M.I.T Media Laboratory Perceptual Computing Section Technical Report No. 302
Multimedia Systems: Special Issue on Content-based Retrieval

Vision Texture for Annotation*

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

Abstract

This paper demonstrates a new application of computer vision to digital libraries – the use of texture for *annotation*, the description of content. Vision-based annotation assists the user in attaching descriptions to large sets of images and video. If a user labels a piece of an image as "water," a texture model can be used to propagate this label to other "visually similar" regions. However, a serious problem is that no single model has been found to be good enough to reliably match human perception of similarity in pictures. Rather than using one model, the system described here knows several texture models, and is equipped with the ability to choose the one which "best explains" the regions selected by the user for annotating. If none of these models suffices, then it creates new explanations by combining models. Examples are given of annotations propagated by the system on natural scenes. The system provides an average gain of four to one in label prediction over a set of 98 images.

1 Introduction

Digital libraries of text, sound, image, video, and other data are rapidly growing in size and availability. The content is diverse – news footage, medical imagery, art collections, whale flukes, consumer goods, weather photos, Disney movies, fabric samples, home video, vacation photos, and more. However, tools for accessing digital content are still in their infancy. This paper describes a new step toward improving content access – the use of computer vision to help generate descriptions for *annotating* image and video.

Traditionally, access to multimedia libraries has been in the form of text annotation – titles, authors, captions, and descriptive labels. Text provides a natural means of summarizing massive quantities of information. Text keywords consume little space and provide fast access into large amounts of data. When the data itself is text, it can be summarized using sophisticated computer programs based on natural language processing and artificial intelligence.

When the data is not text, but rather sound, image, or video, then generating labels is considerably more difficult.

Figure 1: Two scenes labeled by humans as "city" or "country." These labels can also be generated by the computer through the use of "vision texture," scene description based on collective texture features.

The labeling process requires some unknown transformation to first convert the signals to symbols. How humans accomplish this transformation so effortlessly remains a mystery. To date, when a database needs annotation, a person has to enter the labels by hand at great cost and tedium.

Of course, some access to pictures, sound, and video will be best achieved without text – for example, the user may wish to "find another shot like this" or to "skip to the end of the song." In these cases, a signal is compared to another signal – conversion to text is not required. Solving requests like these is the difficult quest of new research in pattern recognition, computer vision and acoustic scene analysis. Progress is rapidly being made in the ability to search for similar scenes based on shape, color, and texture. Examples are the QBIC system of IBM [1], SWIM system of ISS [2], and Photobook system of MIT [3] [4].

Even "purely visual" databases (e.g. a collection of paintings) will still tend to have text associated with them (title, artist, subject, etc.). Annotation is both important for preparing multimedia digital libraries for query and retrieval, and useful for adding personal notes to the user's online collection. But annotation is costly in time and dollars to perform; tools that save work annotating are greatly needed.

Tools that help annotate multimedia databases need to be able to "see" and "hear" as the human sees and hears – so that if the human says "label this stuff grass" the computer will not label just that stuff, but also go find other grass that "looks the same" and label it too.[1] Tex-

*This work was supported in part by BT, PLC, Hewlett-Packard Labs, and NEC.

[1] Eventually, we expect the computer will infer most of these labels without being told; the present system collects

ture features, although low-level, play an important role in the high-level perception of visual scenes – enabling the distinction of regions like "water" from "grass" or "buildings" from "sky." Such features alone will not solve the complete annotation problem, but they will be a key component of the solution.

Ideally, computers will be able to intelligently combine vision, acoustic scene analysis, and text annotation to provide users access to content. This paper describes a first step in the combination of vision and text annotation – specifically, the use of "vision texture" to assist in annotation.

2 Vision texture and annotation

Computer vision has traditionally focused on the "things" in pictures described by their specific geometry, shape, or structure, e.g. people and objects such as cars, houses, or machine parts. However, pictures also contain grass, water, leaves, sky, crowds of people, and other "stuff" that does not depend on specific shape or structure, but is best described by collective properties. There is not a sharp boundary between the categories "things" and "stuff," just like there is not a sharp boundary between count nouns and mass nouns;[2] nevertheless, the distinction prevails as two endpoints of a useful continuum.

This research focuses on the use of collective visual properties, or "vision texture" for annotation. Texture models extract "stuff" features such as directionality, periodicity, randomness, roughness, regularity, coarseness, color distribution, contrast, and complexity, which are hypothesized to be important for human perception and attention [5] [6] [7]. Low-level color texture features [8] have also been shown to be useful for "selection" of regions of interest preceding actual recognition.

Low-level texture features may also play a significant role in high-level tasks such as recognition. Studies with pigeons [9] indicate they can recognize categories such as "water" and "tree;" such studies support the hypothesis for a simple[3] mechanism, that perhaps looks at collective low-level features for making comparatively high-level quick classifications. A recent study [10] demonstrated that features based on texture orientation closely matched human high-level classifications on 91 out of 98 digitized vacation photos (two of which are shown in Fig. 1). Simple color histograms have also been shown to be useful in the recognition task of matching known models to unknown image data [11]. The belief that low-level vision is limited to only low-level tasks is too restrictive.

2.1 Annotation by fickle users

Scene annotation is an ill-posed problem in general. Consider the two pictures from Boston in Fig. 2, where one picture is of rows of pews and one is of a stained-glass window. Both of these frames exist in one of our travel video databases, where the annotator labeled them "church." Simple visual features do not reliably predict this annotation. However, by having a user identify and annotate parts of a picture, e.g. encircle the pews and label them

Figure 2: Two keyframes labeled "church" from a travel video of Boston

"pews," then features can eventually be identified that are consistent with the labelings. Moreover, these feature-label pairings can be combined with AI systems based on text,[4] to begin to learn that pew features might also be likely to predict the label "church." Gradually, the associations between image features and descriptions can be refined until the system begins to recognize annotations automatically and reliably.

No algorithms have yet succeeded in mimicking human description of scenes, except for in a few highly-constrained situations. The solution proposed here is not to try to solve the whole (unconstrained) annotation problem automatically, but to develop tools that assist the human interactively, and that learn the user's preferences.[5] The system proposed here provides an interaction where the user labels a region and the system determines "similar" regions in which to propagate the user's label.

The problem of finding visually similar regions to which to propagate the label is very close to the problem that arises in retrieval of similar pictures [1] [14]. In both cases, features need to be identified which are important for measuring similarity.

Determining which model gives the best features for measuring similarity is complicated by the fact that users are fickle. People are nonlinear time-varying systems when it comes to predicting their behavior. A user may label the same scene differently, and expect different regions to be recognized as similar when his or her goals change.

Many factors influence the way people measure similarity, making human similarity decisions very difficult to predict. In Fig. 3 for example, some would say the top two images are most similar in pattern and scale. Others would say the two brick images are most similar because they are both brick. In fact, measuring similarity is significantly more complicated than these examples reveal, encompassing arguments that feature spaces for measuring similarity may be non-metric [16] and non-symmetric [17]. We briefly identify the following influences:

Visual features Regions may look similar at a quick

feature-label pairings toward this goal.

[2] How many lions in a pride, or how many jellyfish in a smack?

[3] "Simple" enough for a bird brain.

[4] WordNet [12] for example, contains synonyms and semantic relations of English words and concepts. It can be used to retrieve "semantically close" words, e.g. "operating room" retrieves "hospital" with the semantic relation "part-of" [13].

[5] After interacting with many users, however, the system might begin to recognize some "universal" preferences and adopt those for automatic annotation.

glance; you could swap one for the other and probably not notice, e.g. dense leafy treetops and grass.

Viewpoint Images may be of the same scene, but differ in camera viewpoint or lighting. It may take the viewer a little more time to recognize that the images are the same.

Semantics Regions may be similar because they contain similar objects, e.g. windows on an office building and windows on a car.

Culture and past The non-periodic lower right brick pattern of Fig. 3 may be considered to be periodic if it conjures up an image of a brick wall. Perception and semantics are both influenced by what a person has seen, and by what his or her language and culture emphasize.

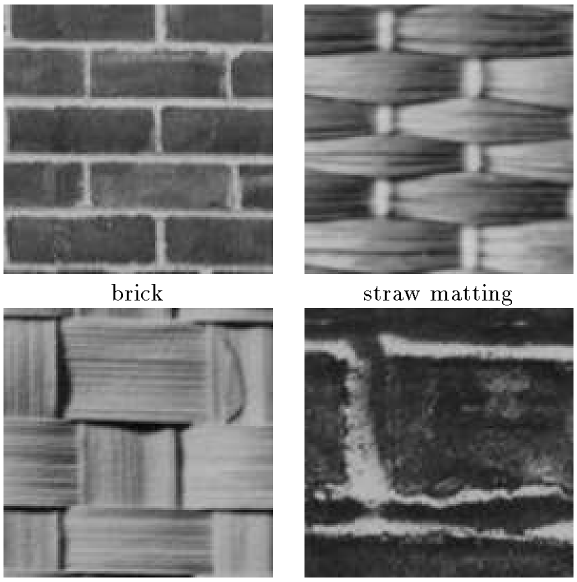

Figure 3: Image similarity can be perceptual or semantic. It can also be influenced by culture, context, or user preference. Presently, vision-based annotation helps the most with perceptual similarity; it notices that all of these have similar horizontal/vertical structure. However, one model may decide that the two brick images are most similar, and another model may decide that the top two images are most similar. People, too, will arrive at different decisions, depending on which set of features best meets their goals. The four images shown here are taken from [15].

Note that the influences above are not mutually exclusive. All four can be at work and the user will still perceive similarity. Humans do not separate these influences naturally. For example, people have difficulty perceiving small changes in perspective; most people find it hard to accurately draw a picture in perspective even though that is the way they have seen it all their life; they recognize and sketch a building as having predominately vertical parallel lines and right-angle corners, when it will almost never appear this way in a photo. Since we want the system ultimately to be evaluated by the human using it, we will have to grapple with this mixture of influences affecting similarity.

Given these influences, it seems misguided to look for some universal measure of similarity within pictures. In restricted applications, e.g. inspection for a mark of a particular size and shape, similarity matches can be made quite precisely. But in general picture retrieval, at least for the immediate future, there are important reasons to keep the human in the system, i.e. to make semi-automated tools.

The discussion above deals with locating similar regions after the human has annotated one region, so that the system can propagate the annotation. The system does not yet generate the description; it is still generated by the user. However, the system can store the user's annotations along with which texture features were used to characterize the annotated region. As the system accumulates this knowledge for different contexts, e.g. art collections vs. wedding videos, it can begin to learn which combinations of features form good predictors of descriptions. Thus, the preliminary steps here will feed naturally into larger "common-sense" systems for querying images.

2.2 There is no one best model

Over the last decade, about twenty basic texture models have been presented in the literature and shown to achieve around 90% classification on homogeneous textures. Most of the models are not designed for non-homogeneous regions as are typical in natural scenes. Very few comparisons have been made on large diverse data sets. Perhaps the strongest conclusion that can be made on these models is that their performance is data-dependent.

The data set for which we develop this annotation system is potentially the most diverse of all, for we cannot predict what images people will want to annotate. To evaluate which texture model does "best" for this unknown

Table 1: Models known by the system

Model	Description	Ref.
HIST-D	Color histogram differences	
HIST-EE	Color histogram energy and entropy	[19]
HIST-I	Color histogram invariant features	[20]
EV	Eigenvectors of RGB covariance	[21]
MSAR	Multiscale simultaneous auto-regressive	[22]
TSW	Tree-structured wavelet transform	[23]

data set is not doable in a meaningful way; it is arguable what measure would provide a universal measure of "best," given the four influences discussed above, and their own data dependency. Moreover, the "best" performance averaged over a large diverse set may coincide with individually poor performance on the region of greatest interest to the user. Conclusive results may be reached someday, but they will require carefully combining data from many digital libraries, many users, and many years of interaction and analysis.

We assume that there is neither one model that will be optimal for recognizing and annotating all kinds of stuff in pictures, nor is there a unique non-overlapping arrangement of labels that users will want to use to annotate a picture.[6] Instead, we assume that a user might assign multiple labels to possibly overlapping regions. We also assume that models will tend to specialize, and that they can work alone or together to model regions in the images. This second assumption is in the same spirit as the "Society of Mind" [18], whereby specialized agents, or models in this case, interact to make sense of what they see. In this case we have a "society of models" which interact to explain "stuff" in pictures.

We expect that only about a dozen different models might be needed; six models are in the current system – five from the recent literature, and one simple model based on a color histogram. The framework described in this paper does not depend on the specific number or choices of models; it is simple to snap models in or out because of the tree paradigm outlined below. It is an open problem to decide which texture models in the literature are best to include in this system.

2.3 Models in the present system

The six models used in the present system are listed in Table 1; they are not proposed as an optimal subset. They include four models which consider color and two which do not, and three models which are based on first order statistics and three which are based on second order statistics or filtering. As mentioned, very few conclusions exist on the relative strengths of texture models for different classes of data; hence, there is not enough information at this time to choose an optimal subset.

The six models above are described and motivated in the references, with the exception of HIST-D which is described below. Presently, it is difficult to state which models are expected to perform best on which kinds of data. The annotation system helps "discover" these relations, but a lot more data is needed before general predictions can be inferred empirically. A few comments are given below on each of the six models used presently.

The HIST-D model works as follows: Concatenate the histograms of the Ohta [24] color components I_1, I_2, I_3 into a feature vector, then compare two feature vectors with the Euclidean distance. We know of no studies which evaluate this model.

The HIST-EE model consists of the energy and entropy features of the three Ohta color channels. These have been shown [19] to improve discrimination (in comparison to filter energy features used alone) among an extended set of images formed from twelve color granite images.

The HIST-I model computes illumination invariant features of RGB histograms. These features were shown to outperform two comparable sets of color features on recognition of twenty-seven images, consisting of nine different scenes, each shot under three different conditions of illumination [20].

For the EV model, three separate covariances are computed on the R, G, and B color components, and the values corresponding to the twenty-one largest eigenvectors are kept as features for each component. Euclidean distance is used for comparison. We have not seen any published studies of the use of this model for discrimination or recognition of color images, although eigenvectors are commonly used for a variety of pattern recognition problems [21].

The MSAR model computes parameters for a second-order simultaneous autoregressive model over three scales, for a total of fifteen features [22]. The fifteen features are computed on the NTSC gray component. A Mahalanobis distance was used for discrimination. This model has been found to work better than eigenvectors for retrieval of images in the Brodatz [15] texture database [25].

The TSW features are also based only on the NTSC gray component. Unlike in [23] where their trees were averaged across several samples from the same Brodatz image, here only one tree of features is generated per selected image region because the regions are not big enough to support several samples. The wavelet trees were expanded down four levels, as in that paper. This method has been shown to outperform similar transform methods on a test set of 30 Brodatz textures [23].

2.4 Annotation to help learn context

A patch from the side of a building and a patch from a street can have identical pixel values. Perceptually indistinguishable texture patches can be extracted from a grass lawn and from mandril fur. Generally, when a person looks at a tiny region of an image they cannot tell what it is; the patches cannot be identified without context. This ambiguity is not restricted to the visual system; if someone says "Atlanta is in Miami" the sentence is confusing unless they first mention they are talking baseball. The context (baseball) must be present for accurate understanding to occur.

If a system uses texture features based only on small patches selected by the user, then the system will sometimes propagate the labels to patches which look the same, but which should not have the same label. This problem will arise in any annotation systems that ignore context.

[6] This departs from the traditional computer vision viewpoint of using one model to segment an image into non-overlapping regions before assigning labels to the regions.

Hence, this drawback of the current context-free use of vision texture is recognized up front. Multiscale and region-adjacency information, along with common-sense knowledge, along with lots of ground-truth feature-label relations are needed to make progress in incorporating context.

The present system assists directly in gathering "labeled ground truth" (annotation) for a large collection of data. This data will be helpful for learning how to deal effectively with context, as well as for other applications. Future improvements to the present system are planned for incorporating context.

3 New Photobook with annotation

We began with the MIT Photobook image retrieval system [3] [4], and have augmented it to perform interactive annotation. The evolved "interactive annotation extension to Photobook" is still called "Photobook" for short. Photobook from the user's perspective is described first below, followed by an explanation of how the system "looks at" the problem.

3.1 What the user sees

A number of decisions are involved in designing an annotation system from scratch. We have made assumptions to bootstrap the process. They are listed here to indicate that they are variables which can be explored further; none are critical to the results here.

1. Users should not be restricted to either "perceptual" labels or "semantic" labels. The system should adapt to the user's notion of similarity.
2. Users should not have to carefully trace out the entire region that they wish to label. They can label a subset and the label should propagate to visually similar regions both within the picture they are looking at, and within other pictures.
3. Users should be able to assign labels to arbitrarily shaped, possibly overlapping regions. Currently, implementation constraints only allow the labeling of coarse subimages (patches).
4. Users should get fast, visual responses from the annotation system.

In the current system, the user can browse through a database of scenes. He or she can select patches from one or more images as "positive examples" for a label, which is chosen from a menu (shown in Fig. 4). The system then propagates the label to new regions in the database that it thinks should also have the label. The user can immediately view the results. Falsely labeled patches can be removed by selecting them to be "negative examples" for the label. This is how the user "corrects" the system. The user can continue to add positive or negative examples for the same label or different labels. The system responds differently depending upon which patches are selected as positive or negative examples.

The system remembers each of the positive and negative examples designated by a user. Unless the user clears the state of the system, the system makes its decisions with the intent to satisfy everything it knows. Hence, it will always try to label everything that "looks like" positive examples while not labeling negative examples or anything that "looks like" them.

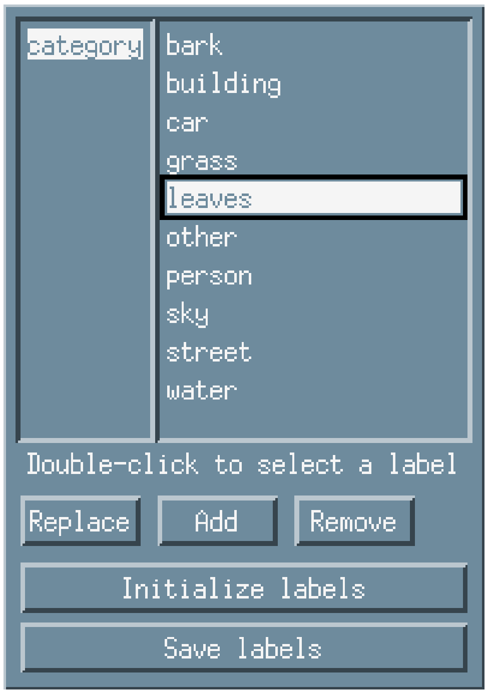

Figure 4: Photobook's label menu

3.2 How the system "sees"

Here we describe the new Photobook system with annotation. The discussion below proceeds in two parts: first, a description of the pre-processing – what is in place before the user arrives; second, a description of the run-time processing – how does the system dynamically choose models and respond to user feedback?[7]

3.2.1 Pre-processing: configuring model trees

In the examples shown in this paper, the test data consists of ten natural 512×512 color scenes taken from a set of 98 digitized vacation photos (the same set used in [10]). Each image in the database is split into square subimages or "patches" (64 per image for the current system; e.g. see Fig. 6). The pre-processing and run-time algorithms deal only with these patches; the tree formation does not currently consider what image a patch came from, or where in the image the patch was located. As mentioned above, no spatial context (region adjacency) is presently exploited.

Before the user interacts with the system, the system has already been designed to know a set of M models, and has pre-computed a tree classification for each model, T_m, $m = 1...M$. In the current system, $M = 6$ trees are built using the following steps:

1. Compute a feature vector for the model for each patch. Note that the number of features and the dynamic range of the feature values are both dependent on m.
2. Cluster the patches based on their feature vectors. Here we used the common-neighbor clustering of Jarvis and Patrick described in [26] since it is independent of the definition of feature distance. Two images are put in the same cluster if they have $k_t = 4$ of their k nearest neighbors in common and are k-nearest neighbors of each other. Neighbors were determined by computing distances using whichever distance metric was presented in the references for each model; typically this was either a Euclidean or a Mahalanobis distance.
3. Form a hierarchical tree of the clusters. Each patch is a bottom "leaf" in the tree after an initial clustering from the previous step for k=5. In this step, k is incremented, $k = 5 \ldots 20$ with each increment linking together clusters, and growing the tree upward hierarchically. Once the tree is established, distances between the feature vectors will no longer be needed; instead, distance will be measured by ancestral distance between two clusters. All patches sharing the same parent (patches are in the same cluster) are distance "0" from each other, patches sharing the same grandparent are distance "1", etc.

In the present research, each tree includes all the patches in the database, and trees are reconfigured as new data is introduced to the collection. In the future, this requirement can be relaxed and trees can specialize over portions of the data set.

[7] There is currently no post-processing, although interesting post-processing is possible based on the knowledge accumulated during a session with a user.

3.2.2 Run-time processing: model selection and user feedback

As mentioned earlier, no single model is able to robustly recognize the variety of different patches that a user might believe should have the same label. In the algorithm described next, certain tree nodes ("covers") are identified as having "best explained" the positive examples, and the "best" of these covers are chosen to guide the labeling, where "best" is defined below.

At run-time, each model has a tree, T_m, which describes that model's sense of similarity between any two patches. At any point while the user is interacting with the system, there exists a set of positive examples and negative examples for a label L. Each time this set changes, the system updates the labeling according to the method below.

Define a "cover" as a tree node which parents (or "covers") at least one positive example and no negative examples. Then a "hypothesis" for the examples is a set of covers which together parent all positive examples. A positive example may be covered by more than one parent, i.e. overlap is permitted.[8] The covers in a hypothesis may be taken from one or more of the M trees.

Under this formulation, the system need only determine the "best" such hypothesis. The "best" hypothesis here is taken to be the hypothesis which has the fewest covers and, of those hypotheses, parents the fewest total leaves, i.e. the covers are few and small.

Finding the best hypothesis over all nodes in all trees is combinatorially prohibitive, so we employ a two step approximation. First, the best hypothesis is found for each tree. The nodes in those M hypotheses are then pooled and searched for the overall best hypothesis. This algorithm can be formalized as:

1. $C_m = \mathrm{BestHyp}(\mathrm{Covers}(T_m))$
2. $C = \mathrm{BestHyp}(\bigcup_m C_m)$

where $\mathrm{Covers}(T)$ is the set of covers in tree T, $\mathrm{BestHyp}(S)$ is the best hypothesis given the set of covers S, and C is the overall best hypothesis. Finding the optimal $\mathrm{BestHyp}(S)$ is NP-hard, so we approximate it with a greedy algorithm that iteratively chooses the "best-looking" cover in S and places it in $\mathrm{BestHyp}(S)$ until all positive examples have been covered. The "best-looking" cover in S is the one which parents the most uncovered positive examples and, of those covers, parents the fewest total leaves.

Once the best hypothesis C is found, all images parented by a cover in C are assigned the label L. The system follows this algorithm for each possible label L, and patches may end up with multiple label assignments. In this case, the system displays a list of the assigned labels under the patch.

The current system has no concept of the "quality" of an example, i.e. the fact that some positive examples are more prototypical of a label than others, and that some negative examples are more anti-typical than others. For the labels used here, a prototype for one label is usually not a good prototype for another; in fact, it is usually a negative example of the other. The current system exploits this phenomenon by assuming that a positive example for

[8] In AI terminology, this system learns "internally disjunctive disjunctive concepts given instances over M tree-structured attributes."

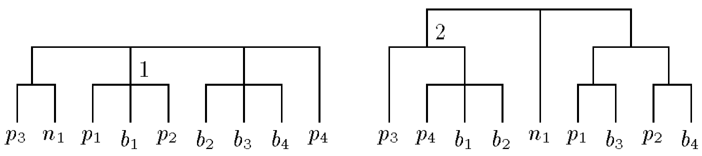

Figure 5: Two hypothetical model trees. Given the four positive examples $p_1 ... p_4$ and one negative example n_1, the overall best hypothesis will contain node 1 (because it covers two positives with 3 total leaves) and node 2 (because it covers the remaining two positives). Hence patches p_1, b_1, p_2, p_3, p_4, and b_2 are labeled. Patches b_3, b_4, and n_1 remain unlabeled. Using the left tree alone would given the same results, except b_2 would also remain unlabeled. Using the right tree alone would have labeled all patches except n_1.

one label is a negative example for all others. That is, the interface fabricates a set of "implicit" negative examples whenever the user inputs a positive one. This is occasionally harmful, but more often it significantly reduces the number of inputs required to reach a satisfactory labeling. In the future, we plan to relax the strict non-overlap of labels imposed by this assumption by adding an internal "quality" factor for examples.

3.3 Multiple labels for the same patch

The user currently is allowed to pick one label per patch; however, a patch may still receive multiple labels. Consider a picture of trees against a blue sky. If the user labels several pure sky patches "sky" and pure tree-top patches "leaves" and does not label the mixed sky/leaves patches, then the system may label these patches with *both* the "sky" and "leaves" labels. The success of this "mixture labeling" appears to be a by-product of the use of multiple models. With only one model, the mixed sky/leaves patches could either be grouped with sky, with leaves, or with neither. With different models for sky and leaves, the mixed patches can be grouped with sky by one model, and with leaves by the other model. Hence, a mixed region can automatically receive multiple labels.

3.4 Performance examples of the system

Examples of annotation in the new Photobook system are shown and discussed below. In our tests, the best overall hypothesis was usually identical to the best hypothesis for some model tree. In such a case, we say that model alone was the "best model" for the labeling. One case where multiple models were used (not illustrated here) was where one kind of building was in one scene and another kind was in another scene. The user selected regions of the buildings from both scenes to be labeled "building." One model gave the best cover for the buildings in one scene, and another model gave the best cover for the buildings in the other scene. The two models were united to explain the common label and produce the overall hypothesis.

In Fig. 6, the user selected four prototypes (positive examples) of "sky" (outlined in red – look in upper left) and two prototypes of "grass" (outlined in red – lower left). The system decided that model MSAR was best for grass, and used it to find and label the ten additional grass patches. The model EV was found to be best for sky, and was used to find and label the other twenty-eight sky patches.

The performance may vary depending on exactly which patches are selected as prototypes. Typically, selecting two to four prototypes that are "representative" of the desired label gives the best performance initially. Because of the large block-size, and correspondingly, the tendency for one block to contain more than one region, the performance in the present system is roughest around the edges. However, fine segmentation is not necessary for most retrieval applications. Knowing that "about half" of an image consists of regions labeled "building" is sufficient for many kinds of retrieval.

Figure 7 shows an example of labeling a city shot. Here the user selected three prototypes of "building," two of "car," and two of "street" (each prototype is outlined in red). The system selected the model HIST-D for all three classes of labels and labeled the regions in a satisfying way to the user. Note that within a class, even though each prototype patch is quite different, they contain the same kind of visual "stuff," in the context of this database. If the "car stuff" is sufficiently unique compared to the other content in the database, this low-level approach meets with success. The idea that simple color histograms could provide useful invariants for object recognition was demonstrated in [11]. Thus, vision texture is sometimes able to find objects like cars, even though cars are not usually considered texture.

The true performance of a labeling system is ultimately judged by the user. In the two examples above, the user achieved a labeling that was deemed satisfactory without having to correct the system.

Figures 8 and 9 illustrate the response of the system to user feedback via negative examples. Initially, the user selected the four red-outlined "leaves" regions in Fig. 8. The model TSW was chosen for finding and labeling fourteen other patches as "leaves." The user disagreed with the labeling of the roof/window patch by TSW and indicated that the patch is a negative example for the class "leaves." This feedback resulted in another labeling hypothesis which labeled more of the house and the fence area as "leaves" (this iteration is not shown). The user then de-selected the fence patch, giving a satisfying labeling. This labeling is shown in Fig. 9, with the two patches in red indicating the two negative examples used after the result of Fig. 8.

Note that each interaction with the user modifies the set of input examples. Hence, the system is free to choose a new model or combination of models at any time during the interaction. This choice happens transparently; the user only sees the selected/de-selected patches and the labels.

Over the three cases illustrated in Figs. 6–9, the average annotation gain (savings to the user) can be measured to be 1:5. This gain is the ratio of inputs from the user to the total outputs which get labeled satisfactorily. Inputs are the total number of positive and negative examples. For example, in Figs. 8 and 9 the total inputs was 6 and the total outputs (after the user was satisfied) was 16. In Fig. 6 the gain ratio was 6:44. Note that performance ratios such as this always depend on error tolerance: the more false labels you allow, the more true labels you can get. This trade-off becomes more apparent in a large diverse database (example below).

4 Extensions and applications

The performance of the system as described above is currently interactive-time (on an HP 735 workstation) showing labelings immediately after the user has selected prototypes. Note that most of the work is done off-line before the user interacts with the system. Only if new data is added suddenly by the user will the system need to pause and reconfigure its trees. Currently the user cannot add new data during annotation, but this is an area for future extension.

We have recently run the above system on the entire set of 98 vacation photos ($98 \times 64 = 6272$ patches). Although the labeling is slower, we have identified small adaptations that should return the process to interactive-time. Our initial implementation builds the same trees except for omitting the TSW.[9] Given 250 inputs of positive and negative examples, the system labeled an additional 1169 patches. Of the system-labeled patches, about 90% were estimated to be correct by the user. Hence, although the system size scaled up by a factor of ten with a diverse collection of images, the labeling still provides a 1:4 gain (with ten percent error tolerance) in terms of saving work for the user.

4.1 Learning from the annotations (post-processing)

One immediate application of the above results is in retrieving images that have a certain content, i.e. "find images that are mostly grass and sky." Another step is to begin to learn higher image categories, e.g. the system may be taught that shots that are half sky at the top and largely grass at the bottom are "outdoor, open shots." Certain mixes of content, e.g. "car, street, building," can also be associated with certain contexts, e.g. city shots.

One can imagine many things to do with the labeled output at this stage. For example, one can go back and determine if particular models "specialize" in finding particular types of image content. Consistent feature-label pairings could also be used to predict annotations given only image data. For example, it might take the HIST-D model features found to correspond well to buildings, and make a note of these for a future query on buildings in an un-annotated system. The results at this stage have opened the door to interaction between high-level description and low-level textural features.

4.2 Sound textures for annotating audio collections

The principles demonstrated here also apply to "sound textures" in digital audio libraries. Like visual textures, sound textures can be stochastic or periodic, e.g. the stochastic sounds of applause or a bubbling fish tank, or the repetitive "chunk chunk sluurrp" of a copy machine. Texture is useful not only for recognizing such sounds, but also for describing concepts such as timbre and rhythm. Many of the same models used for vision texture appear to work well for sound texture [27] [28].

To adapt the method presented here, one may substitute sound features for image features. One can also easily substitute new models for the six here. Once the classification trees are built, the user would select a segment of sound to be labeled, and then the system would decide which model or combination of models best characterized it. It would then propagate the label to "perceptually similar" sounds based on that characterization. As in the case of the visual database, pairings of labels and features can be accumulated and used to learn automatic labelings and higher categories. Eventually the system may acquire enough knowledge to be able to query un-annotated data, and to do most of the common (predictable across different users) annotation on its own.

5 Summary

A new system that combines low-level texture with high-level descriptions to assist users in annotating digital libraries has been demonstrated. To the best of our knowledge, this system is new in the following ways:

- It illustrates the use of texture for assisting the user interactively in annotating multimedia databases.
- It dynamically selects from multiple texture models, based on the behavior of the user in selecting a region for labeling.
- It uses trees of clusters as an internal model representation, which makes it flexible enough to allow combinations of clusters from different models. Hence, if no one model is "best" then it can create a new hypothesis by pruning and merging relevant pieces from the model trees it knows.
- It does not depend upon a metric similarity space during annotation; the metrics are only used to initially cluster the patches into hierarchical trees. The trees are fast to search, permitting interactive-time comparison among the multiple models.
- It has persistent knowledge; it stores what it has learned in terms of positive and negative examples, and uses these to improve its labeling ability.

Acknowledgements

We would like to express our gratitude to A. P. Pentland who provided the impetus to proceed with the annotation application, to S. Elo for discussions about WordNet, to B. Moghaddam for his comments in preparing this paper, and to the following people who contributed code: F. Liu – MSAR, B. Moghaddam – EV, C.-C. J. Kuo and T. Chang – TSW, and D. Slater and G. Healey – HIST-I.

References

[1] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos,

[9] Based on experience searching with TSW in the Photobook environment, we expected it would give poor recognition on the patches. It was also the most computational for preprocessing so it was omitted from the larger study.

and G. Taubin, "The QBIC project: Querying images by content using color, texture, and shape," in *Proc. SPIE Storage and Retrieval for Image and Video Databases* (W. Niblack, ed.), (San Jose, CA), pp. 173–181, SPIE-The Society for Optical Engineers, Feb. 1993.

[2] S. W. Smoliar and H. Zhang, "Content-based video indexing and retrieval," *IEEE Multimedia*, pp. 62–72, Summer 1994.

[3] R. W. Picard and T. Kabir, "Finding similar patterns in large image databases," in *Proc. ICASSP*, (Minneapolis, MN), pp. V–161–V–164, 1993.

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

[5] H. Tamura, S. Mori, and T. Yamawaki, "Textural features corresponding to visual perception," *IEEE T. Sys., Man and Cyber.*, vol. SMC-8, no. 6, pp. 460–473, 1978.

[6] A. Treisman and G. Gelade, "A feature-integration theory of attention," *Cognitive Psychology*, vol. 12, pp. 97–136, 1980.

[7] A. R. Rao and G. L. Lohse, "Towards a texture naming system: Identifying relevant dimensions of texture," in *IEEE Conf. on Visualization*, (San Jose), 1993.

[8] T. Syeda-Mahmood, "Model-driven selection using texture," in *Proc. 4th British Machine Vision Conference* (J. Illingworth, ed.), (Univ. of Surrey, Guildford), pp. 65–74, BMVA Press, Sept. 1993.

[9] R. J. Herrnstein, D. H. Loveland, and C. Cable, "Natural concepts in pigeons," *J. of Exp. Psych: Anim. Beh. Procs.*, vol. 2, pp. 285–302, 1976.

[10] M. M. Gorkani and R. W. Picard, "Texture orientation for sorting photos at a glance," in *Proc. Int. Conf. Pat. Rec.*, (Jerusalem, Israel), Oct. 1994.

[11] M. J. Swain and D. H. Ballard, "Indexing via color histograms," in *Image Understanding Workshop*, (Pittsburgh, PA), pp. 623–630, Sept. 1990.

[12] G. Miller, "Wordnet: An on-line lexical database," *Int. Journal of Lexicography*, vol. 3, no. 4, 1990.

[13] A. S. Chakravarthy, "Toward semantic retrieval of pictures and video," in *RIAO'94, Intelligent Multimedia Information Retrieval Systems and Management*, (New York), Oct. 1994.

[14] R. W. Picard and F. Liu, "A new Wold ordering for image similarity," in *Proc. IEEE Conf. on Acoustics, Speech, and Signal Proc.*, (Adelaide, Australia), pp. V–129–V–132, April 1994.

[15] P. Brodatz, *Textures: A Photographic Album for Artists and Designers*. New York: Dover, 1966.

[16] W. Richards and J. K. Koenderink, "Trajectory mapping ("TM"): A new non-metric scaling technique," Center for Cognitive Science 48, MIT, Oct. 1993.

[17] A. Tversky, "Features of similarity," *Psychological Review*, vol. 84, pp. 327–352, July 1977.

[18] M. Minsky, *The Society of Mind*, p. 45. New York: Simon & Schuster, 1985.

[19] T. S. C. Tan and J. Kittler, "Colour texture classification using features from colour histogram," in *SCIA Conference on Image Analysis*, vol. 2, 1993.

[20] G. Healey and D. Slater, "Using illumination invariant color histogram descriptors for recognition," in *Proceedings of CVPR*, (Seattle, WA), pp. 355–360, IEEE, Braun-Brumfield, Inc., June 1994.

[21] C. W. Therrien, *Decision Estimation and Classification*. New York: John Wiley and Sons, Inc., 1989.

[22] J. Mao and A. K. Jain, "Texture classification and segmentation using multiresolution simultaneous autoregressive models," *Patt. Rec.*, vol. 25, no. 2, pp. 173–188, 1992.

[23] T. Chang and C.-C. J. Kuo, "Texture analysis and classification with tree-structured wavelet transform," *IEEE T. Image Proc.*, vol. 2, pp. 429–441, Oct. 1993.

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

[25] R. W. Picard, T. Kabir, and F. Liu, "Real-time recognition with the entire brodatz texture database," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, (New York), pp. 638–639, June 1993.

[26] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice Hall, 1988.

[27] A. S. Sherstinsky, *M-Lattice: A System for Signal Synthesis and Processing Based on Reaction-Diffusion*. ScD thesis, MIT, 1994.

[28] N. Saint-Arnaud, Spring 1994. Class project for MIT Pattern Recognition and Analysis.

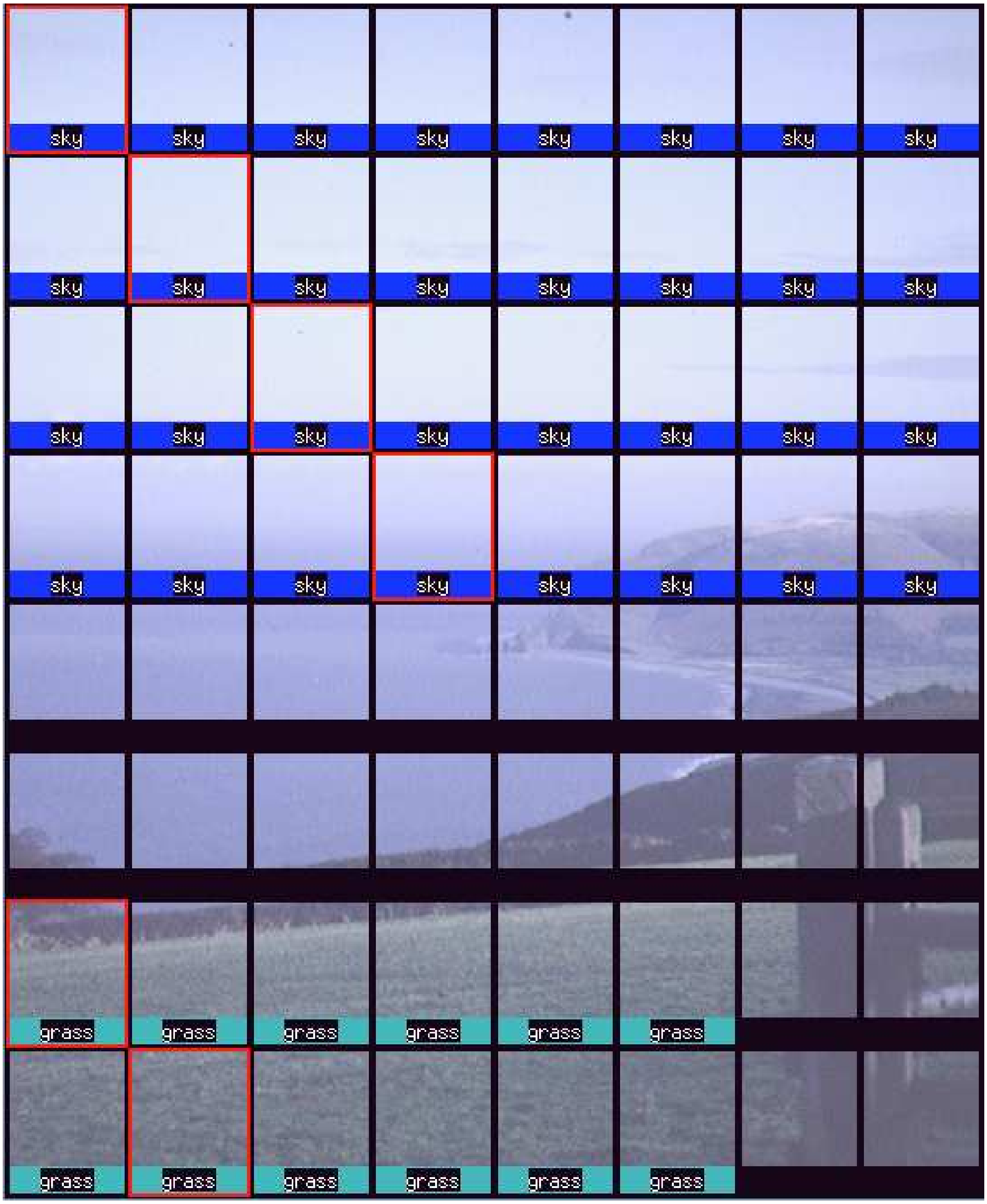

Figure 6: The user input six positive examples for this scene (outlined in red); the computer generated forty-four satisfactory labels

Figure 7: The user input seven positive examples for this scene; the computer generated thirty-seven satisfactory labels

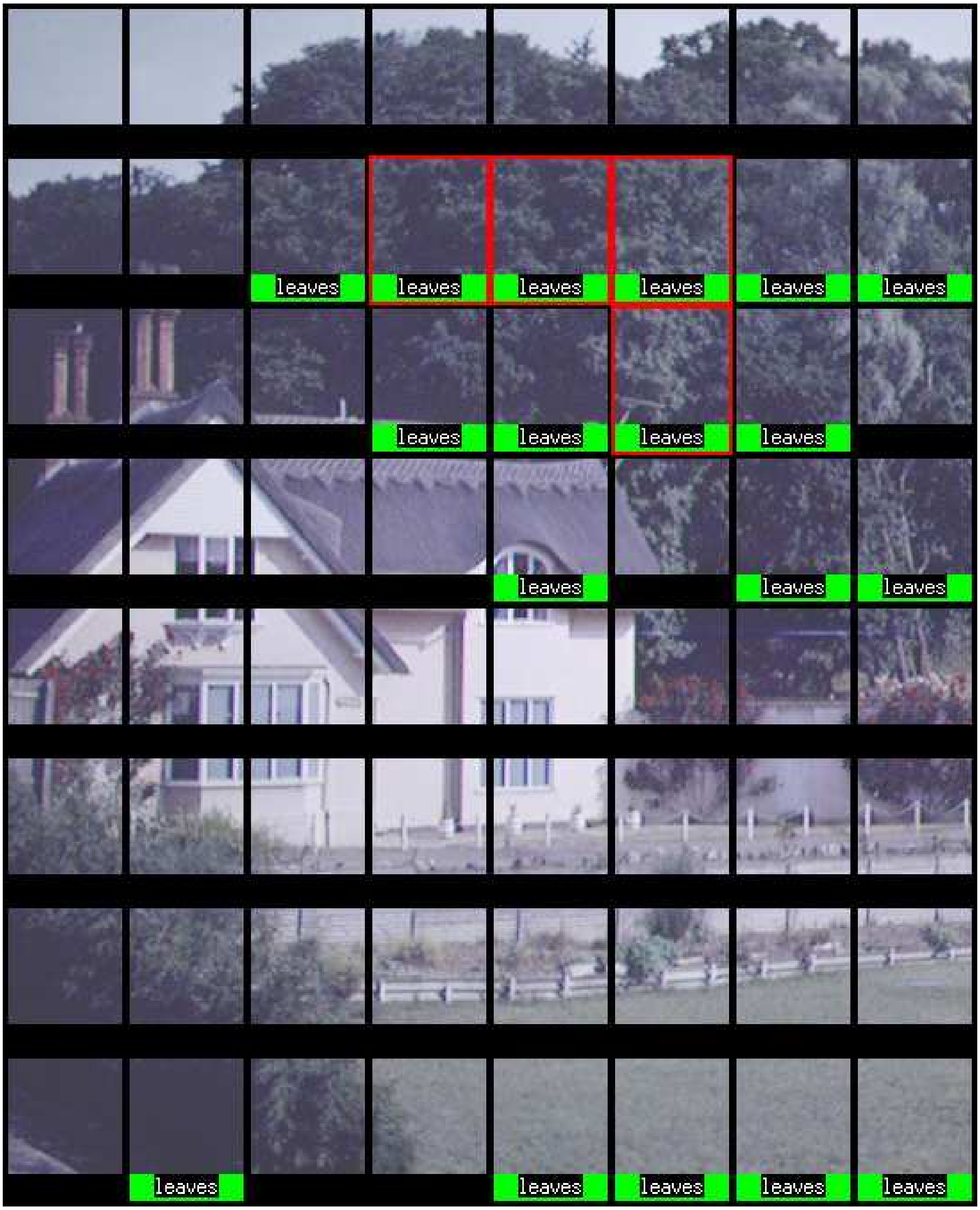

Figure 8: The user input four positive examples; the system generated many satisfactory and unsatisfactory "leaves" labels in this image

Figure 9: The user gave corrective feedback (negative examples outlined in red) to the output shown in Fig. 8, resulting in sixteen satisfactory labels