are distributed across most of the classes to avoid any biased estimates. Each set was used to estimate average response after insertion of every 25000 samples till *1 Million* were inserted.

The next experiment, we portray the behavior of the system in a dynamic setting where the database is continuously changing. We simulate the environment by inserting every query into the database. The plot in Figure 3 of insertion time Vs the total sample count till that query showcases the efficient absorption of the new image into the indexing structure. This experiment effectively shows that the time to insert new images into the index structure is very small. The sharp change in insertion time is observed at the time which can be mapped to splits in the B+ tree leaves.

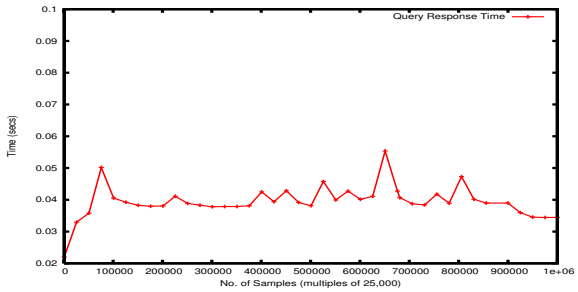

Figure 2: Graph shows the response time for queries with dynamic scaling of the database.

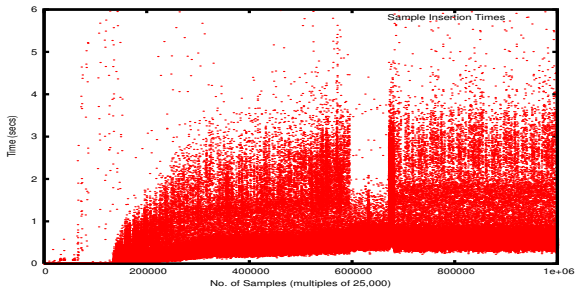

Figure 3: Graph shows the sample insertion time as required for the dynamically scaling database.

Short Term Learning.

Here, we first present the results of the experiment we conducted measuring the average accuracy of the results returned in response to a random set of 1000 queries. In each iteration the top 48 equivalent to first 3 pages of result images received feedback. We plot the results in terms of precision Vs iterations in Figure 4. As the approaches for relevance feedback are similar in their basic nature we show results based on discriminative variance method only. Figure 4 showcases the improvement in accuracy with subsequent feedback iterations over the same query. As is evident from the sharp rise of the accuracy over the initial few iterations, the system readily absorbs the relevance feedback from the user and iteratively tunes the retrieval to the user's intent. The rise slows down over the later iterations and finally nearly flattens out showing convergence of retrieval precision.

The basis of all our claims on performance and effectiveness is the belief that the data is clustered along only a few

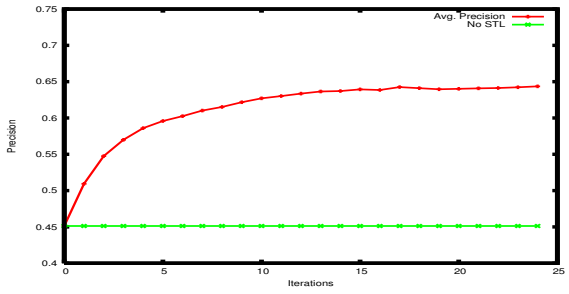

Figure 4: Graphs shows the Precision improvement with user feedback iterations or short term learning.

features for any given category of images. This is best expressed by the weight bias learned by the system towards a few of the features. We conducted an experiment observing the feature weights over iterations. We show the results in Figures 5 and 6 for only a couple of randomly picked samples due to space constraints. The plots clearly show that the initially-unbiased weights curve reaches a highly skewed bias in favor of a few features in the first few iterations itself.

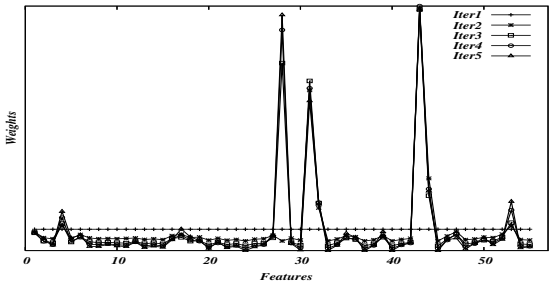

Figure 5: Plot shows how a few of the features become much more important than the rest with feedback iterations.

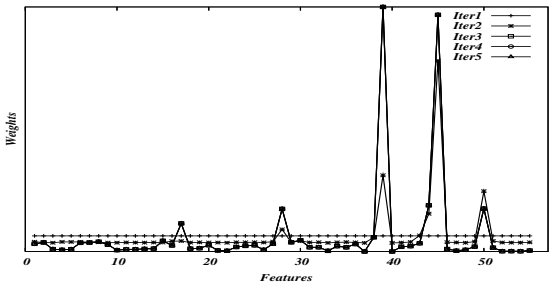

Figure 6: Another example of feature weight convergence

Next, we present an experimental result which showcases the improvement in retrieval accuracy with feedback iterations in terms of the *ranks* of the relevant samples. This method captures the improvement in retrieval performance at a finer rank rearrangement level than the standard precision Vs iteration plot. In this approach we propose to compute the total summed up ranks of the top N relevant samples in the ranked retrieval set where the first sample is the most similar one and is ranked 1. We keep this N as 10 for our experiment. The guiding principle behind this method is the belief that in the ideal case all the top N samples will be members of the query category. As a result there is a lower bound to which the total ranks should ideally converge iteratively.

In Figure 7, we plot the values after down-shifting the entire y-axis by the lower bound. As portrayed by the plot the total rank plot averaged over a set of queries converges to nearly zero on the y-axis. This clearly show the effectiveness of the learning approaches we have adopted in our system. The smooth reduction shows effective incremental learning.

The chains of images showcase the retrieval performance on the working system for a randomly selected query. The first sub-chain corresponds to the first iteration where there are relevant and quite a few irrelevant samples, Figure 9. As

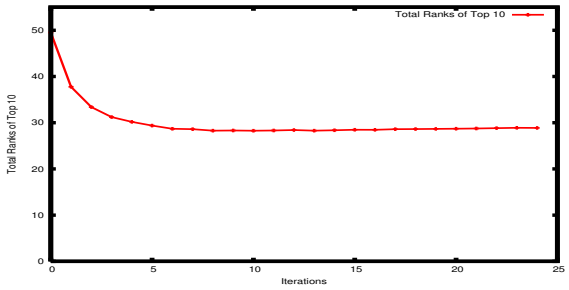

Figure 7: Plot shows how the Total Rank of the Top 10 relevant samples converges towards the ideal state with feedback iterations.

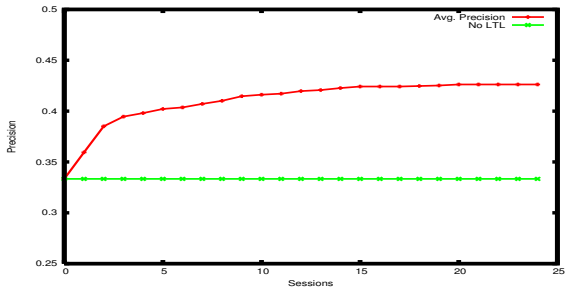

Figure 8: Plot shows how the precision for the first iteration itself improves with long term learning.

can be seen in the subsequent chains (representing subsequent feedback iterations) the number of irrelevant samples continuously falls. This visually showcases that in spite of the variations of the low level descriptors for the images of 'trains on tracks' the learning enables the system to effectively interpret the concept and retrieve iteratively improving results, through short term learning.

Long Term Learning.

In this series of experiments, we first showcase the improvement in retrieval accuracy of the system across query sessions (users). We plot the precision of the first iteration averaged over a random set of 1000 queries populated like in *short term learning* above. In each iteration the top 48 results received feedback. The simulation ran for 5 iterations for each query for 25 user sessions for each. The accuracy for the next session benefits from the learning acquired in the previous ones through the inter-query or long term learning method (see Figure 8).

When we perform weight-driven long term learning the idea is to identify the features that are relevant to the present query and use their importance to update their importance to their parent sample image. If mapped back to the image pixels these weights will have higher values for the relevant features – like in the case of a histogram where the highest weighted bin assumes maximum significance to the image. We can visually evaluate the performance of the system by analyzing the pattern in learning. We map the weight vector back to image pixels by making the pixels corresponding to the highest weight, the brightest. The result is a grayscale image where the brighter regions in the image are the most relevant in that image based on the current level of understanding of the image's semantics by the system. This bias

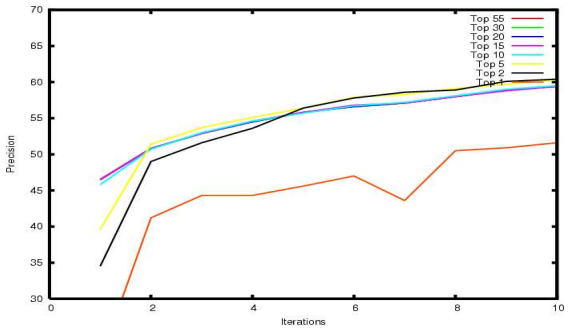

Figure 11: Comparison of Precision across Iterations for exhaustive and optimal number of features

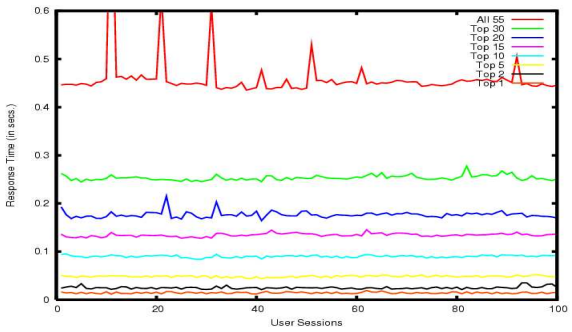

Figure 12: Response Times across Iterations increasing number of features

should ideally improve with sessions/users making the relevant regions brighter while the rest darker leading to a naive region of interest extraction based on long term memory. We have included a few of the sample dataset images with their grayscale images in Figure 10.

Performance Optimizations.

We gain tremendous efficiency with a light trade-off in accuracy by performing an *approximate* $k-NN$ retrieval. We achieve this by retrieving only a small set of samples independently from each of the feature dimensions and merging them to get the final set. We claim that even an exhaustive search spanning all the features is excess. Given the fact that we do a similarity search the loss in terms of accuracy is negligible in view of the tremendous reduction in response time. We present two graphs here validating our claims.

Figure 11 plots the precision Vs iteration curves using increasing number of features. The other graph in Figure 12, presents the response time Vs iteration curves for the same scenario. As can be seen from Figure 11 the accuracy with very few of the most important features nearly matches that achieved by using all the features while at the same time the gain in response time is tremendous as can be seen in Figure 12. This tremendous gain in response time for a possible yet negligible loss in accuracy showcases our claims on the proposed method for dimensionality reduction.

The results of the extensive experiments analyzed above validates our claims on the capabilities of our proposed framework for **F**ast **I**mage **S**earch from **H**uge databases.

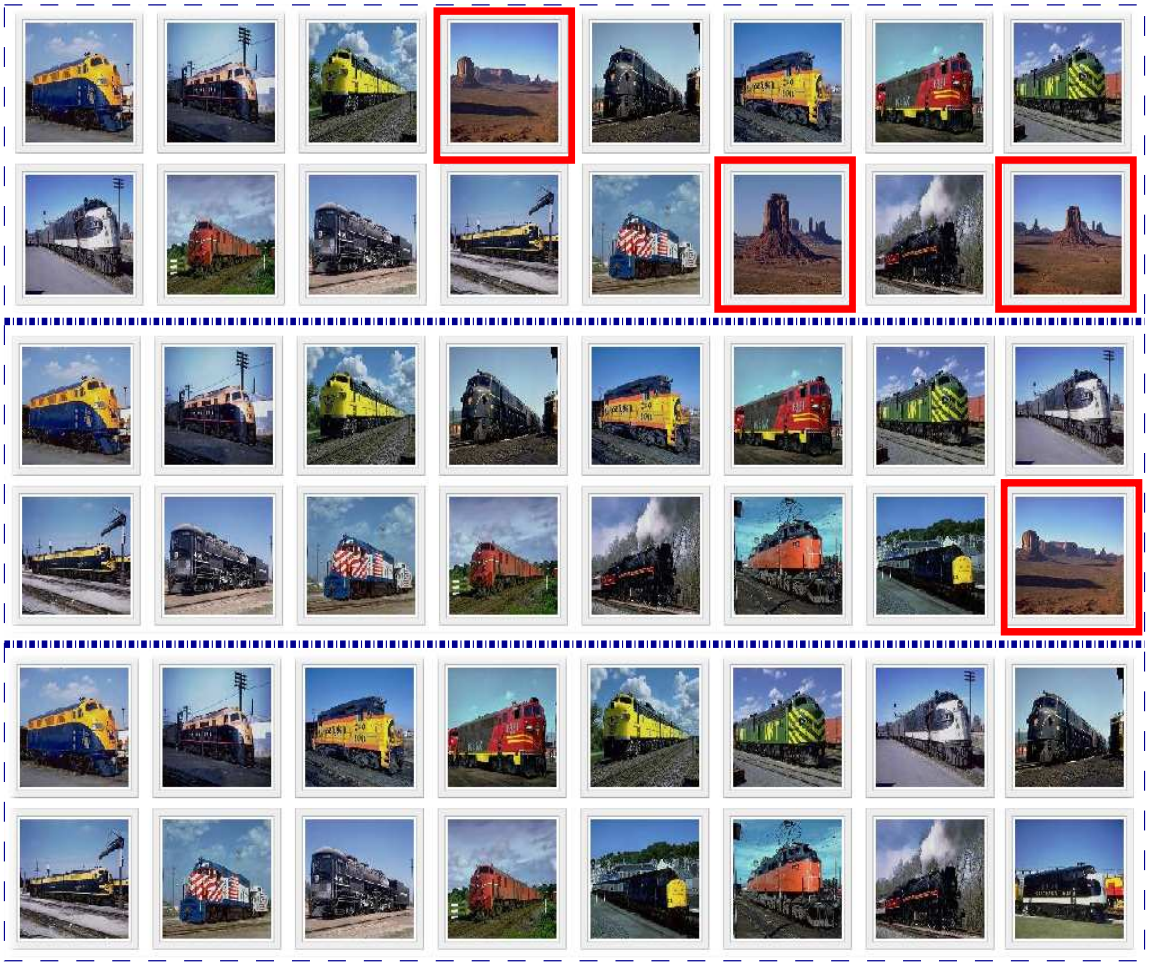

Figure 9: Set of ranked results returned by the System in the 1^{st}, 2^{nd} and 3^{rd} iterations (Irrelevant results highlighted).

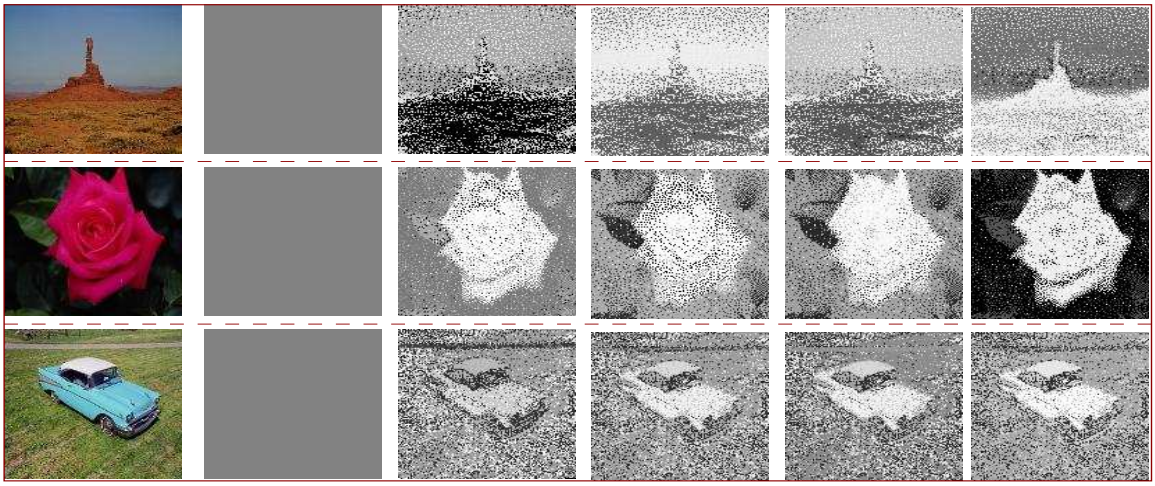

Figure 10: The samples show the progressively improving understanding of image content through Long Term Learning, with passing user sessions.

8. CONCLUSIONS

We have developed and presented a complete end to end system for interactive image retrieval. Our framework seamlessly scales to huge databases. It uses relevance feedback for improving intra and inter query retrieval. Our interface optimizes the interaction by minimizing the load on the user. We have experimentally validated all our claims on performance while reiterating the belief that though feature dependent, the retrieval improves commendably with *short* and *long* term learning. FISH has been designed and developed on the principles of modularity allowing for effortless modifications. In future we would like to enable FISH to also handle multiple concepts in images.

9. REFERENCES

[1] Airliners.net.

[2] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[3] C. Carson, M. Thomas, S. Belongie, J. Hellerstein, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. In *Proc of the Intl Conf on Visual Information and Information Systems*, pages 509–516, 1999.

[4] P. Cord, M.; Gosselin. Image retrieval using long-term semantic learning. *Intl Conf on Image Processing*, pages 2909–2912, 8-11 Oct. 2006.

[5] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *J. Intell. Inf. Syst.*, 3(3-4):231–262, 1994.

[6] J. Fournier and M. Cord. Long-term similarity learning in content-based image retrieval. In *Intl Conf on Image Processing*, Rochester, New-York, USA, September 2002.

[7] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Special Interest Group on Management Of Data Conf*, pages 47–57, 1984.

[8] D. Heisterkamp. Building a latent semantic index of an image database from patterns of relevance feedback. *Pattern Recognition, 2002. Proc. 16th Intl Conf on*, 4:134–137 vol.4, 2002.

[9] C.-H. Hoi and M. R. Lyu. A novel log-based relevance feedback technique in content-based image retrieval. In *Proc of the ACM Intl Conf on Multimedia*, pages 24–31, New York, NY, USA, 2004. ACM.

[10] T. Huang and X. S. Zhou. Image retrieval with relevance feedback: from heuristic weight adjustment to optimal learning methods. *Proc. of the Intl Conf on Image Processing*, 3:2–5 vol.3, 2001.

[11] N. Jammalamadaka, V. Pudi, and C. V. Jawahar. Efficient search with changing similarity measures on large multimedia datasets. In *Conf on Multimedia Modeling*, pages 206–215, 2007.

[12] E. Kasutani and A. Yamada. The mpeg-7 color layout descriptor: a compact image feature description for high-speed image/video segment retrieval. In *Intl Conf on Image Processing*, pages I: 674–677, 2001.

[13] R. Kurniawati, J. S. Jin, and J. Shepherd. Efficient nearest-neighbour searches using weighted euclidean metric. In *Proc of the British National Conf on Databases*, pages 64–76, London, UK, 1998. Springer-Verlag.

[14] J. Laaksonen, M. Koskela, S. Laakso, and E. Oja. Picsom - content-based image retrieval with self-organizing maps. *Pattern Recogn. Lett.*, 21(13-14):1199–1207, 2000.

[15] E. Louupias and S. Bres. Key points-based indexing for pre-attentive similarities: The kiwi system. *PAA*, 4(2/3 2001):200–214, 2001.

[16] W.-Y. Ma and B. S. Manjunath. Netra: a toolbox for navigating large image databases. *Multimedia Systems*, 7(3):184–198, 1999.

[17] J. M. Martínez. Mpeg-7: Overview of mpeg-7 description tools, part 2. *IEEE MultiMedia*, 9(3):83–93, 2002.

[18] D. Messing, P. van Beek, and J. Errico. The mpeg-7 colour structure descriptor: Image description using colour and local spatial information. In *Intl Conf on Image Processing*, pages I: 670–673, 2001.

[19] H. Muller, W. Muller, D. Squire, S. Marchand-Maillet, and T. Pun. Long-term learning from user behavior in content-based image retrieval. *Tech. Rep. 00.04*.

[20] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. H. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The qbic project: Querying images by content, using color, texture, and shape. In *Storage and Retrieval for Image and Video Databases*, pages 173–187, 1993.

[21] M. Ortega, Y. Rui, K. Chakrabarti, S. Mehrotra, and T. S. Huang. Supporting similarity queries in MARS. In *ACM Multimedia*, pages 403–413, 1997.

[22] R. W. Picard, T. P. Minka, and M. Szummer. Modeling user subjectivity in image libraries. In *IEEE Int. Conf. On Image Processing*, volume 2, pages 777–780, Lausanne, Switzerland, 1996.

[23] Y. Rui, T. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: a power tool for interactive content-based image retrieval. *Circuits and Systems for Video Technology, IEEE Transactions on*, 8(5):644–655, Sep 1998.

[24] E. D. Sciascio, G. Mingolla, and M. Mongiello. Content-based image retrieval over the web using query by sketch and relevance feedback. In *Proc of the Intl Conf on Visual Information and Information Systems*, pages 123–130, London, UK, 1999.

[25] D. A. White and R. Jain. Similarity indexing with the ss-tree. In *Proc of the Intl Conf on Data Engineering*, pages 516–523, Washington, DC, USA, 1996. IEEE Computer Society.

[26] T. Yoshizawa and H. Schweitzer. Long-term learning of semantic grouping from relevance-feedback. In *Proc of the ACM SIGMM Intl workshop on Multimedia Information Retrieval*, pages 165–172, New York, NY, USA, 2004. ACM.

[27] J. Zhang, X. Zhou, W. Wang, B. Shi, and J. Pei. Using high dimensional indexes to support relevance feedback based interactive images retrieval. In *Proc of the Intl Conf on Very Large Data Bases*, pages 1211–1214, 2006.

[28] X. S. Zhou and T. S. Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia Systems*, 8(6):536–544, April 2003.