PicHunter: Bayesian Relevance Feedback for Image Retrieval*

Ingemar J. Cox, Matt L. Miller, Stephen M. Omohundro, Peter N. Yianilos

NEC Research Institute
4 Independence Way
Princeton, NJ 08540

Abstract

This paper describes PicHunter, *an image retrieval system that implements a novel approach to relevance feedback. It uses Bayesian learning based on a probabilistic model of a user's behavior. The predictions of this model are combined with the selections made during a search to choose the images to display. The details of our model were tuned using an offline learning algorithm. For clarity, our studies were done with the simplest possible user interface but the algorithm can easily be incorporated into a system which supports complex queries. Even with this constraint and simple image features,* PicHunter *is able to locate randomly selected targets in a database of 4522 images after displaying an average of only 55 groups of 4 images which is over 10 times better than chance.*

1 Introduction

The rapid expansion of computer networks and the dramatically falling cost of data storage are making multimedia databases increasingly common. Digital information in the form of images, music, and video is quickly gaining in importance for business and entertainment. The growth in multimedia databases is creating the need for more effective search techniques. Consequently, there is increasing interest in multimedia retrieval, particularly for image databases [8, 2, 7, 18, 3, 4, 17, 6, 5, 12].

At least three classes of search may be identified. In "target search" users seek to find a specific target image (eg. a graphic artist looking for a specific stock photo). In "category search" users seek one or more images from general categories such as: "sunsets" or "birds". In "open-ended browsing" users have only a vague idea of what they're looking for (eg. an interior decorator might begin with one scheme in mind, but end up with a different one after seeing images from a design database). Each search class is important, but it is hard to define what *correct* behavior means for the second and third classes. We therefore focus on *target search* and measure the performance of a system by the average number of images viewed before a random target is found.

Most experimental and commercial image retrieval systems begin each search with an explicit query. A user of such a system must somehow describe the desired image or images. Several techniques have been tried including keyword search of annotated images, query by visual example, sketch-based retrieval, and specification of image features. It is difficult to precisely specify an image by any of these methods and so there is ambiguity remaining about the desired image after the initial query.

A promising approach to reducing this ambiguity is "relevance feedback". This is an iterative process in which the user indicates the relevance or irrelevance of retrieved items. While relevance feedback has been extensively studied for text retrieval, there has not been extensive investigation of its optimal use for image retrieval.

Most content-based image retrieval systems include some form of query by visual example. By drawing successive queries from images retrieved in earlier searches, users can obtain a crude form of relevance feedback. But this relevance feedback exhibits no adaptation. Each retrieval responds to a specific query and ignores the previous history. In these systems, retrieval ability is entirely due to the similarity metric employed. The metrics are typically a weighted sum over a set of features, the weights often being determined manually by the user (eg. [4, 6]). Research has been directed at finding useful feature sets under constraints such as high retrieval speed and the ability to search compressed images [12, 13].

Research on relevance feedback for textual database systems, by contrast, has focused on adaptively refining a user's initial query to more accurately select the desired data. [16]. Recently Minka and Picard [10] proposed an interactive learning system that takes this basic approach toward image data. Their system forms disjunctions between initial image feature groupings according to both negative and positive feedback given by the user. In an image database system, these learned disjunctions would act as complex, refined queries, much like the queries in textual systems.

In contrast, PicHunter refines its *answers* in reply to user feedback, rather than refining any query. The theory behind PicHunter is described in Section 2 and incorporates both the responses from the user and a probabilistic model

of that user. The user interface is described in section 3. The user model was based on the experiments described in Section 4. Section 6 proposes several extensions of the work.

2 Bayesian Relevance Feedback

In this section, we develop a general, Bayesian framework for relevance-feedback algorithms. A user is looking for a specific datum in a database by means of a series of display/action iterations. Assume

I denotes the database. In the case of `PicHunter`, I is a set of images, but the basic framework developed here holds just as well for other types of data.

$I_t \in I$ is the datum being searched for.

U denotes the user that is searching for the target I_t.

D_k is the information displayed by the system during iteration k. In the case of `PicHunter`, this is a small set of images. But the framework also applies if additional information is displayed, as will be discussed in Section 6.

A denotes the set of possible actions that the user can take in response to each display. This is defined by the user interface. We assume that A includes some actions that terminate the search.

$a_k \in A$ is the action taken in iteration k.

We now ask the question: given a history of display/action pairs, what is the probability that a given datum, $I_i \in I$, is equal to the target, I_t? To find this probability, we first make a simple application of Bayes' rule and express it in terms of the probability that the user would take the given sequence of actions if the target were I_i:

$$P\{I_i|a_1 \ldots a_k, D_1 \ldots D_k, U\} \propto P\{I_i|D_1 \ldots D_k, U\} P\{a_1 \ldots a_k|I_i, D_1 \ldots D_k, U\}$$

We can assume, for our present problem, that the probability of any given datum being the target is independent of who the user is or what the sequence of displays has been. So $P\{I_i|D_1 \ldots D_k, U\} = P\{I_i\}$ is simply the prior probability that I_i was chosen as the target.[1]

Next, we make the simplifying assumption that the user's action in each iteration depends only on the target and on the current display, i.e. it is independent of any of the previous iterations. This assumption may very well be false, since such psychological phenomena as mental set probably affect the user's behavior (see, for example [11]). However, the effects of past history are probably small compared to the effects of the situation at the moment that the user acts.

Making the above two assumptions, then, we have

$$P\{I_i|a_1 \ldots a_k, D_1 \ldots D_k, U\} \propto P\{I_i\} \prod_k P\{a_k|I_i, D_k, U\}$$

[1]This assumption might not be true in open-ended browsing applications, as discussed in Section 6.

From this it follows that, after each iteration, we can compute the change in probability of a given datum being the target, if we have a function for estimating the probability of each possible user action:

$$P\{a_k|I_i, D_k, U\} \approx KS(a_k, I_i, D_k, U)$$

where K is an arbitrary constant. Such a function represents a model of user behavior that can be tested and tuned experimentally, outside the context of our information retrieval system. One such model is developed in Section 4.

Our algorithm maintains the current distribution of probabilities that the data in the database are the target of the search. In each iteration of the search, the N most probable data are shown to the user, and the user's response is then used to update the probability distribution. We initialize the probability distribution with a flat prior (constant value). But it might just as easily be initialized with a distribution computed from an explicit query when used in a query-based system. See [?] for pseudocode.

It should be noted that displaying the N most probable data in each iteration is not likely to be the optimal strategy. This is discussed further in Section 6.

The above algorithm is extremely general. To instantiate it in an actual program requires that the set of possible actions, A, be defined by designing a user-interface, and that a reasonable user model, S, be developed. We now cover each of these tasks in turn, first describing the user-interface of `PicHunter` in detail, and then describing the development of our current user model.

3 User interface

We wish to keep the user interface as simple as possible, placing all our emphasis on intelligent use of the information provided by the user. This contrasts with approaches such as QBIC [4], in which the effort is to design a maximally informative user interface. The entire user interface of `PicHunter` is shown in Figure 1. At any given time during a search, four images are displayed on the screen. The user can select one or more images by clicking on them with the mouse. Selected images are highlighted by red borders. Images may be deselected by clicking on them again.

After selecting zero or more images, the user calls up the next set of four images by hitting the "GO" button. The user continues selecting images and hitting "GO" until the target appears. At this point, the search is terminated by selecting that image and hitting the "FOUND" button. The "ABORT" button terminates the search before the target image is found. The set, A, of possible user responses to the images displayed, contains 20 elements. 16 possible combinations of selected images, plus 4 possible terminations of the search.

We consider it an open question whether this system can be made to work well for all users without the addition of richer controls, such as buttons that the user can press to independently indicate whether an image resembles the target's color, shape, texture, or composition. If additional

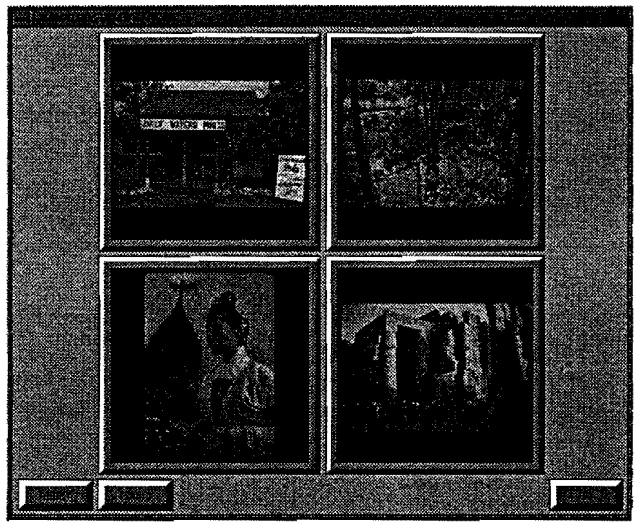

Figure 1. The PicHunter user interface.

controls are unnecessary, then it may be employed in environments where more complex interfaces would be inappropriate. For example, the system could be implemented with a touch-screen and installed in a fabric store to help customers find fabric designs. However, if a more complex interface is required, then the evaluation based on a minimal interface should prevent the final interface from being more complicated than necessary. Of course, the Bayesian framework we are employing here is applicable to any user interface.

4 Development of the User Model

To develop the user model, we make two simplifying assumptions. First, we assume that all users are identical, so U can be ignored. This assumption, of course, is false, and there are interesting possibilities for future work in differentiating between different types of users and in learning specific details about individual users (see [9] for a related project). However, this work should be motivated by failures that arise as a result of assuming all users are the same, so we begin with this assumption. Second, we assume that the user's behavior will be correlated to some degree with a small set of image feature values. This is one of the basic simplifications used in other content-based image retrieval systems [9, 2, 4], where images are indexed by the values of precomputed features such as average color, texture, shape, etc. The set of features we use here is far less sophisticated than the features employed in other systems, but our system can easily accommodate additional features, and we will be adding them in the future. Table 1 describes the 18 features we use.

After computing the 18-element feature vectors for each image in the database, we conducted an informal experiment to look for the relationship between features and user actions. In each iteration of the experiment, the subject was shown a display of four randomly-selected images beside a single, randomly-selected "target" image. The subject was

1	Image width as a fraction of the maximum width in the database			
2	Image height as a fraction of the maximum height in the database			
3 – 13	Percentages of pixels that fall into the following ranges of HSV colorspace.			
		H(°)	S(%)	V(%)
	black	0...360	0...100	0... 3
	grey	0...360	0... 15	2... 85
	white	0...360	0... 15	80...100
	red	−70... 25	10...100	5...100
	orange	15... 50	10...100	2...100
	yellow	25... 80	10...100	8...100
	green	75...185	10...100	2...100
	blue	175...260	2...100	2...100
	purple	255...300	10...100	2...100
	brown	−50... 80	5... 85	1... 40
	pink	−70... 25	10... 60	2...100
14	Mean saturation			
15	Median intensity			
16	Contrast[a]			
17	Number of "edgels" in the image.[b]			
18	The same as feature 17, but thresholded at 10%.			

[a] Y0 is a brightness value that 1/3rd of the pixels are below, and Y1 is a brightness value that 2/3rds of the pixels are below. Feature 16 is Y1-Y0.

[b] computed by first converting the image to grey-scale, then filtering it with a simple, 3 × 3 Laplacian filter and thresholding the result at 20%.

Table 1. Description of feature set.

then asked to select the image from the display that was "closest overall" to the target image. If none seemed close at all, or all seemed equally close, then the subject was allowed to leave them all unselected. If two or more seemed equally close, the subject was allowed to select more than one. We have found that a very simple model corresponds reasonably well with the results of this experiment, and with the behavior of users in further tests of our system. This model is based on a number of highly simplifying assumptions:

1. That the probability of the user selecting a given image depends only on the feature values of the target image and of the other images currently displayed. It is independent of whether any other images are selected.
2. That users base their decisions on only one feature for each image, and that the chosen feature will often correspond to exactly one of the computed features.
3. That the probability of a user choosing a given feature is constant. We denote the probability of each feature as W_f, and refer to it as the feature's weight.
4. That the probability of the user selecting an image is a linear function of the number of other displayed images that are farther from the target in the chosen feature.

These assumptions lead to the conclusion that the probability of a given image being selected should be a linear function of

$$V(D_i, I_t, D) = \sum_{f \in F} W_f \sum_{j \in D} \begin{cases} 1 & \text{if } d(I_t, D_i) < d(I_t, D_j) \\ .5 & \text{if } d(I_t, D_i) = d(I_t, D_j) \\ 0 & \text{if } d(I_t, D_i) > d(I_t, D_j) \end{cases}$$

where F is a set of real-valued functions corresponding to the computed features of the images, and $d(I_t, D_i) = |f(I_t) - f(D_i)|$. We call $V(D_i, I_t, D)$ the "image score" of image D_i.

We tabulated the relative frequency with which images were selected by three different subjects as a function of their image scores computed in this way, using the 18 features described in Table 1, and $W_f = 1$ and found a clear correlation between image scores and frequency of selection. The primary difference between the users is the frequency with which they selected no images at all. If we ignore all the trials in which no images were selected, then the performance of the three subjects is strikingly similar, as illustrated in Figure 2. For comparison, the dotted line shows the performance of a purely random "user". This suggests that subjects first decided whether or not any image should be selected at all, and that different subjects had different thresholds for this judgment. But, once the decision to select an image was made, the rest of the procedure is well modeled by Equation 1.

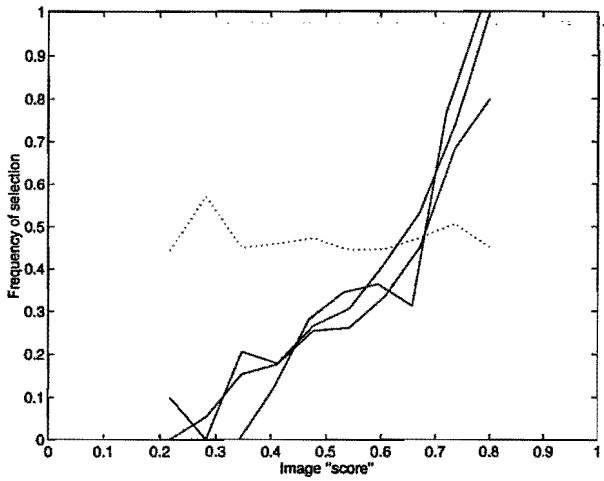

Figure 2. Frequency of image selection versus "image score" for 3 individuals when trials in which no image is selected are ignored. Dotted line shows the performance for a purely "random" user.

Using "image scores" and conceding that the model does not apply when no images are selected, we compute our S function. When the user selects no images, this function returns a constant value, independent of its inputs, which will leave the probability distribution of the database unchanged. When the user selects one or more images, the procedure finds the probability of the user selecting each image according to its image score, and returns the product of the probabilities of all selected images, times the product of one minus the probabilities of all unselected images.

According to our simplifying assumptions, the probability of an image being selected should be a linear function of its score. However, such a function would entail too much trust in the model. If we were to use a linear function of image score, then extreme image scores would lead to probabilities that would have to be clipped at 0 or 1. But users may make mistakes, in which case extreme probabilities could cause the correct target to be completely eliminated from the search. To avoid this, we fit a sigmoid to our experimental data, which is nearly linear in the range for which we have data, but never quite reaches extreme probabilities. The parameters of the sigmoid, M and σ, were set by hand to yield a reasonable fit to the data.

The only remaining task is to choose values for the 18 feature weights, W_f, such that they fit experimental data reasonably well. To gather more data for this step, we began by implementing a version of `PicHunter` in which all weights were set to unity. We then performed a number of searches, and recorded the user's actions. The performance of this version of `PicHunter` was slightly worse than the performance of the tuned version, reported in Section 5. After verifying that users' behavior during these searches matched the behavior shown in Figure 2, we combined this data with the data collected in our initial experiment.

Next, we found the correlation coefficients between the recorded selections and image scores computed with each of our 18 features alone. The features were then sorted in decreasing order of correlation, and the weight for the feature with the highest correlation was set to 1. The weights for the other features were initialized to 0. We then found a weight for each successive feature, in decreasing order of correlation, by trying 100 weights between the values of 0 and 2, and choosing the one which gave the overall best correlation with our data. The weights were then normalized to sum to 1. The resulting weights are shown in Table 2.

W_1	0.0223	W_7	0.0625	W_{13}	0.0112
W_2	0.1362	W_8	0.0201	W_{14}	0.0893
W_3	0.0469	W_9	0.0603	W_{15}	0.0826
W_4	0.0290	W_{10}	0.1116	W_{16}	0.0491
W_5	0.0290	W_{11}	0.0647	W_{17}	0.0134
W_6	0.0848	W_{12}	0.0335	W_{18}	0.0536

Table 2. Feature weights

5 Experimental Results

In this section, we present the results of preliminary target tests of `PicHunter`, and discuss some subjective aspects

of PicHunter's behavior. However, we first discuss our experimental method.

5.1 The Target Testing Experimental Paradigm

Research in content-based image-retrieval has been hampered by the lack of a quantitative metric with which to compare approaches. Researchers have usually provided only qualitative evidence of the utility of their systems based on example responses to user queries. For example, a sketch-based query might be shown to yield images that intuitively appear to match the sketch. We believe qualitative evaluations hinder progress in image database retrieval because (1) they provide no reliable method for comparing different systems and (2) they lead to no methodology for optimizing database design.

One test of an image-retrieval system's effectiveness may be the average (over many trials) of the difficulty a user has in finding individual, randomly selected, target images in a large database. Several variations of this theme are possible.[2] The difficulty the user has finding a target image can be measured in a variety of ways including the amount of time or number of user interactions required (i.e. mouse clicks, iterations of the search, keystrokes, etc.).

Good performance in this type of target testing would clearly indicate that a system is suited for target searching applications. Whether it also indicates how the system will perform in the less directed applications of category searching and open-ended browsing remains to be experimentally verified. But it seems reasonable to think that it does, since the three problems are obviously related.

An important property of target testing is that an image retrieval system which fails to retrieve relevant images will perform the test poorly. This is an important point since often systems are assessed only on the basis of the fraction of retrieved images that are relevant. Such a measure ignores the images which *should* have been retrieved, but were not, i.e. the false negatives. In our view, relevant but unretrieved images are probably a more serious problem than retrieved but irrelevant images (false positives). This is because false positives are easily filtered by the user, albeit at the cost of time, but false negatives are never recovered and these very images may determine the success of a search.

In the experiments reported here, the target image is displayed continuously on the computer monitor. Searching for remembered targets would be a more realistic, but it involves additional complications, such as users finding images which are *almost* identical to the target, users forgetting the target altogether, etc. We believe, however, that results obtained with this simpler test are applicable to more realistic tests.

Since our system is queryless, each search consists of a series of displays and user actions. A natural measure of performance is the average number of such display/action iterations required to find the target image. In the future, comparison may also be based on expended time.

5.2 Target tests

To test the system, we purchased a commercial database of over 20,000 stock photographs on 201 CD's[1]. 200 of the CD's contain 100 images each, grouped together by unifying themes, such as images of models, images of aircraft, etc. The remaining CD is a collection of 122 dissimilar images. We transferred the images from this CD, plus those from 44 thematic CD's, to a hard disk at a resolution of 128x192 pixels, and 24-bits of color. This gave us a total database of 4522 images, for which the 18 features of Table 1 were precomputed.

PicHunter was tested on this database by two individuals, who conducted 20 and 24 searches, respectively. The first subject was one of the system's authors (MLM), and his results represent the performance that can be achieved by a user with maximal expertise – that is: a user whose behavior is likely to match our S function as closely as can be expected. The second subject was a systems programmer who was uninvolved in the development of PicHunter. His results represent the performance of a computer expert who has no special expertise in the use of PicHunter.

In addition, PicHunter was briefly tested by eight secretaries, who performed just one or two searches each. These results represent the performance of people with normal computer literacy and no knowledge of PicHunter whatsoever.

We have tried to minimize the amount of instruction given to all the testers, presenting them with only the following instructions:

1. Select the image that's most similar to the target by clicking on it. A selected image has a red border.
2. If you can't decide between two or more images, you can select more than one.
3. If none of the images looks at all close to the target, or if all of them look equally close, don't select any of them.
4. If you change your mind about which image(s) is (are) closest, you can deselect images by clicking on them again.
5. Once you've selected the best image(s), click on the "GO" button to get the next set of images.
6. When the target image appears, select it and then hit the "FOUND" button.

Of course, these instructions are highly uninformative, and any real system should include hints about how best to work with it. But the less we need to depend on detailed instructions, the more widely applicable our system will be.

The results of all the tests are shown in Table 3. Figure 3 graphs the results for the two computer experts, showing

[2]For example, each target image can be displayed on the computer monitor beside the image-retrieval interface for the duration of each search. Or each target image might only be displayed for a short time interval before each search begins and then the user must search for the remembered image. Or target images may be distorted, e.g. a hardcopy with color distortion, from the true images present in the database.

	Expert User	Computer Expert	Computer Literates
Number of Searches	20	24	11
Number of targets found	20	19	6
Median search length	22	65	234
Mean successful search length	53	57	75
Standard dev. of successful searches	78	40	86

Table 3. Test results for various subjects.

the percentage of searches that were completed successfully within a given number of iterations. The dotted line indicates the corresponding percentages that would be expected if the database were searched at random.

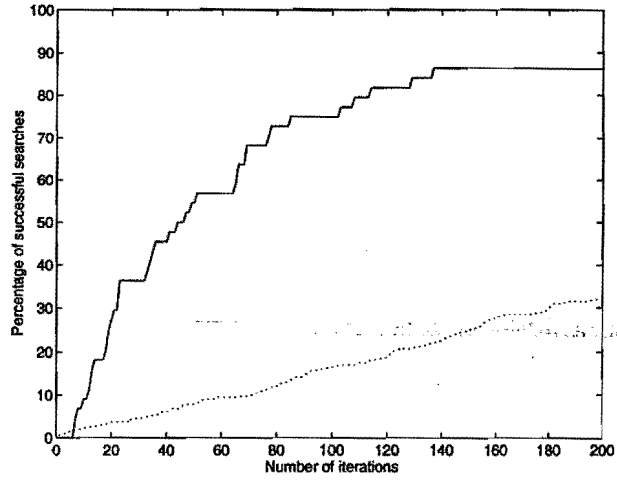

Figure 3. Percentage of successful searches as a function of search length. Dotted curve shows expected result if database is searched at random.

The results from the two computer experts clearly show that PicHunter makes a substantial improvement over random search, in spite of the simplicity of the user interface and precomputed image features. On average, images were successfully found after just over 55 iterations, or 220 images viewed out of 4522. We consider these results to be encouraging in view of the difficulty of this test for a relevance feedback system without explicit queries. In a full system the PicHunter relevance feedback algorithm would be used together with an explicit query and would not have to resolve such a large amount of ambiguity.

The results from secretaries were not as good. Only half of their searches were successful, and the mean length of a successful search was 75 iterations. While this is still significantly better than random chance (assuming that users tend to give up after 150 iterations or so), the user model should be improved. One possible reason for the difficulty is that the secretaries often focused on features which PicHunter does not measure. For example, one secretary, who was searching for a picture of a flamingo, reported that she was basing her selections on whether images contained curves matching the curve of the flamingo's neck. Given that our feature set contains no measure of curvature whatsoever, it is not surprising that the image was not found. We believe that a better, more comprehensive feature set will substantially improve the performance for novice users.

5.3 Subjective behavior

The above tests give us a quantitative measure of performance which we can use to compare PicHunter against other relevance-feedback algorithms and to judge our progress as PicHunter is improved, but it is also interesting to discuss the qualitative performance of the system.

PicHunter exhibits a remarkable ability to find images that resemble one another in apparently semantic ways. Figure 4 illustrates an example. In the first iteration shown,

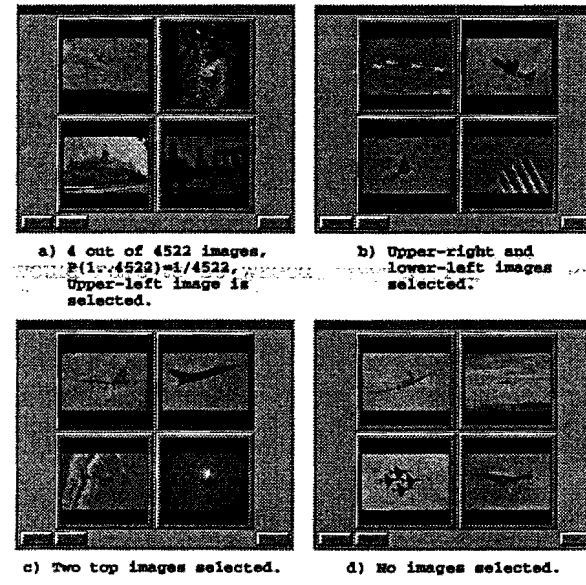

Figure 4. Four iterations of a search

the probability distribution over the database is flat, and a single image of an aircraft in the sky is selected. This is followed by three iterations in which only images of aircraft in the sky are displayed. The database also contains images of eagles in the sky, which have very similar visual characteristics, but PicHunter does not display any of them for some time. Similar behavior occurs, though not as quickly, with more complex categories like images of faces or images of crowds.

It is often difficult to understand how the trivial image features we computed can be used to make the discriminations that the system appears to be making. There are a

number of possible explanations.

1. The features may simply be more powerful than we expected. Certainly, color histograms, which are similar to our color features, have been shown to be effective in discriminating semantic content of images [14, 15], and this might well explain Figure 4. But there are other cases in which the images have wildly different color schemes, yet still seem to be sorted by the correct semantics.

2. We are probably impressed by much lower success rates than would be considered satisfactory in other applications. For example, when looking for faces, we feel the program is succeeding if we get at least one face per iteration. But this is a success rate of only 25%, which would be unacceptable in an application like automatic face detection.

3. It may be that the system is capable of producing probability distributions that reflect extremely complex functions of the features used, and that these complex functions are sufficient to make a trivial feature set more powerful than would be expected.

However intriguing this "semantic" behavior of PicHunter is, the phenomenon illustrated in figure 4 is actually a *problem* with the system. If the target is a picture of an eagle in the sky, the user would probably make the selections shown in this figure. But quite a number of pictures of aircraft would have to be waded through before an eagle is found. The figure, then, shows the system overlearning. This is an important fact to note: the very phenomenon that makes the system appear subjectively successful turns out to be a *problem* when the system is judged by objective target testing.

A related phenomenon is a disconcerting tendency to "hop" from one semantic category to another. If the target is an aircraft, we might quickly find a number of similar images. But, before we reach the particular aircraft image that we are looking for, we often find ourselves suddenly presented with a collection of images of sailboats, or of blue-green landscapes. These incorrect images will be shown for a few iterations before the system returns to aircraft. What appears to be happening here is that the probability distribution which has developed is multimodal. There are many aircraft images and many landscape images that are highly probable. The most probable 4 or 8 or 12 images might easily all come from one group, and the next 4 or 8 or 12 from the other, so the system will hop between these two modes.

A better user model and feature set will probably help reduce the number of modes that develop, thus reducing this disconcerting behavior. And a better algorithm for choosing the set of images to display should reduce the damage done by multiple modes.

6 Future Work

In each portion of PicHunter's design, we intentionally chose the simplest of many possibilities, so that our experimental results would be more easily interpretable. Our current work explores some of the more complex alternatives and extends the ideas to other search tasks. Some of the design choices are discussed below.

6.1 More complex user feedback

The current user interface was chosen to be as simple as possible. This interface is immediately understandable by a user. On the other hand, it provides the search engine with rather impoverished information about the user's desires. We are examining more complex interfaces which allow the user to give the engine more precise information. For each proposed addition we must weigh any improvement in search performance against the increased complexity for the user.

The simplest extension to the current interface would allow the user to indicate the *strength* with which a displayed image matches the goal. For example, each displayed image could have several associated buttons representing different match strengths.

Next are mechanisms that allow the user to indicate the *features* on which a match occurs. For example, each image may have a set of feedback buttons, one for color, one for texture, etc. With these buttons, users can indicate which dimensions caused them to select various images.

A more complex variant of this would allow users to select the particular portions of images which are relevant to them. They could simply click on the relevant image regions or could "lasso" them with the mouse. This allows users to be quite specific while not requiring them to learn a complex query language or drawing tool.

6.2 More complex displays

In the other direction, it would be helpful to users to gain more information about the search engine's current "beliefs". This would allow them to better understand the affect of their choices. At the simplest level, the displayed images are presented in the order of estimated probability. Alternatively, the engine's current strength of belief for each image could be displayed as a number or bar.

For more detailed information it is useful to know which features caused each image to be displayed. In the simplest variant, each displayed image might have a label such as "texture" or "color". An interesting alternative would be to display a second copy of each image with only the pixels that contributed to the features which are estimated as desirable. For example, an image of an ocean scene might have gained a high probability because the user selected many images which have light blue in their upper regions. In addition to displaying that image, PicHunter would display the same image with everything but the sky masked out.

6.3 Choice of image features

The features which are computed from each image in the current version of PicHunter were chosen by hand and selected for simplicity more than search usefulness. We are currently exploring a system which systematically searches over sets of features for those most predictive of a database of recorded user selections. The feature set is adapted to improve the prediction performance. We hope that this approach will semi-automatically determine a set of features that closely match those used by users in making similarity judgments.

6.4 Choice of user model

We are exploring several avenues for improving the user model. Currently, we adjust the parameters and architectural choices partly by judging performance on a database of user selections. We would like to further automate this process, so that user models can be automatically constructed for new domains by just running a set of training trials.

In addition to learning user models from large databases, we are considering user models which adapt to the user during a single search. If one model's the user as having a dynamically changing hidden "mental set", then the stochastic user model may be described by a hidden Markov model.

We are also studying models which dynamically and automatically adapt measures of feature salience. In this case the user model is a probabilistic mixture of models dependent on subsets of the features. As a search proceeds, the mixture coefficients adapt to select the features that are most predictive of the user's selections. As described above, this salience information could then be provided in specialized displays to the user.

6.5 Image selection algorithm

In the current version of PicHunter, the displayed images are always those estimated to have the highest probability. In any search there is a conflict between exploration and exploitation. It can explore directions which are likely to give the most *information* or directions which are most likely to immediately provide payoff. These alternatives often lead to different choices. As the posterior distribution gets peaked, the most probable images tend to be quite similar to one another. It can happen that the system will learn more by displaying less probable images which are less similar to one another.

We have made preliminary studies of a more intelligent display algorithm that tries to choose sets of images which will yield high expected drops in entropy of the posterior probability distribution over potential targets. Keeping in mind that displaying the actual target causes the entropy to drop to zero, this algorithm automatically makes near-optimal tradeoffs between exploration and exploitation.

6.6 Other media

While we have only describe search in image databases, virtually all of the concepts are relevant to other media. The two most obvious are audio and video databases. A potentially important domain may be databases of 3 dimensional object models that should be searched on shape. One can imagine more abstract models for particular domains such as databases of dance movements or sign language gestures. The search paradigm we have described here could be applied to these domains without creating more new and obscure description languages for each domain.

6.7 Category search and open-ended browsing

All of these issues and more are also relevant to category search and open ended browsing. For category search our Bayesian model has to be modified to provide probability distributions over *sets* of images rather than individual images. An appropriate prior for such a model would include a preference for sets with semantic coherence. Such a prior might be learned from training sets of semantic similarity judgments.

Open-ended browsing involves a search in which the user's goal may change during the course of the search. Thus the system's user model should exhibit gradual "forgetting" in that more recent information should dominate older information. We are considering search engines in which the Bayesian model includes a Markovian temporal component. This naturally gives rise to an exponentially decaying contribution from the past when the user's goal changes.

7 Conclusion

PicHunter is based on a general Bayesian framework for using relevance feedback to direct a search. It maintains a posterior probability distribution that each image in the database is the target. This distribution is used to select the next images to display. The user's selections are then incorporated into the posterior distribution via a probabilistic user model. We presented a methodology for building the user model from a database of user selections. A distinguishing characteristic of this framework is that, while other relevance feedback methods refine the query, our approach refines the answer.

The initial implementation uses the simplest versions of each of these components and yet provides significant search speedup. We introduced "target testing" as a quantitative, theory-independent measure for evaluating the performance of multimedia retrieval systems. Target testing is simple and is applicable to a variety of databases and retrieval systems. We expect that target testing will also provide useful information on the system's performance for more complex queries.

We have described a number of extensions of the system in the same framework which we believe will lead to better and more practical retrieval performance. Of course, the PicHunter relevance feedback algorithm can easily be used in conjuction with any other retrieval method.

Acknowledgments

We thank Y. Hara and K. Hirata of NEC Central Laboratories for stimulating our interest in this area, and making their system and image database available to us. We also thank Talal Shamoon and Harold Stone for helpful discussions. We are especially grateful to the

growing list of test subjects who continue to play a crucial role in the refinement of our approach.

References

[1] Corel stock photo library. Corel Corp., Ontario, Canada.

[2] J. Barros, J. French, W. Martin, P. Kelly, and J. White. Indexing multispectral images for content-based retrieval. In *Proceedings of the 23rd AIPR Workshop on Image and Information Systems, Washington DC, Oct.*, 1994.

[3] A. D. Bimbo, P. Pala, and S. Santini. Visual image retrieval by elastic deformation of object sketches. In *Proceedings IEEE Symposium on Visual Languages*, pages 216–23, 1994.

[4] 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.

[5] M. Hirakawa and E. Jungert. An image database system facilitating icon-driven spatial information definition and retrieval. In *Proceedings 1991 IEEE Workshop on Visual Languages*, pages 192–8, 1991.

[6] K. Hirata and T. Kato. Query by visual example; content based image retrieval. In *Advances in Database Technology–EDBT '92 Sprinter-Verlag Berlin Heidelberg in Pirotte, A., Delobel, C., and Gottlob, G. (Eds.)*, 1992.

[7] T. Kato, T. Kurita, H. Shimogaki, T. Mizutori, and K. Fujimura. Cognitive view mechanism for multimedia database system. In *IMS '91 Proceedings. First International Workshop on Interoperability in Multidatabase Systems*, pages 179–86, 1991.

[8] P. Kelly and T. Cannon. Candid: Comparison algorithm for navigating digital image databases. In *Proceedings Seventh International Working Conference on Scientific and Statistical Database Management*, pages 252–8, 1994.

[9] T. Kurita and T. Kato. Learning of personal visual impressions for image database systems. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 547–52, 1993.

[10] T. P. Minka and R. W. Picard. Interactive learning using a "society of models". Technical Report 349, MIT Media Lab, 1995.

[11] M. Oda. Context dependency effect in the formation of image concepts and its application. In *Conference Proceedings 1991 IEEE International Conference on Systems, Man, and Cybernetics. Decision Aiding for Complex Systems*, volume 3, pages 1673–8, 1991.

[12] A. Pentland, R. W. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. In *SPIE Storage and Retrieval Image and Video Databases II*, volume 2185, 1994.

[13] H. S. Stone and C.-S. Li. Image matching by means of intensity and texture matching in the Fourier domain. In *Proc. SPIE Conf. in Image and Video Databases*, 1996.

[14] M. Stricker and M. Swain. The capacity of color histogram indexing. In *Proceedings 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 704–8, 1994.

[15] M. Swain and D. Ballard. Indexing via color histograms. In *Proceedings Third International Conference on Computer Vision*, pages 390–3, 1990.

[16] H. R. Turtle and W. B. Croft. A comparison of text retrieval models. *The Computer Journal*, 35(3):279–290, 1992.

[17] V. V.N. Gudivada and Raghavan. Content-based image retrieval systems. *IEEE Computer September*, 28(9):18–22, 1995.

[18] G. Yihong, Z. Hongjiang, and C. Chuan. An image database system with fast image indexing capability based on color histograms. In *Proceedings of 1994 IEEE Region 10's Ninth Annual International Conference. Theme: Frontiers of Computer Technology*, volume 1, pages 407–11, 1994.