Interactive Learning with a "Society of Models" *

Thomas P. Minka and Rosalind W. Picard
Vision and Modeling Group
MIT Media Laboratory
20 Ames Street; Cambridge, MA 02139
{tpminka, picard}@media.mit.edu

Abstract

Digital library access is driven by features, but the relevance of a feature for a query is not always obvious. This paper describes an approach for integrating a large number of context-dependent features into a semi-automated tool. Instead of requiring universal similarity measures or manual selection of relevant features, the approach provides a learning algorithm for selecting and combining groupings of the data, where groupings can be induced by highly specialized features. The selection process is guided by positive and negative examples from the user. The inherent combinatorics of using multiple features is reduced by a multistage grouping generation, weighting, and collection process. The stages closest to the user are trained fastest and slowly propagate their adaptations back to earlier stages. The weighting stage adapts the collection stage's search space across uses, so that, in later interactions, good groupings are found given few examples from the user.

1 Issues for digital libraries

One important issue for digital libraries is finding good models and similarity measures for comparing database entries (Figure 1). A part of this difficulty is that feature extraction and comparison methods are highly data-dependent. Similarity measures are also user and task dependent, as demonstrated by Figure 2. Unfortunately, these dependencies are not, at this point, understood well enough, especially by the typical digital library user, to permit careful selection of the optimal measure beforehand.

Next, the scope of queries that databases need to address is immense. Current computational solutions attempt to offer location of perceptual content ("find round, red objects") and objective content ("find pictures of people in Boston"). Desirable queries also extend to subjective content ("give me a scene of a romantic forest"), task-specific content ("I need something with open space, to place text"), and collaborative content ("show me pictures children like"). Answering such queries requires a variety of features, or metadata, to be attached to the data in a digital library, some of which may not be computable directly from the data. The implication for algorithms is that they cannot rely on one model or one small set of carefully-picked features but will have to drink from a veritable "feature hydrant" from which only a few drops may be relevant for the query.

Some recent systems which perform retrieval on visual data are QBIC [9], SWIM [10], Photobook [11], and CORE [12]. A notable quality of these systems is that they present many different ways of organizing the data but offer little assistance in actually choosing one of these organizations or making a new one. Users are often forced to determine what features and feature combinations will be relevant to their intent, if any, instead of addressing their intent directly. Since intentions can vary greatly and features can be very opaque, another solution is needed. The example-based interaction in FourEyes, coupled with a learning element that selects and constructs organizations, provides such an alternative.

2 Multiple models

Dealing with these issues requires the use of multiple features, computed from the data or not, as well as ways to make informed, automatic selection of models and the features they describe. At this point in time, there seems to be no lack of specialized models, just a lack of knowing the best ways of utilizing them. Two well-known multiple model approaches are Bayesian combination [1] and the rule-based blackboard [2], but this paper advocates a different approach which is more desirable for the interactive digital library setting.

The approach described in this paper allows many different models to be easily incorporated without the computational complexity that usually plagues multi-model methods. The idea is to precompute many plausible groupings of the data, where groupings are induced by different models. Then the system selects and combines the groupings, based on positive and negative examples from the user. Relevance information, viz. which groupings were most useful, can then be fed back to modify these groupings or influence future grouping generation. In this way, the system is not only trained during individual example-based sessions with a user, but also trained across sessions to suit the tasks which it is asked to perform. This makes sure that the search space of groupings is always small but still contains desirable solutions.

An important optimization comes from the observation that when a reasonably large number of groupings

*Supported in part by BT, Interval, HP Labs, and NEC.

Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)
1063-6919/96 $10.00 © 1996 IEEE

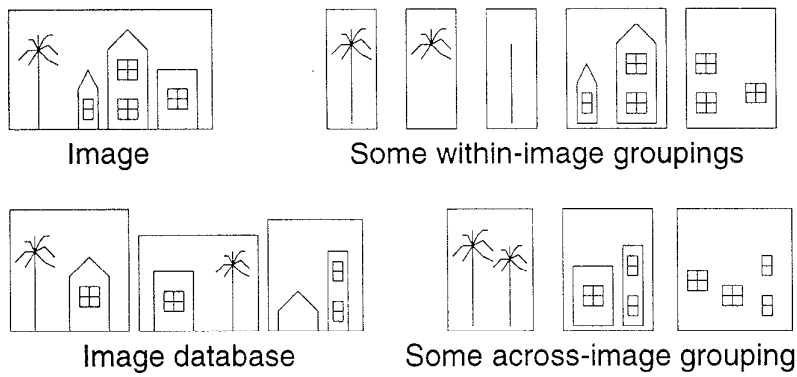

Figure 1: A basic task for image retrieval, segmentation, and annotation tools, which is addressed in this paper: recovering useful within-image or across-image groupings. A grouping is just a set of related regions. Note that useful groupings generally cannot be captured by a single model, or even a single partition or hierarchy, and the similarity measure required to induce these groupings may be quite complex.

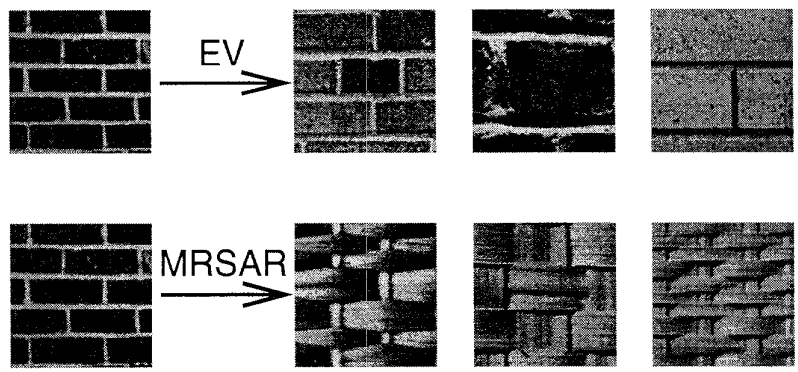

Figure 2: Task-dependent performance of texture models. The three patterns on the right are ordered by their similarity to the pattern on the left, given the particular model space EV or MRSAR. Both results capture the horizontal/vertical structure, but the EV returns a more semantically pleasing result since all images are bricks. However, these bricks are at different scales, and have different microtexture. Depending on the user's task, e.g. "find other images that look like bricks," the MRSAR result, or that of another model, may be preferable.

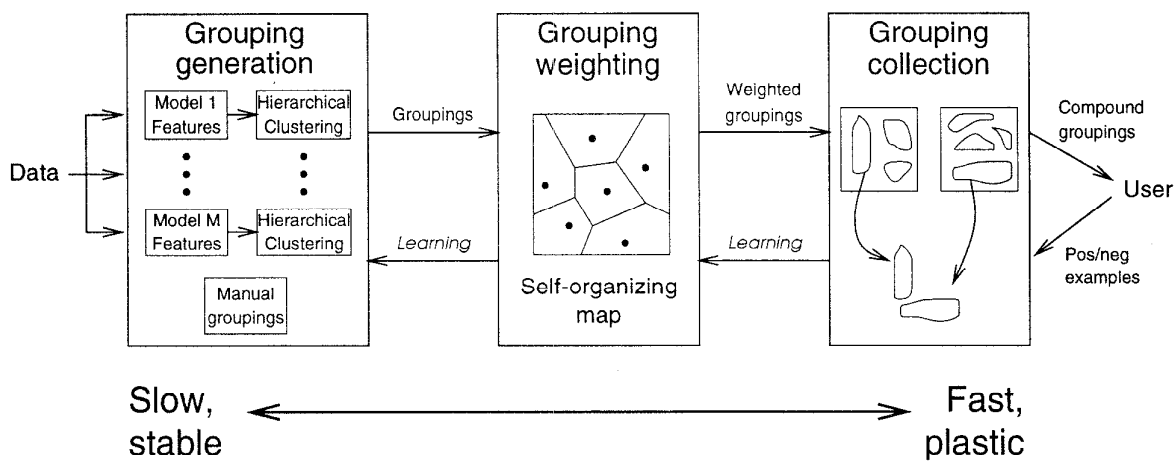

Figure 3: Interactive pattern recognition framework. The arrow at the bottom describes the rate at which the three stages learn.

Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)
1063-6919/96 $10.00 © 1996 IEEE

is available, the correct groupings are usually present but are hard for the system to identify, given only a few training examples from the user. Therefore, the system can significantly improve itself just by changing the *relative weights* of groupings, not the groupings themselves. This optimization is realized by placing a separate weighting stage in between the generation and collection stages. Weighting does not change the size of the search space, but it does change the shape, by putting the "better" groupings first.

The resulting three-stage method, illustrated in Figure 3, differs from conventional feature extraction and classification in three crucial ways. First, the feedback arc between the classifier and the features is performed by the computer, not the designer. Second, each stage develops at different times and different rates, with the stages closest to the user changing fastest. This allows the computations to be distributed in time and space, facilitating interactive use and the incorporation of more complex models. This differs from Bayesian combination which essentially executes and adapts all stages at once, restricting the Bayesian approach to simple models for acceptable speed. Third, training is accumulated across sessions with the user, so that the system improves over time and can solve similar problems better, i.e. learn faster, the next time.

This paper describes an interactive-time learning system, called "FourEyes," which assists a user in finding groupings both within and across images based on features from a society of models. The current implementation obtains groupings for still images from color models, texture models, and the user. For images from a sequence, optical flow groupings are also used. The grouping representation used by FourEyes allows for a variety of arbitrary models, and could easily be extended to include audio, text, or other data. However, the focus in this paper is on visual data.

3 Generating groupings

A grouping is a set of image regions ("patches") which are associated in some way. The elements of a grouping may not necessarily come from the same image. This representation is useful since it admits different kinds of associations without adding complexity. For example, one set may represent "regions containing between 15% and 25% blue pixels" while another may represent "regions containing waterfalls" while yet another may represent "regions which were browsed very often this week." It also allows specific associations between patches to be weighted independently, since each set may have its own weight. This is important because, for example, lettering may be best grouped by shape whereas sky may be best grouped by brightness and location in the image. The notion of grouping, including the generation methods to be described, apply to all kinds of data, not just images.

FourEyes computes within-image groupings from a model feature, such as color or texture, in three steps as illustrated in Figure 4. This is the first stage of Figure 3. The result is a hierarchy of image regions for each image, for each model. Then it computes across-image groupings from a hierarchical clustering

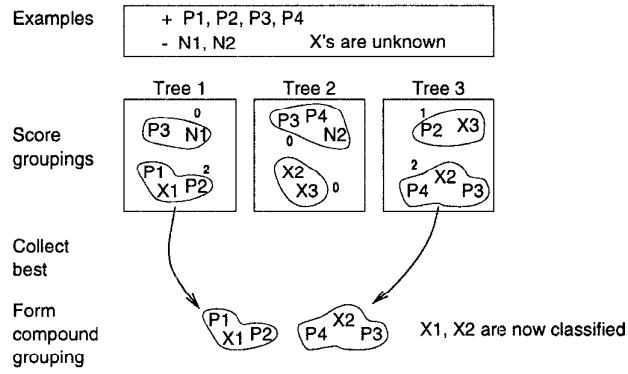

Figure 5: Collecting groupings

of a feature measured over the within-image groupings. The within-image groupings need not have been generated by the same feature used for across-image grouping; they may have come from optical flow or even manual outlining. For example, if a within-image grouping utilizes face detection to produce segments containing faces, the across-image grouping can use a face classifier. The clustering algorithm is described in [4], a longer version of this paper. All of these within-image and across-image groupings are computed off-line, before the user begins interaction with the system. On-line modification of groupings, included in the FourEyes model as the link from the second stage to the first, will be described in a future publication.

4 Collecting groupings

Once a set of groupings has been formed, the next task is to select or combine these to form compound groupings for the user. This is the third stage of Figure 3, referred to below as "the learner." At every point in the interaction, the learner must try to generalize from a set of examples provided by the user. The result is a set of image regions which contains all of the positive examples, and none of the negative. This set is formed from multiple groupings and so is called a compound grouping.

The learning algorithm used in FourEyes descends from AQ [5]. AQ is a greedy method that collects groupings one at a time, such that each one includes no negative examples but their union includes all positive examples. Starting from an empty union, the grouping which adds the most positive examples but no negative ones is iteratively added. Since the hierarchies generated in the first stage include the smallest-scale patches at the leaves, this algorithm can always satisfy any set of examples, no matter how arbitrary.

The algorithm used in FourEyes differs from AQ in its evaluation of the next grouping to add. Instead of choosing the grouping which simply maximizes the number of positive examples (as in our previous work [6]), it maximizes the product of this number and the *prior weight* of the grouping. This means that, e.g., a grouping with twice the prior weight can cover half as many positive examples before it is chosen. Thus the

Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)
1063-6919/96 $10.00 © 1996 IEEE

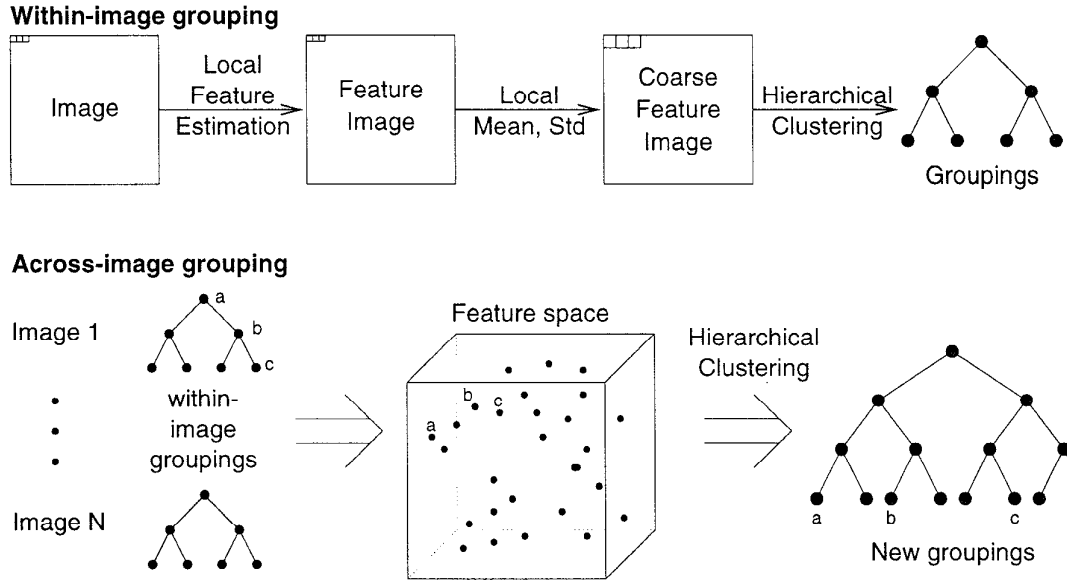

Figure 4: Computing within-image and across-image groupings. In image 1, grouping a contains b, which contains c; e.g. they might be house, door of the house, window on the door. When projected into feature space, they are considered individually, and look different. The resulting clustering says that a looks more like b than c.

prior weights directly influence the learner's inductive bias. The prior weights are determined from statistics collected over multiple learning sessions.

5 Weighting groupings

The learner in the third stage tries to find the best compound grouping according to consistency with the user's examples and an inductive bias. When the number of examples is large, consistency alone can serve to isolate good groupings. However, each example is expensive in terms of the user's time. When the number of examples is small, many groupings will be consistent; consequently, the bias is crucial in determining which groupings are chosen. Thus it is important that the learner have adaptive prior weights which change between interactions with the user, so that the groupings which were satisfactory this time will be selected earlier (i.e. with fewer examples) next time. This is the purpose of the second stage of Figure 3.

5.1 Modeling weight-space

FourEyes adapts to different learning problems by clustering weight-space. Currently this is done via a self-organizing map (SOM) [7]. During user interaction, each SOM unit (stored vector of weights) competes for consistency with the user's examples; the winning unit propagates its weights over the groupings. When the user is satisfied with the output of the learner, the winning unit is updated to more closely match the examples. In this way, the SOM defines a clustering of the weight-vectors for the problems it has seen, where each SOM unit is a cluster center. Note that a self-organizing map is typically used for the classification of feature vectors in a learning problem;

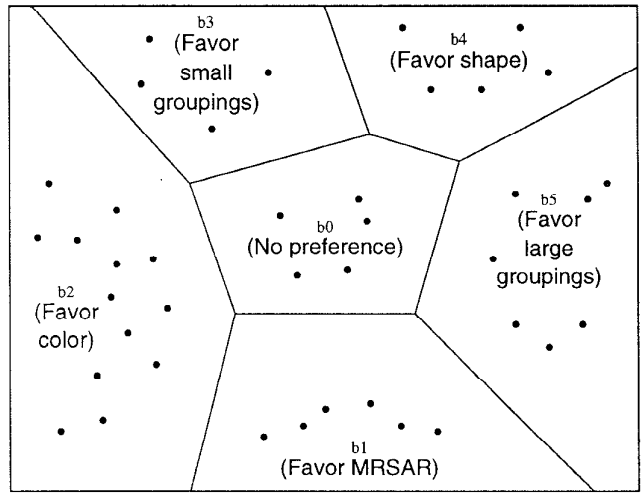

Figure 6: Hypothetical weight-space for learning. Each point is a vector of weights for all of the groupings. The optimal weights for different problem domains will fall into distinguished regions. These regions can be approximated by the Voronoi cells (in bold) of units in a self-organizing map, which clusters all of the points it sees. A unit which "favors color" weights most highly those groupings which come from a particular color model, from a combination of color models, or from non-color models that happen to be consistent with color.

Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)
1063-6919/96 $10.00 © 1996 IEEE

here it is being used for classifying learning problems themselves, in terms of the grouping weights they favor. Each SOM unit then represents a prototypical learning problem.

Each SOM unit stores, for every grouping, the probability of including a positive or negative example. These probabilities are thus the "features" which are used to describe a learning problem. The distance function used to compute the winning unit is roughly the correlation between the probability distribution stored in a unit versus the actual distribution of the user's examples over the groupings (the precise function is described in [4]). When a unit is updated, the user's examples contribute to that unit's probability estimates.

5.2 Learning speedup

The learner described in Section 4 was modified in two ways:

1. After every e examples received, the SOM was consulted for each label to provide a prior weight to be used when selecting groupings for that label. The choice of e is a time/accuracy tradeoff, since SOM lookups are expensive; the experiments used $e = 10$.

2. When the learner was signaled that the learning task was completed, for each label it updated the SOM unit whose prior weight was selected for that label.

Experiments have shown that this self-organizing map approach is effective at learning good weights for several different learning problems simultaneously [4].

6 Performance on natural scenes

The performance of FourEyes was measured by its labeling performance on the natural scenes in the "BT images" (`http://vismod.www.media.mit.edu/vismod/imagery/Bt_photos`). In these images, the regions are of irregular shapes and sizes, and contain many different scales and inhomogeneous textures. Three human subjects were asked to freehand outline regions in 25 of the natural scenes and assign the seven labels "building," "car," "grass," "leaves," "person," "sky," and "water" to them. They were not asked to make precise boundaries or make decisions on a strictly perceptual basis (both of which would have aided FourEyes). Then a majority vote among the subjects was used to derive a single, approved ground-truth segmentation and labeling of those images. Since within-image groupings were computed using a 16×16 coarse feature image (Figure 4), the ground-truth segmentations were quantized to that resolution. Note that finer tessellation-sizes could be used, or overlapping tessellations, or even single pixels, if required by the application.

FourEyes was then used to simulate what the same labeling process would have been like with a computer assistant, where the user incrementally labels 32×32-pixel patches. This corresponds more closely to the database access scenario, where the user makes decisions while browsing the data, than the traditional train/test scenario. Five experiments were conducted with different sets of groupings in stage 1. Patch size varied in these groupings, but the results in Table 1 are given in terms of 32×32-pixel patches only. There were 4546 labeled 32×32-pixel patches and 7 classes so these are the theoretical maximum and minimum numbers of examples required to reach zero error, i.e. perfectly label all patches.

Groupings	Zero error	25% error
8×8	1.6	1.8
plus MRSAR	2.0	3.3
plus Ohta	2.4	5.7
second run	2.9	8.8
plus human	2.6	20
second run	2.9	28
plus ideal	38	47
second run	69	107

Table 1: Annotation savings for natural scenes. Numbers are the ratio between the total number of correctly labeled 32×32-pixel patches (4546 for zero error, 3410 for 25% error) and the number of examples. The higher the ratio, the more help the system is to the user.

The baseline experiment (row 1 in Table 1) used a set of 1600 groupings corresponding to an 8×8 tessellation of each image, i.e. into groups of four 32×32-pixel patches. This corresponds to a simple bias toward giving nearby patches the same label. It required 2924 examples to reach zero error, for an annotation savings of 1.6:1.

Next, the feature set of FourEyes was enlarged to include within-image groupings computed from the MRSAR texture feature over 64×64-pixel patches (1740 groupings, or about 70 per image). This allowed FourEyes to achieve a savings of 2:1. Third, within-image groupings computed from the Euclidean distance between unnormalized histograms of 32×32-pixel patches in the Ohta color space [8] were added (1663 groupings), which raised the savings to 2.4:1.

Further improvement was found when the SOM was updated and run again on the same problem. Performance improved to 2.9:1, which is therefore the most that can be expected with these two models.

The learning curves exhibited diminishing returns after reaching 25% error; the last experiment spent 75% of its examples after this point. This indicates that the system is most effective at getting a quick first-cut labeling rather than a perfect labeling. Interestingly, adding across-image groupings computed from the MRSAR or Ohta histogram features did not improve performance. This indicates that the across-image perceptual variations in this data's semantic classes were high enough to confuse these image models. Another cause might be the scale-sensitivity of these particular across-image features.

The fourth test added human-provided within-image groupings to the first stage of FourEyes. This test would correspond to the system forming new groupings to better match that person's preferences.

Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)
1063-6919/96 $10.00 © 1996 IEEE

The new groupings were provided by one of the sets from which the ground-truth was derived, but deliberately did not match exactly the ground-truth used in our tests. This raised the zero-error annotation savings slightly and allowed the learner to reach 25% error much faster. In the fifth test, the correct within-image groupings were added (an ideal situation) and FourEyes improved by an order of magnitude. This indicates that there is still room for improvement in the image groupings, either by using better models or by learning better groupings over multiple interactions.

7 Summary

The "FourEyes" learning system for assisting users in digital library segmentation, retrieval, and annotation, has been described. Digital library access requires the use of many context-dependent or noisy features, whose relevance is not always obvious. FourEyes addresses this problem on multiple fronts:

1. It first makes tentative organizations of the data, in the form of groupings. The grouping representation provides a common language for different measures of similarity. Groupings can be manually provided, induced by color/texture models, derived from optical flow information, etc. FourEyes uses both within-image groupings and across-image groupings composed of these.

2. The user no longer has to choose features or set feature control knobs. Instead, the user provides positive and negative examples which allow FourEyes to choose groupings (hence, similarity measures) automatically. The interaction is more like a conversation where both parties give each other prompt and relevant feedback in order to resolve ambiguities.

3. With many groupings to choose from, the number of examples required to isolate good groupings can get large. FourEyes circumvents this by having *prior weights* on the groupings and preferring groupings with more weight. These weights are learned across interactions with users, so that the system gets better, i.e. learns faster, from repeated use.

4. Since the optimal weights on groupings change with context, FourEyes employs a self-organizing map to remember useful weight settings. As the user interacts with it, FourEyes chooses the most appropriate weights in the map. This way, FourEyes can improve its joint performance on a wide range of tasks.

5. FourEyes offers interactive performance by explicitly separating these grouping generation, weighting, and collection stages. It does this without sacrificing adaptability or the use of multiple models, because feedback between the stages allows the whole system to learn, though each stage at a different rate.

References

[1] S. C. Zhu, T. S. Lee, and A. L. Yuille, "Region competition: Unifying snakes, region growing, energy/Bayes/MDL for multi-band image segmentation," in *Int. Conf. on Computer Vision*, (Boston, MA), pp. 416–423, 1995.

[2] T. M. Strat and M. A. Fischler, "Context-based vision: Recognizing objects using information from both 2-d and 3-d imagery," *IEEE T. Patt. Analy. and Mach. Intell.*, vol. 13, pp. 1050–1065, Oct. 1991.

[3] R. A. Jarvis and E. A. Patrick, "Clustering using a similarity measure based on shared near neighbors," *IEEE T. Comp.*, pp. 1025–1034, Nov. 1973.

[4] T. P. Minka and R. W. Picard, "Interactive learning using a 'society of models'," *Submitted for Publication*, 1995. Also appears as MIT Media Lab Perceptual Computing TR#349.

[5] R. S. Michalski, "A theory and methodology of inductive learning," *Artificial Intelligence*, vol. 20, no. 2, pp. 111–161, 1983.

[6] R. W. Picard and T. P. Minka, "Vision texture for annotation," *Journal of Multimedia Systems*, vol. 3, pp. 3–14, 1995.

[7] T. Kohonen, *Self-Organization and Associative Memory*. Berlin, Heidelberg: Springer, 1984. 3rd ed. 1989.

[8] Y.-I. Ohta, T. Kanade, and T. Sakai, "Color information for region segmentation," *Comp. Graph. and Img. Proc.*, vol. 13, pp. 222–241, 1980.

[9] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. 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* (W. Niblack, ed.), (San Jose, CA), pp. 173–181, SPIE, Feb. 1993.

[10] H.-J. Zhang and S. W. Smoliar, "Developing power tools for video indexing and retrieval," in *Proceedings SPIE Storage and Retrieval for Image and Video Databases II* (W. Niblack and R. C. Jain, eds.), (San Jose, CA), pp. 140–149, SPIE, Feb. 1994. Vol. 2185.

[11] A. Pentland, R. W. Picard, and S. Sclaroff, "Photobook: Tools for content-based manipulation of image databases," in *SPIE Storage and Retrieval of Image & Video Databases II*, (San Jose, CA), pp. 34–47, Feb. 1994.

[12] J. K. Wu, A. D. Narasimhalu, B. M. Mehtre, C. P. Lam, and Y. J. Gao, "CORE: a content-based retrieval engine for multimedia information systems," *Multimedia Systems*, vol. 2, pp. 25–41, Feb. 1995.

Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)
1063-6919/96 $10.00 © 1996 IEEE