An Optimized Interaction Strategy for Bayesian Relevance Feedback

Ingemar J. Cox, Matthew L. Miller, Thomas P. Minka, Peter N. Yianilos *

NEC Research Institute
4 Independence Way
Princeton, NJ 08540

Abstract

A new algorithm and systematic evaluation is presented for searching a database via relevance feedback. It represents a new image display strategy for the `PicHunter` system [2, 1]. The algorithm takes feedback in the form of relative judgments ("item A is more relevant than item B") as opposed to the stronger assumption of categorical relevance judgments ("item A is relevant but item B is not"). It also exploits a learned probabilistic model of human behavior to make better use of the feedback it obtains. The algorithm can be viewed as an extension of indexing schemes like the k-d tree to a stochastic setting, hence the name "stochastic-comparison search." In simulations, the amount of feedback required for the new algorithm scales like $\log_2 |D|$, where $|D|$ is the size of the database, while a simple query-by-example approach scales like $|D|^a$, where $a < 1$ depends on the structure of the database. This theoretical advantage is reflected by experiments with real users on a database of 1500 stock photographs.

1 Introduction

Searching for digital information, especially images, music, and video, is quickly gaining in importance for business and entertainment. As databases expand, more effective search techniques are needed. A search typically consists of a query followed by repeated relevance feedback, where the user comments on the items which were retrieved. Having a good query language is an important first step, though the meaning of "good" strongly depends on the type of media and user population. This paper focuses on optimizing the relevance feedback phase, which by contrast can be done in a fairly universal manner: all information about the media and users is encapsulated in a single stochastic model.

The first step in developing a relevance feedback algorithm is to determine an unambiguous, quantitative measure of performance, by which one algorithm may be compared to another. Many published results are qualitative, e.g. a sketch-based query might be shown to yield images that intuitively appear to match the sketch. Then relevance feedback is provided, yielding images which presumably are better matches. A more quantitative approach is to count how much feedback is required to achieve a well-defined level of retrieval accuracy.

The desired level of accuracy used in this paper is simple: find a particular target item. This target testing measure, introduced in [2], is unambiguous, simple and robust. For example, having a large collection of eagle images will make it easy to retrieve random eagle images, but it will remain difficult to find a particular eagle image on the basis of relevance feedback. Since the measure is simple, it admits a detailed analysis. Our analysis can be used as a platform from which to attack more complex search criteria.

The second step is to determine what information the user can still provide, given that the query language has proven inadequate. Many systems use categorical feedback, where the user indicates what items are in the same category of the target [6, 11]. However, this forces the user to decide on a useful category: one that is not too large or too small and contains the target. This is a drawback in the applications we envision, where the user has little familiarity with the content of the database. Inventing a category should be unnecessary for finding a particular target.

This paper makes the more lenient assumption that the user can make relative judgments between items, i.e. "item A is more relevant than item B." This kind of input still can be used effectively without a notion of category or when no retrieved items are in the same category as the target.

Our retrieval system, `PicHunter` [2, 1], uses a minimally sufficient interface for this kind of interaction (figure 1). A set of n images is displayed. After selecting zero or more images, the user calls up another set of images by hitting the "GO" button. The selected images are not ranked by the user in any way: the selected images are simply intended to be more like the target than the unselected ones. This cycle is repeated until the target appears. At this point, the search is terminated by selecting the target and hitting the "FOUND" button. The set of user responses

* in alphabetical order

Figure 1: The `PicHunter` user interface.

to the images displayed totals 2^n possible combinations of selected images and n possible terminations of the search. The interface is intentionally sparse to permit comprehensive analysis.

The third step is to decide on a representation for the information that the user has provided. A complete representation for the information received is a probability distribution over possible targets, which is what this paper uses. The initial query, if any, provides the starting condition for this distribution, which may be multi-modal or have other complex structure.

This scenario was also considered in earlier papers [2, 1]. The new observation here is that once we decide on (1) target testing, (2) relative judgments, and (3) representing the full target distribution, database retrieval reduces to the specific computer science problem of *comparison searching*. The next section discusses this problem and gives a novel solution for the case when comparisons have stochastic outcomes. This generic algorithm, called "stochastic-comparison search," replaces the image display strategy for `PicHunter` and can be used for relevance feedback with any kind of database. Simulations reported in section 3 suggest better performance with the new approach, and especially contrasts it with repeated query-by-example. Section 4 again contrasts the `PicHunter` approach with repeated query-by-example, this time with real users searching a database of natural scenes.

2 Multidimensional comparison searching

This section presents a new derivation of the algorithm in [2] that better motivates the new display strategy. It is based on a generalization of existing algorithms for comparison searching, so that they can accommodate arbitrary kinds of comparisons, stochastic comparison outcomes, and comparisons made in parallel. These generalizations allow the relevance feedback problem introduced in the last section to be addressed as a comparison searching problem where the user is performing the comparisons.

Comparison searching is the problem of finding T or the closest element to T among a set of elements D, by only making comparisons between an element and T. The familiar binary search algorithm is a one-dimensional special case, where comparisons are of the form "$X < T$?".

For the more general case, the *vantage-point tree* has recently been proposed [12]. It is a variant of the k-d tree which is based on thresholds of distance measurements, i.e. the comparison "$d(X, T) < t$?". The search tree is constructed top-down in a greedy fashion, choosing the vantage point X and threshold t which are expected to prune away as much of D as possible. This can be interpreted as optimization with one step of lookahead. Optimizing this criterion over X is difficult when the distance measure is arbitrary, so a Monte Carlo approach was used: sample several random vantage points and choose the one which minimizes the lookahead cost. For generically distributed data, this greedy algorithm has been shown to perform near-perfect [12].

The comparison "$d(X_1, T) < d(X_2, T)$?", i.e. "is X_1 or X_2 closer to the target?", is more relevant for the relevance feedback scenario. Fortunately, the vantage-point tree is easily generalized to comparisons of this form and the same Monte Carlo approach can be used to find X_1 and X_2 at each step (i.e. both are sampled randomly to find a good pair). The resulting tree, when used with Euclidean distance, is a recursive binary subdivision of the representation space using half-planes. This is similar to oblique classification trees [7], except search time is being minimized, not classification error.

2.1 Nondeterminism

In the retrieval application we envision, a human will be carrying out the comparisons. This requires that the algorithm be able to cope with an imperfectly modeled, possibly nondeterministic comparison operation. In this more general scenario, the comparison "$d(X_1, T) < d(X_2, T)$?" will return "true" with some probability $p(A = 1|X_1, X_2, T)$, and "false" with probability $p(A = 2|X_1, X_2, T)$. For simplicity, assume that the operation is memoryless, except that making the same comparison again is likely to return the same answer. More complex models should be motivated by the failure of a simple one.

The only change that needs to be made to the above algorithm is the criterion for choosing X_1 and X_2 at each step. In the deterministic case, every comparison completely removes part of D from consideration. That is no longer true: elements are only removed with some probability. In other words, each element X has a probability

$p(T = X)$ of being the target T and each comparison operation changes this probability.

The idea is to estimate the number of comparisons left in the search, based on the distribution $p(T = X)$. Call this estimate $C[p(T)]$. Then choose the comparison which minimizes the expected number of future comparisons (the one-step lookahead cost). In other words, the best choice of X_1 and X_2 minimizes

$$(1 - p(T = X_1) - p(T = X_2)) \times \sum_a C[p(T|A = a)]p(A = a|X_1, X_2)$$

$$\text{where } p(A = a|X_1, X_2) = \sum_X p(A = a|X_1, X_2, T = X)p(T = X)$$

and $p(T|A = a)$, defined below, is the distribution over targets after user response a (a is either 1 or 2). The leading factor incorporates the fact that if X_1 or X_2 is the target then there are no further comparisons. The sum over X is over the entire database.

Information theory suggests entropy as an estimate:

$$C[p(T)] \approx -\alpha \sum_X p(T = X) \log p(T = X) \quad (1)$$

for some positive constant α which is irrelevant for the purpose of minimization. Again the sum is over the entire database. This offers an alternative interpretation of minimizing future cost: maximizing immediate information gain. This heuristic has also been used to design classification trees [7].

The remaining question is how to update $p(T = X)$ after the outcome of a comparison $d(X_1, T) < d(X_2, T)$. Since this outcome occurs with probability $p(A = 1|X_1, X_2, T)$, Bayes' rule tells us

$$p(T = X|A = 1) = \frac{p(A = 1|X_1, X_2, T = X)p(T = X)}{p(A = 1)} \quad (2)$$

where the denominator is simply a normalization factor that can be computed at run-time. Since the outcomes of different comparisons are assumed independent, the posterior $p(T = X|A = 1)$ can be considered a new prior $p(T = X)$ for the next comparison. If either X_1 or X_2 is the target, then the search halts and the comparison doesn't occur at all. Therefore, $p(T|A)$ is zero whenever T is X_1 or X_2, and these items will never be presented again.

To illustrate this rule, consider the ideal case:

$$p_{ideal}(A = 1|X_1, X_2, T) = \begin{cases} 1 & \text{if } d(X_1, T) < d(X_2, T) \\ 0.5 & \text{if } d(X_1, T) = d(X_2, T) \\ 0 & \text{if } d(X_1, T) > d(X_2, T) \end{cases} \quad (3)$$

If $A = 1$, all elements farther from T than X_1 will get zero probability. The remaining elements will have uniform probability (assuming no ties). Thus we get the usual algorithm for vantage-point trees.

Now consider the generalization

$$p_{sigmoid}(A = 1|X_1, X_2, T) = \frac{1}{1 + \exp((d(X_1, T) - d(X_2, T))/\sigma)} \quad (4)$$

When $\sigma \to 0$, this is the same as p_{ideal}. When $0 < \sigma < \infty$, there is a smooth transition from probability 1 to probability 0 as T varies. When $\sigma \to \infty$, outcomes are completely random. This formula can be interpreted as p_{ideal} after corrupting the distance measurements with Gaussian noise. The parameter σ can therefore be interpreted as the degree of precision in the distance measurements.

2.2 Related work

Comparison searching with errors has also been studied in the theoretical computer science literature. The algorithm in [10] assumes that the number of errors has a known bound. Nevertheless, their algorithm is similar to the one presented here, in the sense that it minimizes at each step an information-theoretic bound on the number of future comparisons. The algorithm in [9] allows errors to occur at random but requires them to be independent of the comparison and the target and furthermore does not guarantee that the target is found. So while both of these algorithms run in provably logarithmic time, they also operate under more restrictive conditions than stochastic-comparison search.

The general idea of maximizing the expected information from a query has also been pursued in the machine learning literature under the name "Active Learning" or "Learning with Queries" [4]. Active learning techniques have been shown to significantly outperform simple probability ranking for document classification [5]. We know of no application of active learning techniques to database retrieval.

2.3 Simultaneous comparisons

Bringing the problem even closer to reality, suppose multiple comparisons can be made at once. This is readily handled by changing the comparison model.

For example, suppose the user picks the closest element to T, say X_a, from a set of elements $X_1..X_n$. This is equivalent to the $n - 1$ individual comparisons "$d(X_a, T) < d(X_i, T)$?" for $i = 1..n$, $i \neq a$. The probability of this outcome is denoted $p(A = a|X_1..X_n, T)$.

If we assume that all of the comparisons are independent, then

$$p_{indep}(A = a|X_1..X_n, T) = \alpha \prod_{i \neq a} p(A = 1|X_a, X_i, T) \quad (5)$$

where α is a normalizing term that depends on $X_1..X_n$ and T.

Alternatively, generalizing the rule for one comparison leads to the softmin:

$$p_{softmin}(A = a|X_1..X_n, T) = \frac{\exp(-d(X_a, T)/\sigma)}{\sum_i \exp(-d(X_i, T)/\sigma)} \quad (6)$$

Note that transitive ordering of the elements is not being assumed by either of these two models.

Now suppose the user can pick multiple elements, as in the `PicHunter` interface. If k elements are selected, this is equivalent to the $k(n-k)$ individual comparisons "$d(X_a, T) < d(X_i, T)$?" where a is any selected element and i is any unselected element. The probability of this outcome is denoted $p(A = \{a_1..a_k\}|X_1..X_n, T)$.

As before, one can assume the selections to be independent:

$$p_{indep}(A = \{a_1..a_k\}|X_1..X_n, T) = \alpha \prod_i p(A = a_i|X_{a_i}, X_{unselected}, T) \quad (7)$$

where $X_{unselected} = \{X_1..X_n\} \backslash \{X_{a_1}..X_{a_k}\}$. The right-hand side reduces to the previous case. Or one can compute the the softmin between all $\binom{n}{k}$ possible selections of k elements, where the numerator is $\exp(-\sum_i d(X_{a_i}, T)/\sigma)$.

It is an open question how to handle $k = 0$, where the user selects no images at all. The above two models give this action a constant probability, independent of the target and the choices, which may be undesirable.

2.4 Pseudocode

The "stochastic-comparison search" algorithm for comparison searching or relevance feedback works as follows:

1. Find the choices $X_1..X_n$ which minimize

$$(1 - p(T = X_1) - \cdots - p(T = X_n)) \times \sum_a C[p(T|A = a)]p(A = a|X_1..X_n) \quad (8)$$

where $p(A = a|X_1..X_n) =$

$$\sum_X p(A = a|X_1..X_n, T = X)p(T = X) \quad (9)$$

and $C[p(T)] = -\sum_X p(T = X) \log p(T = X)$ (10)

   This step is the crucial difference with previous `PicHunter` algorithms. If the interface allows multiple selections, summing over all a would be prohibitive. As an approximation, `PicHunter` only sums over single selections, which takes $O(n|D|)$ time. All experiments in this paper used the best of seven random choices for $X_1..X_n$.

2. Get the user's response A, i.e. perform the comparisons.

3. If the user indicates that one of $X_1..X_n$ is the target then stop.

4. Compute $p(A|X_1..X_n, T)$ for all potential targets T.

5. Use Bayes' rule to update the target distribution:

$$p(T = X|A) = \frac{p(A|X_1..X_n, T = X)p(T = X)}{p(A)} \quad (11)$$

   The last two steps take $O(|D|)$ time.

6. Iterate back to step 1.

3 Simulation results

This section evaluates the Entropy criterion in stochastic-comparison search by comparing it to other plausible methods for choosing $X_1..X_n$:

Most Probable Let $X_1..X_n$ be the n items which maximize $p(T = X)$. Ties are resolved randomly. This algorithm was used in previous work [2, 1].

Sampling Sample $X_1..X_n$ from the distribution $p(T = X)$.

Query by Example Let $X_1..X_n$ be the n closest items to the winner of the last comparison. This is a favorite approach in systems without relevance feedback [3]. It does not exploit previous comparisons or a model of nondeterminism.

The idea is to simulate a user's responses by sampling from the stochastic comparison model. The database is synthetic, consisting of points uniformly-distributed inside the unit square. This allows databases of varying sizes to be easily drawn. The simulated users used the Euclidean distance measure.

3.1 Deterministic case

Figure 2 plots the empirical average search time for finding a randomly selected target as a function of database size, using the first three search strategies. The number of choices n was two. Comparison outcomes were generated by the p_{ideal} model. In all experiments, the average is over 1000 searches, each with a different target, and the database was resampled 10 times. We see that the performance of these three schemes is comparable, scaling like $\log_2 |D|$. In particular, the Entropy scheme is virtually optimal, with deviations only due to a limited number of Monte Carlo samples.

The Query-by-Example method is quite different, as shown in figure 3; note the change in vertical scale. The Query-by-Example method is not exploiting comparison information very well; its time scales as $|D|^{0.5}$. Of course, the difference between the four schemes can be reduced by increasing n or the dimensionality.

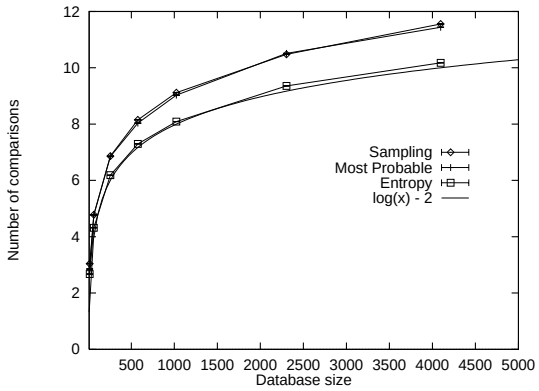

Figure 2: The number of comparisons needed to find an element, for varying database sizes and search strategies. Comparison outcomes were generated according to p_{ideal}. The database contains uniformly-distributed points inside the unit square.

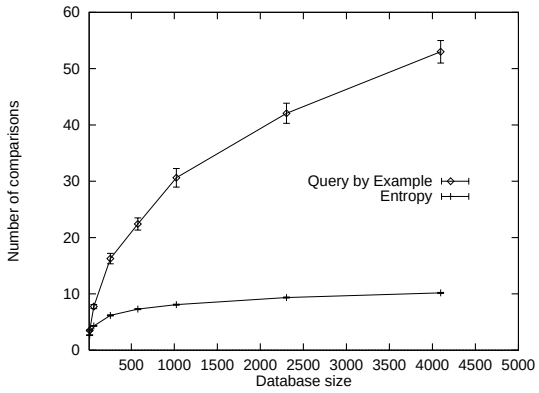

Figure 3: Same as figure 2 but including the Query-by-Example method.

3.2 Nondeterministic case

Figure 4 shows what happens when comparison outcomes are generated by the $p_{sigmoid}$ model, with $\sigma = 0.1$. Increasing the database size causes the unit square to be sampled more and more finely, while the distance uncertainty threshold σ remains the same. Thus it is much harder to isolate a particular target in a large database than in a small one, as would be true in a real situation.

Again, the Sampling and Entropy schemes are similar in search time, which scales like a square root. However, the fragility of the Most Probable scheme is evident here. Figure 5 also reveals a large discrepancy in the Query-by-Example scheme.

An explanation for this is that the Most Probable and Query-by-Example schemes tend to choose items which are close together in feature space—exactly when comparisons are most unreliable. Entropy-minimization, by contrast, automatically chooses comparisons for which answers are reliable. The Most Probable scheme also does not properly exploit broad and nonuniform distributions, or distributions which are multi-modal. Furthermore, a multi-modal distribution causes this scheme to switch to different parts of the database between iterations, which is disconcerting to a real user.

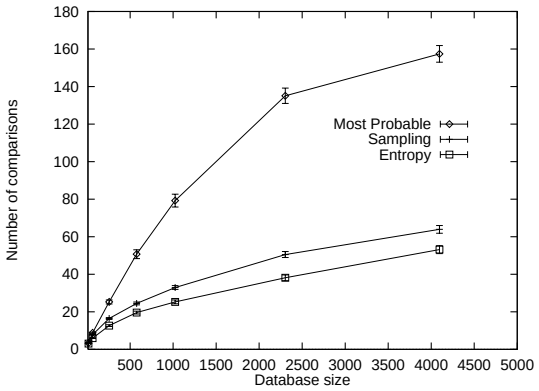

Figure 4: Here comparison outcomes were generated according to $p_{sigmoid}$ with $\sigma = 0.1$.

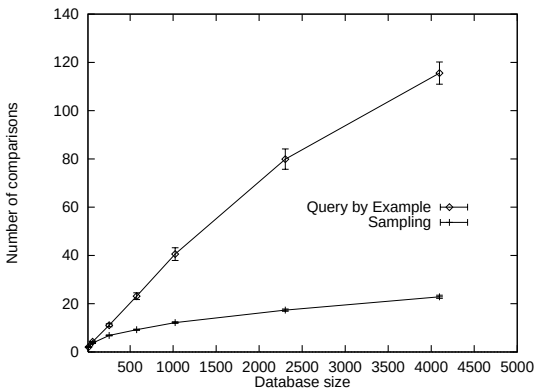

Figure 5: Same as figure 4 but including the Query-by-Example method.

4 Experimental results

This section compares different feedback schemes with real users in an image retrieval setting. The database was a commercial collection of stock photographs. We used the 1,500 images from 15 thematic CDs of 100 images each: bald eagles, air shows, Arabian horses, Los Angeles, Israel, Africa, flowers, the arctic, patterns, etc. This assortment makes it easy to find a random image of flowers but rather difficult to find a specific image of flowers, since there are 100 of them. Each image was 128x192 pixels with 24 bits of color.

To train the model, a subject was shown two randomly-

selected database images X_1, X_2 beside a randomly-selected target T. The subject was asked to choose the image that was "closest overall" to the target. This yielded 3,681 samples over 9 different subjects.

The system assumed independent comparisons (p_{indep}) generated by $p_{sigmoid}$. The distance metric was a weighted sum of distances based on the 18 image features in [2] plus keyword annotations [1]. The σ parameter and the mixing weights were chosen to approximately maximize the likelihood of the training trials.

A disjoint set of six subjects was used for testing. Here the number of images per display was $n = 9$ and users could select multiple images, except when using the Query-by-Example strategy. The entropy-minimization strategy was not implemented at the time so the Most Probable strategy (section 3) was used instead. The Query-by-Example strategy required 33 inputs in average, while the Most Probable strategy required 25. The full experiment is described in [8]. Based on the results of the previous section, the performance difference is expected to increase with a bigger database and more accurate user model. As this paper went to press, a preliminary experiment was run with two subjects which placed entropy-minimization ahead of both of these two strategies.

Simulations on this database give the same performance ranking, with entropy-minimization best, which is intuitive in the following way. When using the Most Probable scheme (section 3), the system gives the impression of being smart, since it shows many similar images, e.g. images of aircraft. On the other hand, the entropy-minimization scheme tries to display reasonably diverse images, e.g. aircraft and eagles. For the purpose of minimizing search time, the latter is better, at the early stages of search. This seems obvious, but it contradicts the widely-practiced belief that a system should be judged on the basis of a single iteration of feedback.

On this database, the `PicHunter` algorithm runs well within interactive time. The slowest step is optimizing the display, which takes three seconds.

5 Conclusions

`PicHunter` introduced a relevance feedback algorithm which incorporated a predictive model of user input. This work represents two additional innovations:

- A relevance feedback algorithm which deliberately minimizes the amount of feedback.
- A quantitative, domain-independent analysis of relevance feedback algorithms.

The stochastic-comparison search algorithm can be incorporated into virtually any of the existing database retrieval systems. One simply needs to be able to convert a query result into a distribution over targets. Existing systems already measure distance between database items; these measures can be plugged into the $p_{sigmoid}$ model, with the σ parameter fitted to user interaction logs.

This relevance feedback algorithm is also strong enough to be used without a query language. This is useful for domains where a query language would be awkward or new domains for which query languages have not yet been devised, e.g. textures, faces, artwork, 3D models, and sounds.

Acknowledgments

We are grateful to Thomas Papathomas for orchestrating the human study, to Bob Krovetz for helpful discussions, and to the growing list of test subjects who continue to play a crucial role in the refinement of our approach.

References

[1] I. J. Cox, J. Ghosn, M. L. Miller, T. V. Papathomas, and P. N. Yianilos. Hidden annotation in content based image retrieval. In *Proc IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 76–81, 1997.

[2] I. J. Cox, M. L. Miller, S. M. Omohundro, and P. N. Yianilos. Pichunter: Bayesian relevance feedback for image retrieval. In *Int. Conf. on Pattern Recognition*, pages 361–369, 1996.

[3] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32, 1995.

[4] Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. In *Advances in Neural Information Processing Systems*, Cambridge, MA, 1993. MIT Press.

[5] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proc. of ACM-SIGIR Conf. on R&D in Information Retrieval*, Dublin, Ireland, July 1994. Springer-Verlag.

[6] T. P. Minka and R. W. Picard. Interactive learning using a "society of models". *Pattern Recognition*, 30(4), 1997.

[7] K. V. S. Murthy. *On Growing Better Decision Trees from Data*. PhD thesis, Johns Hopkins University, 1995.

[8] T. V. Papathomas, T. E. Conway, I. J. Cox, J. Ghosn, M. L. Miller, T. P. Minka, and P. N. Yianilos. Psychophysical studies of the performance of an image database retrieval system. In *IS&T/SPIE Symposium on Electronic Imaging: Science and Technology, Conference on Human Vision and Electronic Imaging III*, 1998.

[9] A. Pelc. Searching with known error probability. *Theoretical Computer Science*, 63:185–202, 1989.

[10] R. L. Rivest, A. R. Meyer, D. J. Kleitman, K. Winklmann, and J. Spencer. Coping with errors in binary search procedures. *Journal of Computer and System Sciences*, 20:396–404, 1980.

[11] Y. Rui, T. S. Huang, and S. Mehrotra. Content-based image retrieval with relevance feedback in MARS. In *Proc. of IEEE Int. Conf. on Image Processing*, Santa Barbara, CA, October 1997.

[12] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1993.