Localized Content Based Image Retrieval [*]

Rouhollah Rahmani
Department of CSE
Washington University
St Louis, MO 63130
rahmani@wustl.edu

Sally Goldman
Department of CSE
Washington University
St Louis, MO 63130
sg@wustl.edu

Hui Zhang
Department of CSE
Washington University
St Louis, MO 63130
huizhang@wustl.edu

ABSTRACT

Classic *Content-Based Image Retrieval* (CBIR) takes as input a single query image provided by the user, and retrieves similar images from an image repository. Inherently, such a search must rely upon a holistic (or global) view of the image. Yet often the desired content of an image is not holistic, but is localized. Specifically, we define *Localized Content-Based Image Retrieval* as a CBIR task where the user is only interested in a portion of the image and the rest of the image is irrelevant. While many classic CBIR systems use relevance feedback to obtain labeled images that are used to refine the search, most such systems rank images using a global similarity measure. In Localized CBIR, labeled images must be used to do more than re-weighting the features (e.g. color, texture). In this paper we present a localized CBIR system, A*cc*io! , that uses labeled images in conjunction with a multiple-instance learning algorithm to first identify the desired object, and then to rank images in the database by their content. We evaluate our system using a five category natural scenes image repository, and benchmark data set that we have constructed of 25 image categories.

General Terms

Image Retrieval, Relevance Feedback

Keywords

Content Based Image Retrieval, Multiple Instance Learning

1. INTRODUCTION

Classic *Content-Based Image Retrieval* (CBIR) takes a single *query image* provided by the user, and retrieves similar images from an image repository. Since the user typically provides a single image with no indication of which portion of the image is of interest, inherently, such a search must rely upon a holistic (or global) view of the image. Yet often the desired content of an image is not holistic, but is localized. Specifically, we define *Localized Content-Based Image Retrieval* as a CBIR task where the user is only interested in a portion of the image and the rest of the image is irrelevant.

Unless the user explicitly marks the region of interest or the region of interest is at a fixed location, localized CBIR must rely on multiple labeled images to learn which portion of the image is of interest to the user. We define the *query image set* as a set of images labeled by the user as positive or negative depending on whether or not they contain the desired content. The query image set can either be directly provided by a user or can be obtained using relevance feedback by adding labeled feedback images from the result of a search to a single, initial query image (labeled as positive). While many classic CBIR systems use relevance feedback to refine the search, most such systems still rank images using a global similarity measure that is re-weighted based on the feedback images. In Localized CBIR, the query image set must be used to identify the portion of the image that is relevant to the user's search, as well to determine an appropriate weighting of the features (e.g. color, texture). Furthermore, the relevant portion of the image is often an object that may vary in size and location. In this paper we present a Localized CBIR system, A*cc*io! . A*cc*io! uses a small query image set in conjunction with a multiple-instance learning algorithm to identify the desired local content, to re-weight the features, and then to rank images in the image repository by their content.

The image representation used within A*cc*io! enables it to obtain good performance even when the desired content is a complex object defined by many parts. Furthermore, A*cc*io! 's ranking algorithm is general enough for the situation in which the desired images are defined by many different objects. Thus A*cc*io! can be applied to standard CBIR tasks as illustrated by its performance on recognizing natural scenes.

We evaluate A*cc*io! using a natural scenes image repository with five categories containing 900 images, and using a benchmark data set that we have constructed with 25 image categories containing 4650 images. This data set emphasizes the task of Localized CBIR. The categories consist of images of single objects photographed against highly diverse backgrounds. The objects may occur anywhere spatially in the image and also may be photographed at a wide-angle or close up. Often the object of interest occupies less than 10% of

[*]Draft – Do Not Distribute

the image area.

We demonstrate the performance of A*cc*i*o*! by showing the top ranked images for a sample query image set. We also present some recall and recall-precision plots for some query sets. However, one motivation for providing a benchmark data set is to enable systematic comparison between different systems. In order to achieve this goal, a single numeric measure of performance is needed to compare across a large number of tasks, and to determine when differences in performance are statistically significant. There are many measures that could be used as the measure of performance, such as the value where recall=precision or the area under the ROC curve. In this paper, the measure we use is fraction of the images that are properly labeled as positive or negative (i.e. the *accuracy*) when the top p images in the ranking are classified as positive where p is the number of positive images in the test set. We use 30-fold cross validation, where for each category, a fixed number of positive and negative images are randomly selected from the image repository to be used for the query image set. The remaining images, called the *test set* are used to measure the performance. This process is repeated 30 times.

We compare the performance of A*cc*i*o*! under different levels of granularity for the image segmentation and also against the performance of a variation of Simplicity [16] that is designed to use a query image set (of any size), as opposed to using a single image. [[[Reviewers: Additional text will be added summarize our findings.]]]

The remainder of this paper is organized as follows. Section 2 discussed previous work that is most relevant to our work. A*cc*i*o*! is described in Section 3 starting from the system architecture followed by a discussion of all the components. Experimental results are presented in Section 4. Finally, our conclusions and a discussion of future work is in Section 5.

2. PREVIOUS WORK

Content-Based Image Retrieval (CBIR), is the problem of retrieving semantically relevant images from a large image database. In a typical CBIR application, there is a query language used by the human user to specify a set of images that are desired. *Relevance feedback* is defined as having the the user to label images from the results of a query as either relevant or irrelevant. These labeled images are commonly referred to as the *feedback images*. The feedback images are typically used to weight the features of the query image and then to refine the results [5, 10, 2, 7, 8, 14].

While traditional relevance feedback weights the global features of the image [5, 10], in recent years several papers have focused on weighting spatially local features. Tian et al. [15] introduced two relevance feedback techniques for retrieving images by local content – *Region-of-Interest* (ROI) and *Spatial Layout*. Both techniques partition all images into a uniform $N \times N$ grid of image blocks, where N is typically between 2 and 5. Spatial Layout employs standard relevance feedback at the block level by weighting each feature of each blocks of the query image using the spatially corresponding blocks of the feedback images. The images in the image repository are ranked based on the combined similarity of its blocks to the spatially corresponding blocks of the weighted query image. Images with a rank above a given threshold are classified as positive (i.e. desirable). ROI is an extension to Spatial Layout in which the user is asked to draw a bounding box around the region of interest in the query image. ROI then weights each block relative to how much of its area is in the region of interest box. Both ROI and Spatial Layout can be viewed as an approach for localized CBIR that is based on an image-independent partitioning of the image into regions. Furthermore, they assume that the content of interest appears in the same region in the query image, feedback images, and relevant images from the repository.

Kim, Park, and Kim [9] describe an image segmentation technique for a task they call *central object extraction*. Their technique aims to segment a possibly complex object that occurs in the center of the image from the background in which it occurs. They then discard the background and employs standard relevance feedback using only the central object. When applied to localized CBIR, this technique requires the the object of interest is in the center of the image. Thus, as with ROI and Spatial Layout, it does not generalize to retrieval of local content that may occur anywhere within the image.

Maron and Ratan [11] proposed a different approach to Localized CBIR that applies *multiple-instance* (MI) learning to the task of recognizing a person from a series of images that are labeled positive if they contain the person and negative otherwise. Their technique first divides all images into nine fixed, uniform-size, overlapping regions that consist of a single central block and four neighboring blocks. A new multiple-instance learning algorithm, Diverse Density, was used to both learn which block was of interest, and to weight the features. The same technique was used to learn descriptions of natural scene images (such as a waterfalls) and to retrieve similar images from a large image repository using the learned concept [12]. (See Section 3.3 for for further discussion of the MI model.) The benefit provided by the multiple-instance learning algorithm is that the content of interest need not be in the same block in all images. However, since pre-defined blocks are used to divide the image, difficulties arise when the object(s) of interest has important pieces that cross two blocks, or if it is too small to significantly affect the average feature values in a block. As with ROI and Spatial Layout, the work of Maron et al., use an image-independent partitioning into blocks.

Zhang and Goldman [18] applied multiple-instance learning to localized CBIR using a segmentation algorithm to partition the image into segments (that can vary in size and shape). In addition, they introduced features to capture texture, and used the EMDD multiple-instance learning algorithm [19]. However, unlike the work of Maron et al, which uses neighboring blocks to provide information about the context of each block, Zhang and Goldman did not use any such neighboring information, which is much less straightforward when the image is not divided into pre-defined blocks.

Huang et al. [8] presented a variation of Zhang and Goldman's work that incorporated a different segmentation scheme, and a neural network based multiple-instance algorithm. Hofmann et al. [1] also applied the MI model to natural scene selection through two new algorithms which combined the multiple instance model and support vector machines.

3. ACCIO!

In this section we describe Ac*cio*! . We first describe the overall architecture for our system that is outlined in Figure 1. The user provides a query image set to Ac*cio*! . As discussed in Section 1, the query image set could be obtained by adding the feedback set from relevance feedback to a single, initial query image (labeled as positive). There are some settings in which the user may directly create a query image set. For example, frames from surveillance video could be available for times when suspicious activity occurred (labeled as positive), and others for times when nothing out of the ordinary occurred (labeled as negative). The image repository would contain unlabeled video frames. Ac*cio*! could be used to to search for frames in the image repository that have some object in common with those in which there was suspicious activity.

Each image used by Ac*cio*! is first pre-processed and segmented as described further in Section 3.1. While the currently used segmentation algorithm produces contiguous segments [1], Ac*cio*! is designed to use any segmentation algorithm. Next the *bag generator* is used to convert the segmented image into the format used by the multiple-instance learning algorithm. During this phase, each segment in the image has feature information (color and texture) added for both it and its neighboring segments. Section 3.1 describes this in further detail. To reduce overhead during the ranking phase, all images in the repository have already been processed. Their representation as a bag is stored in the image repository along with their corresponding raw images and segmented images. Whenever an image in the query set is from the image repository, the segmentation algorithm and bag generator are by-passed. Instead, the stored bag for that image is used.

The next step is to provide the labeled bags from the query image set to the *multiple-instance learning algorithm*. Currently, Ac*cio*! uses the EMDD algorithm (as discussed in Section 3.4). EMDD performs a gradient search with multiple starting points to obtain multiple hypotheses that are consistent with both the positive and negative images in the query image set. While, one could select the "best" hypothesis returned from among all runs of the gradient search, it is beneficial to use all hypotheses when the desired content involves more than one object or a single object that can be defined in multiple ways. This provides a more robust localized CBIR system. Thus the multiple-instance learning algorithm returns a set of hypotheses which together can be used to rank the images in the repository.

Finally, a ranking algorithm combines the set of hypotheses output by the learning algorithm to obtain an overall ranking of all images in the repository. (See Section 3.5.) These ranked images are shown to the user. As with classic CBIR systems that use relevance feedback, if the user is satisfied with his results, the current search has been completed. Otherwise, the user has the option of marking some of the ranked images as "desired" or "not desired," and either augmenting the query images by this additional set, or by using these labeled images as a new query image set.

[1]A segment x said to be contiguous if there exists a connected path of pixels in x between any two pixels of x.

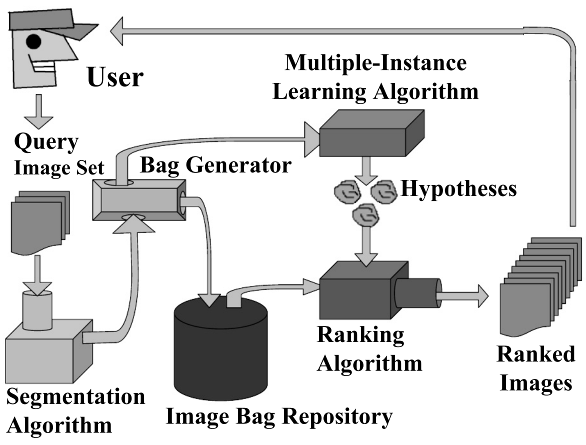

Figure 1: The system architecture of Ac*cio*! . The query set can consist of a single image or multiple images. The image repository holds the raw images, as well as the representation as a bag used by the multiple-instance learning algorithm.

3.1 Image Representation

All images are transformed into the YCrCb color space[2] and then pre-processed using a wavelet texture filter. After pre-processing, each pixel in the image has six features consisting of three color values and three texture values. (The texture values are defined using a neighborhood around the pixel.) If desired, other features such as those based on shape could also be included. Next, the *Improved Hierarchical Segmentation* (IHS) algorithm [17] is used to segment the image into a specified number of segments as set by the user. Including more segments generally leads to better performance, but at the cost of increased computational complexity for the learning algorithm.

The IHS algorithm uses a bottom-up approach that can make use of any desired similarity measure between two segments. The current implementation uses the Euclidean-distance for this similarity measure. IHS begins with a segmentation where each pixel is an individual segment. Until a stopping condition is reached, IHS groups adjacent regions that are the most similar. The output of IHS is a tree in which there is a leaf for each pixel and an internal node represents two or more merged regions (segments) from the next lower level of hierarchy. Thus IHS creates a hierarchy where the top level of the hierarchy is the final segmentation, and each level defines a finer-grained segmentation. Several parameters can be used in IHS to define a stopping condition. The option currently used by Ac*cio*! is the number of segments. Another option that would be appropriate for Ac*cio*! is to use a feature distance threshold for a stopping

[2]Ac*cio*! could be easily modified to use a different color space, if desired.

```
Neighbors($I$)
  For each $x \in I$            //$x$ is a segment in $I$
    For each $p \in x$          //$p$ is a pixel in $x$
      For each $j \in \{North, East, South, West\}$
        If $N_j(p) \in y$ for $y \neq x \Rightarrow$ vote for $y$
        If $N_j(p) \in$ image border $\Rightarrow$ vote for $x$
        If $N_j(p) \in x \Rightarrow$ do not vote
  For each direction $j$
    $N_j(x)$ = segment with the most votes
```

Figure 2: Pseudo-code for Neighbors. Let I be the set of all segments in the image, and let $N_j(p)$ be the pixel that neighbors pixel p in cardinal direction j.

condition. Thus while the stopping condition currently used by A*cc*io! results in all images having the same number of segments, IHS could result in images having a varying number of segments.

The content desired by a user cannot always be defined by a single segment. For example, a flower is a set of petals adjacent to green leaves. A mountain is snow and rock adjacent to a blue sky. There is a semantic gap between the information provided by a single segment and the high level content the user perceives. The petals in and of themselves do not define the flower, nor does the snow define the mountain. Often it is the immediate surroundings that allow for a more complex and descriptive definition of the desired contents in the image. Thus our final representation augments the feature information for each segment with the difference between the corresponding feature of their neighbors.

Since A*cc*io! is designed to use any segmentation algorithm, it is necessary provide a well-defined notion of a neighbor even when segments may or may not border the image on any side, can have any number of adjacent segments, and may not even define a contiguous regions (such as for the k-means segmentation algorithm [6]). Figure 2 presents our algorithm to compute four neighbors for every segment, one in each cardinal direction. Intuitively, the algorithm selects the most prominent neighbor in each cardinal direction.

Figure 3 illustrates this algorithm on an image of a bird flying past a mountain which has a dirt road leading to it. The segment for the mountain selects the sky as its northern, western, and eastern neighbor. It ignores the clouds, and the bird. The ground is selected as the southern neighbor, and the smaller road is ignored. The segment for the bird, selects the the mountain and the sky as its neighbors. So, a mountain is defined as a sky above and the ground below. And a bird is defined as being an object surrounded by the sky or by a mountain.

The feature vector for each segment is augmented with the difference between its neighbors' value and its value for each of the six features, for all four neighbors. We use the difference between these feature values to allow for robustness against global changes in the image, such as changes in brightness and hues that result from different lighting.

Throughout the remainder of this paper, we use the following geometric view for the image representation. Let I be a segmented image. Each segment $x \in I$ can be viewed as a point in a 30-dimensional feature space. Specifically, the first 6 features hold the average color and texture values for x. The next 6 features hold the difference between the average color and texture values of the northern neighbor and x. Similarly there are 6 features for the difference information between x and its other 3 cardinal neighbors. Thus there are a total of $6 + 6 \cdot 4 = 30$ features. Each image is thus a set (or *bag*) of points. For example, an image that is segmented into 8 components, is represented as a bag holding 8, 30-dimensional points. While it is important that all points have the same number of dimensions, the size of the bag can vary between images. That is $|I|$ need not be the same for all images. Finally, the query image set is stored and provided to the learning algorithm as a set of labeled bags, with one bag for each image in the query set.

3.2 Hypothesis Representation

A*cc*io! uses a multiple-instance learning algorithm to learn a set of hypothesis that are then used to rank the images. Each hypothesis captures the "ideal" feature values for the desired object along with a weighting of the features to indicate the degree of importance to the user's query. Each hypothesis is represented by two 30-dimensional feature the vectors, $\vec{h}$ and $\vec{s}$, where $\vec{h}$ represents the point in the 30-dimensional space that corresponds to the "ideal" feature vector, and $\vec{s}$ is a weight vector used to define the weighted Euclidean metric that is used as a similarity measure between points.

For example, suppose the image query set contained several red, yellow, and orange flowers labeled as positive, along with some non-flower images (which may also contain yellow, orange, or red objects in them) labeled as negative. A ideal hypothesis describing these flowers may be a feature vector that has brightly colored features for the central segment (representing the petals) and dimmer, roughly textured features for the neighbor segments (representing the leaves and stem). The importance of the bright colors and the surrounding rough texture is embodied by the scale vector.

In this example, it may not be possible to find a single hypothesis that distinguishes flowers from non-flowers. However, it may be possible to find a hypothesis that distinguishes red flowers from non-flowers, and one that distinguishes orange flowers from non-flowers, and so on. It is in such cases that A*cc*io! can benefit from the multiple hypothesis returned from the multiple-instance learning algorithm. Each hypothesis might describe one flower color and then they are combined by the ranking algorithm (see Section 3.5).

3.3 Multiple-Instance Learning Model

The multiple-instance (MI) learning model was first formalized by Dietterich et al. [4] for the problem of drug discovery. Unlike standard supervised learning in which each instance is labeled in the training data, in this model each example

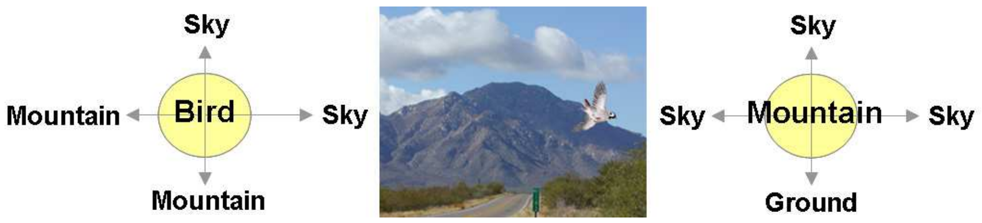

Figure 3: Illustration of the neighbor algorithm of Figure 2.

$$\begin{array}{lll} \{\underbrace{\langle p_{1,1}, \ell_{1,1}\rangle, \ldots \langle p_{1,|B_1|}, \ell_{1,|B_1|}\rangle}_{\{\langle B_1, \ell_1\rangle,}, & \underbrace{\langle p_{2,1}, \ell_{2,1}\rangle, \ldots \langle p_{2,|B_2|}, \ell_{2,|B_2|}\rangle}_{\langle B_2, \ell_2\rangle,}, & \ldots \ \} \ \text{standard supervised learning} \\ & & \ldots \ \} \ \text{multiple-instance learning} \end{array}$$

Figure 4: An illustration of the distinction between multiple-instance learning and standard supervised learning.

is a set (or *bag*)[3] of instances which is labeled as to whether any single instance within the bag is positive. The individual instances are not given a label. The goal of the learner is to generate a hypothesis to accurately predict the label of previously unseen bags.

We now formally define the MI learning model and relate it to standard supervised learning. For ease of exposition, we define this model for learning a target concept that is a d-dimensional weighted hypothesis where each instance is a d-dimensional point. In particular, the target concept $(\vec{t}, \vec{s})$ consists of point $\vec{t} = (t_1, \ldots, t_d)$ and scale (weight) vector $\vec{s} = (s_1, \ldots, s_d)$. For an arbitrary point $\vec{p} = (p_1, \ldots, p_d)$, we define

$$dist_{\vec{s}}(\vec{t}, \vec{p}) = \sqrt{\sum_{i=1}^{d} (s_i(p_i - t_i))^2}. \qquad (1)$$

The label for $\vec{p}$ is given by $\ell_{\vec{p}} = e^{-dist_{\vec{s}}(\vec{t},\vec{p})^2}$. Observe that when the weighted Euclidean distance (based on $\vec{s}$) between $\vec{p}$ and the target is 0, then the label is 1, and as the weighted Euclidean-distance between $\vec{p}$ and the target approaches ∞, the label approaches 0. Furthermore, the label decays exponential with respect to the distance. For concept learning in which boolean labels are expected, we say that $\vec{p_i}$ is positive if and only if $\ell_{p_i} >= 0.5$.

In standard supervised learning, the learner would receive the label for each point:

$$\langle (\vec{p}_1, \ell_{p_1}), (\vec{p}_2, \ell_{\vec{p}_2}), \ldots, (\vec{p}_m, \ell_{\vec{p}_m}) \rangle$$

For example, consider an application to CBIR in which a global feature vector such as a color histogram is used. In this setting, each image would be represented as a single point. For this situation, $\vec{t}$ is the feature vector corresponding to the ideal value for each color in the histogram and $\vec{s}$ as the ideal weighting of the features. Here each image is a single point with a label of positive (if the user likes the image) or negative otherwise. Such an approach captures how relevance feedback is used for re-weighting in standard CBIR. If there is such a target concept so that all desirable images are closer to $\vec{t}$ (when weighted by $\vec{s}$) than all undesirable images, then color histogramming will yield good results.

[3]We use the standard terminology of the field, in which a bag of points is a set, not a mathematical bag, of points.

We now define the MI model. Here the target is still defined by $\vec{t}$ and $\vec{s}$, and the label for an individual point $\vec{p}$ is as in the standard supervised learning model. The difference in MI learning is that the examples are not single points but rather bags of points. More formally, the training data is $\{\langle B_1, \ell_1\rangle, \langle B_2, \ell_2\rangle, \ldots \langle B_m, \ell_m\rangle\}$ where ℓ_i is the label of bag B_i. $B_i = \{\vec{p}_{i,1}, \vec{p}_{i,2}, \ldots \vec{p}_{i,|B_i|}\}$. Let $\ell_{i,j}$ be the label of point $\vec{p}_{i,j} \in B_i$. Then $\forall i \ \ell_i = \max\{\ell_{i,1}, \ell_{i,2}, \ldots, \ell_{i,|B_i|}\}$. In standard supervised learning, the learning algorithm receives the label of each point, whereas in multiple-instance learning, the learner only receives labels of the bags. (This distinction is illustrated in Figure 3.3.)

Observe that the assumptions of the MI model match the requirements of Localized CBIR when using the representation described in Section 3.1. In the MI model each example is a bag of points, and we represent each image as set of points with one point for each segment. In a desirable image, many or most of the points represent irrelevant content – content that is not interesting to the user. However, we assume that at least one point is of interest (i.e. is similar to the target). In images that were labeled as negative in the query image set, all of the points are assumed to not be similar to the target.

3.4 EMDD

In order to describe EMDD, the MI algorithm used to learn our hypothesis, we first must describe the Density (DD) algorithm [11] upon which EMDD is based. The diverse density at a point $\vec{p}$ in the feature space is a probabilistic measure of both how many *different* positive bags have an instance near $\vec{p}$, and how far the negative instances are from $\vec{p}$. Intuitively, the diversity density of a hypothesis $\vec{h}$ is just the likelihood (with respect to the data) that $\vec{h}$ is the target. A high diverse density indicates a good candidate for a “true” concept. They then add a scale factor (weighting) for each dimension, and use a gradient search (with multiple starting values) to find the point that maximizes the diverse density.

EMDD views the knowledge of which instance corresponds to the label of the bag as a missing attribute and applies the Expectation-Maximization (EM) algorithm of Dempster, Laird, and Rubin [3] to convert the multiple-instance learning problem to a standard supervised learning problem. EMDD starts with some initial guess of a target point $\vec{h}$ and scale vector $\vec{s}$, and then repeatedly performs the following two steps. In the first step (E-step), the current $\vec{h}$ and $\vec{s}$ are used to pick one instance from each bag which is most likely (given the generative model) to be the one responsible for the label. In the second step (M-step), a single-instance variation of the diverse density algorithm is used. These steps are repeated until the algorithm converges. EMDD is started at an initial value based on every point $\vec{p}$ from three randomly selected positive bags with initial values of $\vec{h} = \vec{p}$ and $\vec{s} = \vec{1}$.

3.5 Ranking Algorithm

The final stage of A*cc*io! is to take the set of hypotheses returned by EMDD and combine them to rank the images in the image repository. The ranking of the image repository depends on how the hypotheses of EMDD are used. While the original EMDD algorithm proposed directly using the hypothesis with the highest diverse density value, extensive experiments with CBIR data sets have shown statistically significant improvement in retrieval accuracy when retaining all the hypotheses [18, 13].

The final similarity measure for an image in the repository is a geometric average among the similarity measure given by each hypothesis. Let $\mathcal{H} = \{(\vec{h}_1, \vec{s}_1), \ldots, (\vec{h}_k, \vec{s}_k)\}$ be the set of hypotheses returned by EMDD. Let I be an image in the image repository that is segmented into r segments and represented by test the bag $B = \{\vec{p}_1, \ldots, \vec{p}_r\}$. The Hausdorff distance between hypothesis $H_i = (\vec{h}_i, \vec{s}_i)$ and B is given by

$$d(H_i, B) = \min_{j=1,\ldots,r} dist_{\vec{s}_i}(\vec{h}_i, \vec{p}_j)$$

where *dist* is the weighted Euclidean metric given in Equation (1). The label for bag B is given by

$$\ell_B = \frac{1}{k} \cdot \prod_{i=1}^{k} e^{d(H_i,B)^2}.$$

When the hypotheses are unnormalized, the method perform better than any variation in our experiments of using a single hypothesis or multiple hypotheses. Finally, the images are ranked in decreasing order based on the label.

This ranking method works well since a good hypothesis is one that occurs very close to a point in a positive test bag. The label drops exponentially with relation to the distance of the hypothesis from a point in the bag. Thus a good hypothesis will give a label that is several magnitudes larger than a mediocre hypothesis. A good hypothesis will reflect more heavily in the average than many mediocre hypotheses. Furthermore, when the scale vectors (i.e. $\vec{s}$) of the hypotheses are unnormalized, the sparsity of the points in the hypothesis space causes the probability of a negative bag being falsely predicted as positive to become very low. Hence, accuracy in label prediction is increased.

A second conjecture as to why average labeling works well is that even in localized CBIR, the desired images may belong to a broad image domain where no single hypothesis can capture what images the user desires. For example, when looking for a mountain, the whiteness of snow neighboring the course texture of rock is one way to represent a mountain. The gray color of rock neighboring the smooth blue of the sky is another representation. Each of these are represented by a different hypothesis. By combining all hypotheses, A*cc*io! can capture such disjunctive concepts. Thus, both an image with snow neighbored by rock and an image with rock neighbored by sky will be ranked highly. The combining of all hypothesis enables A*cc*io! to generalize to broad retrieval tasks such as image scenes. Global content in image scenes would be captured by A*cc*io! piecewise by each hypothesis. Then test images containing the most similar set of segments will be ranked the highest.

4. EXPERIMENTAL RESULTS

We first report on results using a natural scene images from the Corel image suite. There are 5 image categories (mountain, sunset, waterfall, field and flower) in our data sets. We refer to the task of learning to distinguish mountain from non-mountain images as the *mountain task* and likewise for the other image groups. The results we report are all based on a test data set of 900 examples in which there are 180 positive examples (i.e. from the desired image group) and 720 negative examples (180 from each of the remaining four image groups). We used 30 fold cross-validation with replacement for all of our experiments.

The results from a sample query is shown in Figure 5. The natural scene database we used is strongly characterized by high variation between images within any category. There exists a significant amount of irrelevant features (color and texture) in the data. Finally, the categories are not disjoint. For example, some waterfall images are of a waterfall in the mountains. In the mountain task, if images of waterfalls are not provided as negative examples (as they are in the query of Figure 5), then some waterfalls that occur in rocky areas often rank highly in the mountain task. However, for the mountain task, if such a waterfall image is classified as a mountain it is treated as a wrong prediction since we label each image based solely on the category given by Corel.

We use Maron and Ratan's representation and hypothesis selection scheme as a baseline [12]. However, we use the improved EMDD algorithm (as opposed to the DD algorithm), because of its empirical performance gains in both retrieval accuracy and time [18]. We refer to this bag generator as "SBN" for single-blob with neighbors. We compare this to the results from A*cc*io! when using k-means as a segmentation algorithm (for $k = 8$), IHS-8 (which produces 8 segments per image), IHS-16, and IHS-32. Also, to show the gains obtained by including the neighboring information, we also include IHS-16 (no nbrs) that represents each segment as 6-dimensional point (leaving out the 24 features used for neighbors). These results are given in Table 1. Note that when using a segmentation algorithm like IHS in which the segments are contiguous a larger number of segments is needed than with k-means. As an example imagine there is a blue sky with four non-connected clouds in the sky. To represent this part of the image with k-means only requires two segments (since all clouds can be a single segment even

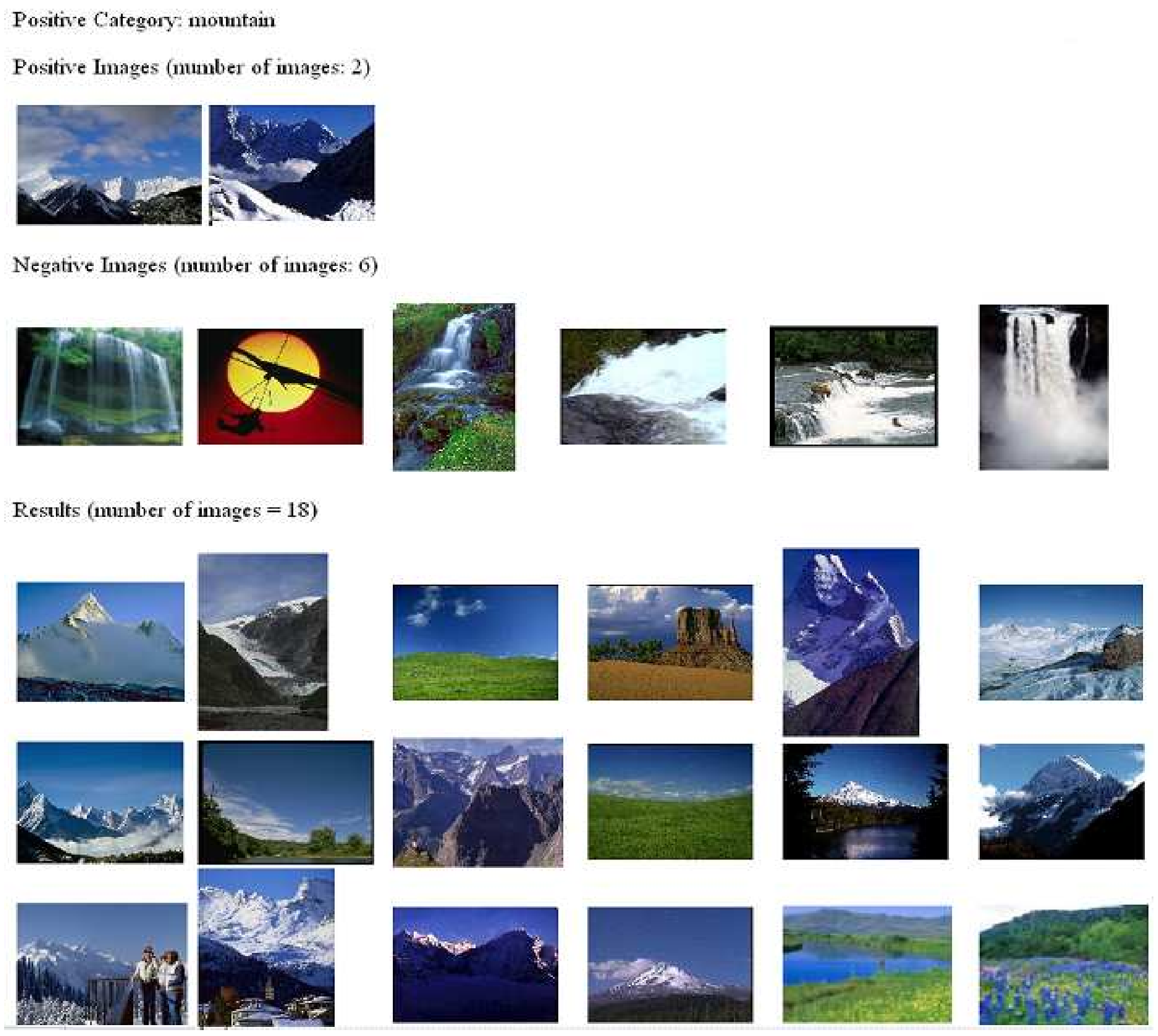
Positive Category: mountain
Positive Images (number of images: 2)
Negative Images (number of images: 6)
Results (number of images = 18)

Figure 5: Result from a sample query for the mountain task.

though they are not contiguous). For IHS, five segments would be needed.

[[[Reviewers: We are still completing our experiments on 25-category image set and the remainder of this section will report those results. We plan on comparing the performance of our algorithm to a variant of Simplicity [16] in which we give each image in the query image set to Simplicity as the query image. (We also hope to add a comparison to simplicity for the natural scene image set.) Simplicity uses a segmentation algorithm in conjunction with a sophisticated matching algorithm to define a similarity measure between the query image and all images in the image repository. Thus for each image in the repository there is a similarity measure defined based on each image in the query image set. For each image in the repository, we compute the average similarity with respect to the positive images in the query set and subtract from this the average similarity with respect to the negative images in the query set. The images are then ranked based on this composite similarity measure.]]]

5. CONCLUSIONS

We have presented, A**c***cio*! , a localized CBIR system that does not assume that the desired object(s) are in a fixed location or have a fixed size. In addition, our system incorporates information about the relationship of an object to its neighboring objects and combines the results from several hypothesis to obtain a robust system that can handle a wide variety of image sets. In addition, we have provided a benchmark data set for localized CBIR and an evaluation framework that will allow systematic comparisons to be made between different systems.

[[[Reviewers: We are also still writing this section since it will discuss the results. We will also discuss future research directions. One important direction is to make comparisons with other CBIR systems that are designed to use Relevance

Task	Bag Generator	**Training Set Size** ($\#pos, \#neg$)			
		4 (2,2)	8 (2,6)	8 (4,4)	16(4,12)
Sunset	SBN	73.6 ± 1.6	72.9 ± 1.6	75.7 ± 1.8	75.8 ± 1.3
	k-means	86.1 ± 2.3	86.8 ± 1.8	88.8 ± 1.3	89.0 ± 1.5
	IHS-8	86.7 ± 2.4	88.5 ± 1.3	90.6 ± 0.6	90.8 ± 0.8
	IHS-16	88.2 ± 1.8	89.8 ± 0.7	91.1 ± 0.7	91.6 ± 0.6
	IHS-16 (no nbrs)	87.7 ± 2.0	89.5 ± 0.8	91.1 ± 0.6	91.2 ± 0.7
	IHS-32	90.4 ± 0.9	89.6 ± 1.4	91.3 ± 0.7	92.3 ± 0.4
Mountain	SBN	74.6 ± 1.7	76.1 ± 1.6	75.4 ± 1.6	77.0 ± 1.7
	k-means	76.6 ± 1.3	77.7 ± 1.3	78.7 ± 0.9	80.7 ± 1.1
	IHS-8	76.5 ± 1.1	76.3 ± 0.8	78.3 ± 0.7	77.3 ± 1.2
	IHS-16	76.9 ± 1.0	75.9 ± 1.2	77.7 ± 0.7	78.1 ± 1.2
	IHS-16 (no nbrs)	75.8 ± 1.1	74.8 ± 1.0	75.9 ± 0.7	76.3 ± 1.0
	IHS-32	77.6 ± 1.1	77.5 ± 1.2	78.7 ± 0.7	79.8 ± 1.0
Waterfall	SBN	75.6 ± 1.7	75.9 ± 1.4	74.8 ± 1.5	75.6 ± 1.8
	k-means	80.8 ± 1.3	78.2 ± 1.3	81.5 ± 1.1	79.6 ± 1.3
	IHS-8	79.0 ± 1.1	78.9 ± 0.9	81.9 ± 0.8	81.2 ± 0.9
	IHS-16	80.6 ± 1.2	80.7 ± 1.2	82.2 ± 0.9	82.7 ± 0.6
	IHS-16 (no nbrs)	80.3 ± 1.2	78.9 ± 1.4	82.5 ± 0.9	81.7 ± 0.6
	IHS-32	79.9 ± 1.4	81.3 ± 1.4	83.8 ± 0.9	82.2 ± 0.7
Flower	SBN	73.2 ± 2.1	76.9 ± 2.3	75.6 ± 2.0	77.7 ± 2.1
	k-means	76.6 ± 1.6	78.0 ± 1.5	79.0 ± 1.3	80.1 ± 1.2
	IHS-8	74.4 ± 1.8	75.7 ± 2.3	76.0 ± 2.0	79.6 ± 1.0
	IHS-16	76.0 ± 1.6	75.5 ± 1.9	78.1 ± 1.9	80.4 ± 1.5
	IHS-16 (no nbrs)	76.7 ± 1.6	74.8 ± 1.9	78.5 ± 1.8	79.9 ± 1.6
	IHS-32	78.3 ± 2.1	79.0 ± 2.1	80.5 ± 1.8	81.2 ± 1.7
Field	SBN	76.8 ± 1.1	76.8 ± 1.7	76.6 ± 1.2	76.9 ± 1.3
	k-means	78.8 ± 2.1	81.4 ± 1.8	84.0 ± 1.7	83.1 ± 1.5
	IHS-8	79.0 ± 2.0	79.2 ± 1.8	80.9 ± 1.3	82.6 ± 1.5
	IHS-16	81.5 ± 1.7	80.8 ± 2.0	82.8 ± 1.5	85.1 ± 1.3
	IHS-16 (no nbrs)	79.8 ± 1.3	79.3 ± 1.8	80.9 ± 1.2	82.4 ± 1.4
	IHS-32	82.5 ± 1.5	83.3 ± 1.5	84.0 ± 1.2	84.1 ± 1.3

Table 1: Results for natural scenes image repository. Each entry shows the mean accuracy (when the top ranked 180 images are labeled as positive and the rest of negative), and the width of the error bar for the 95%-confidence interval.

Feedback.]]]

6. ACKNOWLEDGMENTS

We would like to thank John Krettek, Sharath Cholleti, and Ibrahim Noorzaie for all of their valuable feedback and ideas. This material is based upon work supported by the National Science Foundation under Grant No.0329241.

7. ADDITIONAL AUTHORS

Additional authors: Jason Fritts (Department of CSE, Washington University, email:`fritts@wustl.edu`).

8. REFERENCES

[1] S. Andrews, T. Hofmann, and I. Tsochantaridis. Multiple instance learning with generalized support vector machines. *Artificial Intelligence*, pages 943–944, 2002.

[2] I. Cox, M. Miller, S. Omohundro, and P. Yianilos. Pichunter: Bayesian relevance feedback. *International Conference on Pattern Recognition*, pages 361–369, 1996.

[3] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistics Society*, pages 1–38, 1977.

[4] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, pages 31–37, 1997.

[5] T. Gevers and A. Smeulders. *Emerging Topics in Computer Vision*, chapter Image Search Engines: An Overview. Prentice Hall, 2004.

[6] J. Hartigan and M. Wong. Algorithm as136: a k-means clustering algorithm. *Applied Statistics*, pages 100–108, 1979.

[7] X. He, W.-Y. Ma, O. King, M. Li, and H. Zhang. Learning and inferring a semantic space from user's relevance feedback for image retrieval. *ACM Multimedia*, 2002.

[8] X. Huang, S.-C. Chen, M.-L. Shyu, and C. Zhang. User concept pattern discovery using relevance feedback and multiple instance learning for content-based image retrieval. *8th Int. Conf. on Knowledge Discovery and Data Mining*, pages 100–108, 2002.

[9] S. Kim, S. Park, and M. Kim. Central object extraction for object-based image retrieval. *2nd Int. Conf. Image and Video Retrieval*, pages 39–49, 2003.

[10] F. Long, H. Zhang, and D. Feng. *Multimedia Information Retrieval and Management- Technological Fundamentals and Applications*, chapter Fundamentals of Content-Based Image Retrieval. Springer, 2003.

[11] O. Maron and T. Lozano-Pérez. A framework for multiple-instance learning. *Neural Information Processing Systems*, 1998.

[12] O. Maron and A. Ratan. Multiple-instance learning for natural scene classification. *15th Int. Conf. on Machine Learning*, pages 341–349, 1998.

[13] R. Rahmani, S. Goldman, H. Zhang, and J. Fritts. Content based image retrieval using multiple instance learning. Technical report, Washington University in St Louis, 2005.

[14] Y. Rui, T. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: A power tool in interactive content-based image retrieval. *IEEE Transactions Circ. Sys. Video Technology*, pages 644–655, 1998.

[15] Q. Tian, Y. Wu, and T. S. Huang. Combine user defined region-of-interest and spatial layout for image retrieval. *Int. Conf. on Image Processing*, 2000.

[16] J. Wang, J. Li, and G. Wiederhold. SIMPLIcity: semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 947–963, 2001.

[17] H. Zhang and J. Fritts. Improved hierarchical segmentation. Technical report, Washington University in St Louis, 2005.

[18] Q. Zhang, S. Goldman, W. Yu, and J. Fritts. Content base image retrieval using multiple-instance learning. In *19th International Conference on Machine Learning*, pages 682–689, 2002.

[19] Q. Zhang and S. Goldman. EM-DD: An improved multiple-instance learning technique. *Advances in Neural Information Processing Systems 14* (Proceedings for NIPS 2001), pages 1073–1080, 2002.