Open access • Proceedings Article • DOI:10.1145/1031171.1031213

Image similarity search with compact data structures — Source link

Qin Lv, Moses Charikar, Kai Li

Institutions: Princeton University

Published on: 13 Nov 2004 - Conference on Information and Knowledge Management

Topics: Image retrieval, Similarity (network science), Feature (computer vision), Nearest neighbor search and Feature vector

Image Similarity Search with Compact Data Structures

Qin Lv
Dept. of Computer Science
Princeton University
qlv@cs.princeton.edu

Moses Charikar
Dept. of Computer Science
Princeton University
moses@cs.princeton.edu

Kai Li
Dept. of Computer Science
Princeton University
li@cs.princeton.edu

ABSTRACT

The recent theoretical advances on compact data structures (also called "sketches") have raised the question of whether they can effectively be applied to content-based image retrieval systems. The main challenge is to derive an algorithm that achieves high-quality similarity searches while using compact metadata. This paper proposes a new similarity search method consisting of three parts. The first is a new region feature representation with weighted L_1 distance function, and EMD* match, an improved EMD match, to compute image similarity. The second is a thresholding and transformation algorithm to convert feature vectors into very compact data structures. The third is an EMD embedding based filtering method to speed up the query process. We have implemented a prototype system with the proposed method and performed experiments with a 10,000 image database. Our results show that the proposed method can achieve more effective similarity searches than previous approaches with metadata 3 to 72 times more compact than previous systems. The experiments also show that our EMD embedding based filtering technique can speed up the query process by a factor of 5 or more with little loss in query effectiveness.

Categories and Subject Descriptors:
H.3.3 [Information Storage and Retrieval]: [Information Search and Retrieval]

General Terms: Algorithms, Design, Performance

Keywords:
image similarity, search, compact data structures

1. INTRODUCTION

During the past two decades, storage density has been doubling about every 18 months, improving at the rate of Moore's law. If this trend continues, the capacity of a single disk will reach 1 terabyte in 2007 and 1 petabyte in 2022 [16]. Such storage capacity will allow users to store hundreds of millions of images on a single disk. In order to manage such large-scale image collections efficiently, it is becoming increasingly important that image retrieval systems use highly compact data structures.

The main challenge of using compact data structures in image retrieval is to devise methods to create and use highly compact data structures while achieving high image retrieval quality. The recent progress of theoretical studies on *sketches* has motivated us to investigate how to achieve high-quality and efficient similarity searches of image data with highly compact data structures.

In this paper, we propose a new image similarity search method with three components, addressing effectiveness, compactness, and efficiency issues for whole-image search in large-scale image collections:

- EMD* Match: a more effective region-based image similarity measure that uses distance thresholding and adjusted region weights when computing Earth Mover's Distance(EMD).
- Compact Data Structures: a thresholding and transformation algorithm is designed to convert real-valued feature vectors into very compact data structures. The weighted L_1 distance (with thresholding) of feature vectors can be computed very efficiently using the compact data structures.
- Filtering: an approximate EMD embedding algorithm is proposed to embed the EMD of region vectors into sketches for images. The sketches can be used to filter out different images effectively and efficiently, reducing query time substantially.

We have implemented the proposed method in a prototype system and evaluated the method with an image database with predefined sets of similar images. Our experimental results show that the proposed method is quite effective. The EMD* match is 17% - 76% more effective than other region-based image similarity measures. The thresholding and transformation algorithm can compact feature vectors into metadata 3 to 72 times smaller than the region representations of previous systems while achieving high-quality similarity searches. Finally, we show that our filtering algorithm based on approximate EMD embedding can speed up the search time by a factor of 5 or more with little loss in effectiveness.

The rest of the paper is organized as follows: We investigate related work in Section 2. An overview of our method is given in Section 3. We then explain our region representation and image similarity measure in Section 4. Section 5

CIKM'04, November 8–13, 2004, Washington, DC, USA.

explains in details our thresholding and transformation algorithm that generates very compact data structures. The filtering algorithm based on approximate EMD embedding is described in Section 6. Experimental results are presented in Section 7. Finally, Section 8 concludes the paper.

2. RELATED WORK

2.1 Compact Data Structures

Much work has been done on compact data representations. The recent theoretical focus is to represent data compactly via a *sketch* such that the sketch can be used to estimate some function on the original data. For example, a distance function on pairs of data items could be estimated by only examining the sketches of the data items. The existence of such a sketch depends crucially on the function we wish to estimate. If such a sketching technique exists, it gives a significant saving in storage space (since the sketches are much smaller than the original data items) as well as running time. Sketch constructions have been developed for a number of purposes, including estimating set membership [5], estimating similarity of sets [6], estimating distinct elements and vector norms [1, 12], as well as estimating string edit distance [4]. It is shown in [8] how sketch constructions can be derived from rounding techniques used in approximation algorithms. Many of these sketch constructions for estimating similarity and distances can be viewed as embeddings (approximate distance preserving mappings) from the data points to points in a normed space (usually L_1 or L_2). Once such a mapping is obtained, known sketching techniques for L_1 or L_2 can be applied.

We are interested in constructing mappings of images to bit vectors with the Hamming distance as the distance measure (since Hamming distance of bit vectors can be computed very efficiently). The Earth Mover's Distance (EMD) [22] has been proposed before as a useful metric for image retrieval. It has been shown in [8, 14] how EMD on sets of points in d-dimensional Euclidean space can be mapped to L_1 with some approximation. However the quality of approximation deteriorates significantly with the number of points and the dimension d of the point set. The work of Indyk and Thaper [14] is related to our compact representation scheme for images. We note however, two crucial differences: 1) Their technique applies to EMD on points in Euclidean space. As we discuss later, we will need to use EMD on a thresholded version of normed distance for which new techniques are needed. 2) Their methods are designed for low dimensional point sets since the performance deteriorates with dimension d; in contrast, we need to compute EMD on sets of high dimensional bit vectors. Nevertheless, we will use some of the ideas of [14] in devising our final embedding of images into Hamming space. We elaborate on this in Section 6.

2.2 Image Similarity Search

Many techniques have been proposed for image similarity search, as surveyed in [10, 23, 24, 27]. A comprehensive survey of existing work is outside the scope of this paper. Instead, we examine region-based image retrieval (RBIR) methods that are most related to our work, focusing on region representation and region-based image similarity measures.

2.2.1 Region Representation

Most RBIR systems use a combination of color, texture, shape, and spatial information to represent a region. Blobworld [7] represents each region by a 218-bin color histogram, mean texture contrast and anisotropy, centroid, area, eccentricity and orientation, which is a very complicated representation. NETRA [18] also uses a complicated region representation. It quantizes the RGB color space into 256 colors, and each region's color is represented by $\{(c_1, p_1), \ldots, (c_n, p_n)\}$, where c_i is the color code and p_i is the fraction of that color in the region. Texture is represented by normalized mean and standard deviation of a set of Gabor wavelet transformations with different scales and directions. VisualSEEk [25] extracts salient color regions using a back-projection technique and supports joint color-spatial queries. A selection of 166 colors in the HSV color space are used. Each region is represented by a color set, region centroid, area, width and height of the minimum bounding rectangle. WALRUS [21] segments each image by computing wavelet-based signatures for sliding windows of various sizes and then clusters them based on the proximity of their signatures. Each region is then represented by the average signature. Windsurf [2, 3] performs 3-level Haar wavelet transformation in the HSV color space and the wavelet coefficients of the 3rd level LL subband are used for clustering. Each region is represented by its size, centroid and corresponding covariance matrices. SIMPLIcity [28] partitions an image into 4x4 blocks and computes average color and wavelet coefficients in high frequency bands.

Table 1 compares the region representation (with estimated sizes) and region distance function used in some RBIR systems. As we can see, most of these systems use complicated region representations and some of them also use distance functions that are expensive to compute. Our method uses much more compact data structures to represent each region and region distance can be calculated very efficiently by XORing operations.

2.2.2 Region-Based Image Similarity Measure

Current region-based image similarity measures can be roughly divided into three categories:

- Independent best match: Systems such as Blobworld and NETRA find the best matched region for each query region and calculate the overall similarity score using fuzzy-logic operations or weighted sum. Since each query region is matched independently, multiple regions in the query image might be matched to the same region in a target image, which is undesirable in many cases. As an extreme example, consider an image A full of red balloons and a very different image B with a red ball in it. Since each red balloon in A matches the red ball in B very well, these two images will be considered very similar by independent best match.

- One-to-one match: Systems like Windsurf and WALRUS consider matching one set of regions to another set of regions and require that each region can only be matched once. For example, Windsurf uses the Hungarian Algorithm to assign regions based on region distance. Region size is then used to adjust two matching regions' similarity. Image similarity is defined as the sum of the adjusted region similarity. One-to-One

System	Region Representation	Distance Function	Estimated Size per region
Blobworld	218-bin color histogram contrast, anisotropy centroid, area, eccentricity, orientation	quadratic Euclidean Euclidean	900 bytes
NETRA	$\{(c_1, p_1), \ldots, (c_n, p_n)\}$ $\{\mu_{0,0}, \sigma_{0,0}, \ldots, \mu_{s-1,k-1}, \sigma_{s-1,k-1}\}$ f_K(32-D), f_R(32-D), f_Z(64-D)	$O(n^2)\times$Euclidean Euclidean Euclidean	$(n = 7)$ $(s = 4, k = 6)$ $\Rightarrow$ 739 bytes
VisualSEEk	166-bit color vector centroid, area width, height of bounding box	quadratic (modified for bit vector) Euclidean, L_1 L_2	41 bytes
Windsurf	area 12-D centroid (4 subbands, 3-D color space) 24-D covariance matrices	Bhattacharyya metric	148 bytes
SIMPLIcity	average color (L, U, V) sqrt of 2nd-order moments, in 3 subbands normalized inertia of order 1 to 3	$\sum_i w_i(f_i - f_i^{'})^2$ $\sum_j w_j(f_j - f_J^{'})^2$ $\sum_k w_k(f_k - f_k^{'})^2$, then quantized	36 bytes

Table 1: Comparison of different RBIR systems' region representation, distance function and estimated size.

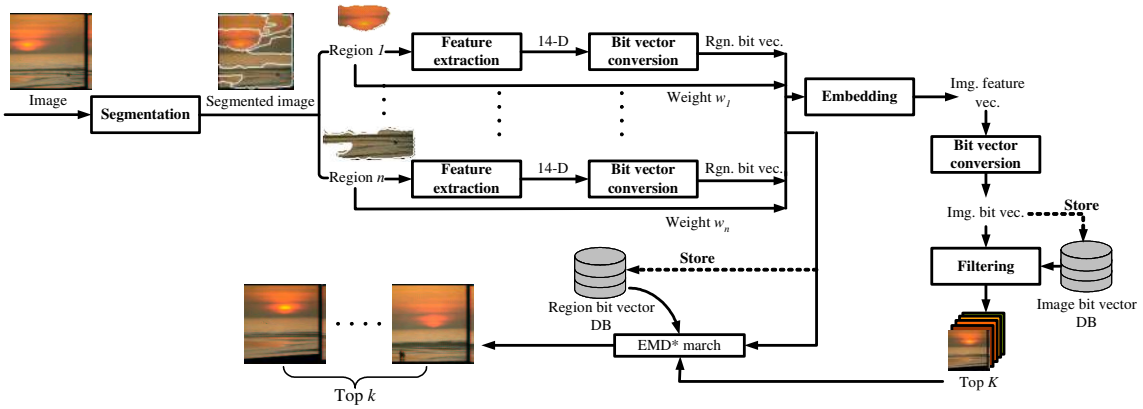

Figure 1: Method overview: main components and steps when inserting an image or querying an image.

match assumes good image segmentation so there is good correspondence between two similar images' regions. But current segmentation techniques [9, 20] are not perfect and regions do not always correspond to objects. Moreover, it is hard to define an optimal segmentation, as one image may need different segmentations when comparing to different images [11].

- EMD match[1]: Some systems[11, 15] use similarity measures based on the Earth Mover's Distance (EMD). Although EMD is a good measure for region matching, its effectiveness is closely linked to the underlying distance function used for pairs of regions as well as the weight given to each region. Since these systems directly use the region distance function as the ground distance for EMD and use normalized region size as the region weight, this creates problems such as regions being weighted inappropriately. (We will elaborate on this in Sections 3 and 6). As a result, these systems do not use EMD very well.

[1]The Integrated Region Matching (IRM) proposed in SIMPLIcity is similar to EMD match.

To overcome the issues with the previous approaches, we have developed a compact data representation, an improved EMD, called EMD*, and an EMD embedding based filtering method.

3. OVERVIEW OF OUR METHOD

Figure 1 shows the main components of our image similarity search method and illustrates the steps an image goes through when it is inserted into the system, or is submitted as a query image.

When an image is inserted into the system, we first segment it into several homogeneous regions. For each region, we extract a 14-dimensional feature vector, and then convert it into a bit vector using the thresholding and transformation algorithm. This results in very compact representation of each region, and the distance between two regions can be calculated efficiently by XORing their region bit vectors. Next, all the n region bit vectors along with their weights are embedded into a single image feature vector, such that the L_1 distance on two images' embedded feature vectors approximates the EMD* between these two images. For compactness and efficiency in distance calculation, the im-

age feature vector is also converted into a bit vector. Both the image bit vector and the individual region bit vectors (with region weights) are stored into a database for future image retrieval.

A query image goes through the same process of segmentation, feature extraction, bit vector conversion, embedding, and bit vector conversion. Then the image bit vector is used to do filtering in the image database and obtain the top K images that are closest to the query image's bit vector. We calculate exact EMD* match between the query image and each of the K images using their region bit vectors. Finally the top k images with smallest EMD* match to the query image are returned.

4. REGION REPRESENTATION AND IMAGE SIMILARITY MEASURE

4.1 Region Feature Vector Representation

We represent each region with a simple feature vector that includes two kinds of information about a region: color moments and bounding box information.

Color moments The idea of using color moments to represent a color distribution was originally proposed in [26]. It is a compact representation and it has been shown in [19] that the performance of color moments is only slightly worse than high-dimensional color histograms. We extract the first three moments from each channel in the HSV color space, resulting in a 9-dimensional color vector.

Bounding box is the minimum rectangle covering a region. For each region, we calculate its bounding box and obtain the following information:

$x\_$	bounding box width
$y\_$	bounding box height
$p\_$	#pixels in a region
$r\_ = x\_/y\_$	aspect ratio
$s\_ = x\_ \cdot y\_$	bounding box size
$a\_ = p\_/s\_$	area ratio
(c_x, c_y)	region centroid

We use a 5-dimensional vector to represent a region's bounding box information: $(ln(r\_), ln(s\_), a\_, c_x, c_y)$.

Weighted L_1 distance is used for both the color vectors and the bounding box vectors. We can easily concatenate a region's 9-D color vector and 5-D bounding box vector into a 14-D vector and the distance between two regions is the weighted L_1 distance between their 14-D vector representations.

4.2 Image Similarity Measure: EMD* Match

We design an image similarity measure based on Earth Mover's Distance(EMD), which is a flexible similarity measure between multidimensional distributions. Given two distributions represented by sets of weighted features and a distance function between pairs of features, EMD reflects the minimal amount of work needed to transform one distribution into another by moving distribution "mass" (weights) around. Consider distributions $Q = \{(q_1, w_1^Q), \ldots, (q_m, w_m^Q)\}$ and $R = \{(r_1, w_1^Q), \ldots, (r_n, w_n^R)\}$, where the pair (q_i, w_i^Q) denotes that distribution Q has weight w_i^Q for feature q_i. We will assume that the weights are normalized, i.e. they sum up to 1. Thus, $\sum_i w_i^Q = \sum_j w_j^R = 1$. The EMD between distributions Q and R is computed as follows:

$$EMD(Q,R) = \min \sum_i \sum_j f_{ij} \cdot d(q_i, r_j)$$

where f_{ij} are non-negative and satisfy the conditions:

$$\forall i \quad \sum_j f_{ij} = w_i^Q; \qquad \forall j \quad \sum_i f_{ij} = w_j^R$$

Note that EMD can be computed via (weighted) bipartite matching, but this is a relatively expensive operation.

As mentioned in related work, a couple of RBIR systems have used "EMD match"-based image similarity measures where the region distance function is used as the ground distance of EMD and normalized region size is used as region weight. However, as we explain below, current "EMD match"-based image similarity measures do not use EMD appropriately. In particular, the distance function and region weight information that are inputs to EMD are inappropriate.

The first observation is that a region's importance in an image is not proportional to that region's size. For example, a large region (e.g. front door) usually should not be considered much more important than a small region (e.g. a baby). After considering various region weighting schemes, we decide to use the normalized square root of region size as each region's weight, which reduces the difference between small and large regions, and assigns suitable weights in most segmentation scenarios. We pick weights w_i for regions Q_i such that

$$w_i \propto sqrt(area(Q_i))$$
$$\sum_i w_i = 1$$

A second observation is that similar images may still have very different regions (e.g. the same baby with a different toy). If we simply use the region distance function, two similar images may be considered different only because they have two very different regions. To address this problem, distance thresholding is used after we calculate the distance between two regions. Roughly speaking, if the distance between two regions is larger than a threshold δ, we use δ as the region distance. By setting an upper bound on region distance, we reduce the effect that an individual region can have on the whole image, making our image similarity measure more robust.[2]

Based on the two observations, we define image dissimilarity as the EMD using square root region size as region weight, and thresholded region distance as the ground distance function. We call this measure "*EMD* match*"-based image similarity measure.

5. COMPACT DATA STRUCTURES

In this section, we propose a thresholding and transformation algorithm that approximates weighted (and thresholded) L_1 distance of real-valued feature vectors with Hamming distance of bit vectors. As we can see, the bit vector

[2]Note that this technique is designed for whole image matching instead of partial matching, where the user only wants to match one or more particular regions.

representation is much more compact than the real-valued feature vector representation; and it is also much faster to calculate Hamming distance of bit vectors (XORing bits) than weighted (and thresholded) L_1 distance of feature vectors (floating point operations).

Algorithm 1 Generate $N \times K$ Random (i,t) Pairs

input: $N, K, d, l[d], u[d], w[d]$
output: $p[d], rnd\_i[N][K], rnd\_t[N][K]$

$p_i = w_i \times (u_i - l_i)$; $\quad for\ i = 0, \ldots, d-1$
normalize p_i $\quad s.t.\ \Sigma_{i=0}^{d-1}\ p_i = 1.0$

for $(n = 0; n < N; n++)$ **do**
  for $(k = 0; k < K; k++)$ **do**
    pick random number $r \in [0, 1)$
    find i $\quad s.t.\ \Sigma_{j=0}^{i-1}\ p_i <= r < \Sigma_{j=0}^{i}\ p_i$
    $rnd\_i[n][k] = i$
    pick random number $t \in [l_i, u_i]$
    $rnd\_t[n][k] = t$
  end for
end for

We first describe how to generate bit vectors from d-dimensional vectors such that the expected Hamming distance between two bit vectors produced is proportional to the weighted L_1 distance between the corresponding vectors. In order to do this, we describe how to generate a single bit from each d-dimensional vector such that the probability that the bit produced is different for two vectors is proportional to their weighted L_1 distance. We produce the required bit vectors by repeating this process to produce several bits and concatenating them together. Suppose we want to compute weighted L_1 distance for d-dimensional vectors, where the ith coordinate is in the range $[l_i, h_i]$ and has weight w_i. Let $T = \Sigma_i w_i \times (h_i - l_i)$, and $p_i = w_i \times (h_i - l_i)/T$. Note that $\Sigma_i p_i = 1$. To generate a single bit, pick $i \in [0, d-1]$ with probability p_i, pick a uniform random number $t \in [l_i, h_i]$. For each vector $v = (v_1, \ldots, v_d)$,

$$bit = \begin{cases} 0 & \text{if } v_i < t \\ 1 & \text{if } v_i >= t \end{cases}$$

Algorithm 2 Convert Feature Vector to N-Bit Vector

input: $v[d], N, K, rnd\_i[N][K], rnd\_t[N][K]$
output: $b[N]$

for $(n = 0; n < N; n++)$ **do**
  $x = 0$
  for $(k = 0; k < K; k++)$ **do**
    $i = rnd\_i[n][k]$
    $t = rnd\_t[n][k]$
    $y = (v_i < t\ ?\ 0 : 1)$
    $x = x \bigoplus y$
  end for
  $b_n = x$
end for

Note that an (i,t) pair determines the value of one bit for each vector. To make the transformation consistent across all vectors, for each bit we generate, we must apply the *same* (i,t) pair to each vector. The process of generating (i,t) pairs is described in Algorithm 1. Here, we generate NK such pairs where N is the size of the final bit vector we desire (after thresholding) and K is a parameter which will be determined later.

Next we transform the distance function so the distance is thresholded at a given threshold δ. Algorithm 1 generates NK (i,t) pairs which give rise to N groups of K bits each. We produce a single bit from each group of K bits by applying a hash function to them. The hash function could be XOR, or some other random hash function. We later show that this achieves the thresholding we wanted.

An implementation of the algorithm is shown here. Algorithm 1 is the initializing process, where $N \times K$ random (i,t) pairs are generated. Then for each feature vector, Algorithm 2 is called to convert the feature vector to a N-bit vector.

We now show that in the first step, the expected distance between bit vectors produced is the weighted L_1 distance of the original vectors. This follows from the following lemma and linearity of expectation.

LEMMA 1. *If the weighted L_1 distance between two vectors u and v is x, then the probability that the two vectors generate different bits given the same (i,t) pair is $p = x/T$.*

PROOF. Let $r_i = h_i - l_i$. Given two vectors u and v, let $bit(u)$ and $bit(v)$ be the bits generated for u and v respectively. Let C_i denote the event that coordinate i is picked in the bit generation process. Note that $\Pr[C_i] = p_i = w_i \times r_i/T$. Given that coordinate i is picked, the threshold value t is chosen uniformly in the interval $[l_i, h_i]$. Also the only values of t for which $bit(u) \neq bit(v)$ are $t \in [\min(u_i, v_i), \max(u_i, v_i)]$. Hence,

$$\begin{aligned} \Pr[bit(u) &\neq bit(v)|C_i] = |u_i - v_i|/r_i \\ \Pr[bit[u] \neq bit(v)] &= \sum_{i=0}^{d-1} \Pr[bit(u) \neq bit(v)|C_i] \times \Pr[C_i] \\ &= \sum_{i=0}^{d-1} w_i \times |u_i - v_i|/T = x/T \end{aligned}$$

□

We now analyze the thresholding process we described. For two vectors u and v, we express the probability that the bit produced (after hashing a group of K bits) is different for u and v in terms of the weighted L_1 distance between u and v.

LEMMA 2. *If the weighted L_1 distance between two vectors u and v is x, and if XOR is used as the hashing function, then the probability that the bit generated after hashing the K bits is different for u and v is $q = 0.5(1 - (1 - 2x/T)^K)$.*

PROOF. Let $p = x/T$ be the probability that the jth bit generated for vectors u and v is different. Note that the XOR of the K bits generated is different iff an odd number of the K bits generated are different. Hence, the probability q that the XOR of the K bits generated is different for vectors u

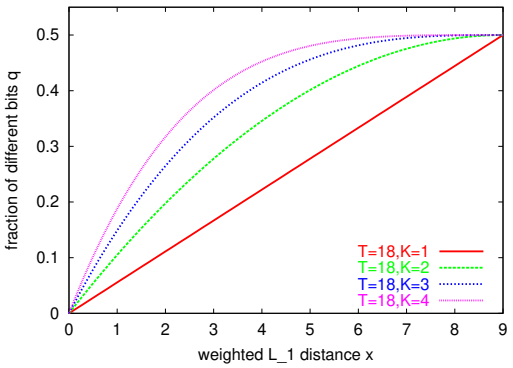

Figure 2: q as a function of x for different K

and v is given by the expression

$$\begin{aligned}
&\sum_{odd\ j}\binom{K}{j}p^j(1-p)^{K-j}\\
=\ &\frac{1}{2}\sum_j\binom{K}{j}p^j(1-p)^{K-j}\\
&-\frac{1}{2}\sum_j(-1)^j\binom{K}{j}p^j(1-p)^{K-j}\\
=\ &\frac{1}{2}(1-(1-2p)^K)=0.5(1-(1-2x/T)^K)
\end{aligned}$$

□

Figure 2 shows the fraction of different bits q as a function of the weighted L_1 distance x for different K values. The T value here is 18. When $K=1$, q increases linearly with x. When $K>1$, we can see that the function is approximately linear initially and then flattens out. The larger the K value, the earlier it starts to "bend". If we want the flattening to happen at around δ, then we should pick K to be about $T/2\delta$.

6. FILTERING

In order to perform similarity searches on a large image database, we propose a filtering method via approximate EMD embedding. The goal is to find a small candidate image set for the EMD* match by filtering out most of the images which are very different from the query image. The challenge is to quickly find a candidate image set that contains most of the similar images.

Previous filtering methods do not work well. The first kind of filtering is to index individual regions and combine the filtering results of all the regions to form the candidate image set. This approach is not effective, because it loses the information of image-level similarity. The second kind is to use a technique to embed EMD into L_1 distance and then use Locality Sensitive Hashing (LSH) to find the nearest neighbor(s) in the latter space [14]. This method has interesting provable properties, but it does not work well with compact data structures nor does it consider distance thresholding on real-valued vectors.

We have designed a new EMD embedding technique that converts a set of region bit vectors into a single image feature vector, and the L_1 distance on the embedded image feature vector approximates the EMD on the original region bit vectors.

The basic step involves picking several random positions $(p_1,\ldots,p_n)$ and checking for a particular bit pattern $(b_1,\ldots,b_n)$ at these positions. Given an image

$$I=\{(r_1,w_1),\ldots,(r_k,w_k)\}$$

where r_i is the bit vector for the ith region and w_i is its weight, and a random pattern

$$P=\{(p_1,b_1),\ldots,(p_h,b_h)\}$$

where $p_j\in 0,N-1$ and $b_j\in 0,1$, we say region r_i fits pattern P if

$$r_{i,p_j}=b_j \qquad for\ j=1,2,\ldots,h$$

Here r_{i,p_j} denotes the p_jth bit of vector r_i. We define the *matched weight* of image I wrt. pattern P as the sum of the weights of the regions in image I that fit pattern P:

$$MW(I,P)=\sum_i w_i \qquad \forall i \text{ st. } r_i \textit{ fits pattern } P$$

In the example below, if we pick random positions 3, 5 and 7, and random bit pattern "011", both r_1 and r_3 fit the pattern (shown in bold numbers), so the matched weight is $0.1+0.3=0.4$.

	1	2	**3**	4	**5**	6	**7**	8	w_i
r_1	1	0	**0**	1	**1**	0	**1**	0	**0.1**
r_2	0	0	1	1	0	1	1	0	0.6
r_3	0	1	**0**	0	**1**	0	**1**	1	**0.3**
MW									**0.4**

Intuitively, if two region vectors are similar, they have more bits in common than other regions. So there is a higher chance that two similar regions both fit (or not fit) a random pattern. Given two similar images, each random pattern picks out the regions in the two images that are similar, in effect matching similar regions to each other. If two images are similar, we expect their matched weight wrt. a random pattern to be close to each other. We will obtain a vector for every image by listing the matched weights for a number of randomly chosen patterns, and distances between images will be computed by L_1 distances between these image vectors. When sufficiently many random patterns are used to generate the image vectors, we expect the L_1 distance between image vectors to be able to distinguish between similar and dissimilar images.

Algorithm 3 Generate M H-bit Random Patterns

```
input:  M, H, N (region bit vector length)
output: P[M][H], B[M][H]

for (i = 0; i < M; i++) do
  for (j = 0; j < H; j++) do
    pick a random position p ∈ [0, N − 1]
    pick a random bit b ∈ {0, 1}
    P[i][j] = p
    B[i][j] = b
  end for
end for
```

We comment on the relationship of our techniques to the EMD embedding proposed by [14]. Our techniques are

designed for distributions on high dimensional bit vectors, while their method is described for distributions of points in R^d, where d is small. Roughly, they decompose the space into collections of disjoint d-dimensional cubes. In fact they have a hierarchy of decompositions for different granularities. For each cube in this decomposition, they calculate the weight of the distribution that falls into this cube and build a vector by listing these counts (suitably weighted). Since we are working with high dimensional bit vectors, there is no decomposition into cubes that we can use. In our technique, the idea of computing the matched weight for a random pattern is analogous to computing the weight that falls into a cube. The embedding in [14] uses different levels of granularity and the weights assigned to them are exponentially decreasing. This creates problems when sampling coordinates to estimate weighted L_1 distance by hamming distance of compact bit vectors; the problem is that the random variables involved have high variance. Our scheme can be thought of as using only one level of granularity and this is designed to get around this problem with using many different levels.

The implementation of the embedding algorithm is divided into two pieces. The first is Algorithm 3 which generates M sets of random positions and picks a random bit pattern for each set.

The second piece is Algorithm 4 which, given an image represented by a list of region bit vectors and their corresponding weights, computes its EMD embedding using the random patterns generated by Algorithm 3,

After the embedding, each image is represented by a M-dimensional real-valued vector. We further convert it to a bit vector using the same algorithm we proposed for converting region feature vectors to region bit vectors (see Section 4). As a result, each image is now represented by a compact bit vector and the Hamming distance between two images can be efficiently computed by XORing their bit vectors. Our filtering algorithm ranks images based on the Hamming distance of their embedded image bit vectors to the query image's bit vector and return the top K images for exact EMD computation.

Algorithm 4 Image EMD Embedding

```
input:  k, r[k][N], w[k], M, H, P[M][H], B[M][H]
output: MW[M]

for (i = 0; i < M; i++) do
  mw = 0.0
  for (j = 0; j < k; j++) do
    h = 0
    while (h < H) && (r[j][P[i][h]] == B[i][h]) do
      h++
    end while
    if h == H then
      mw = mw + w[j]
    end if
  end for
  MW[i] = mw
end for
```

7. EXPERIMENTAL RESULTS

We are interested in answering the following questions:

- How effective is our EMD* match compared to other region-based image similarity measures?
- How compact can the data structure be in order to achieve high-quality similarity searches?
- How effective and efficient is our embedding and filtering algorithm?

This section first describes our evaluation methodology and then presents our experimental results to answer each of the questions above.

7.1 Evaluation Methodology

Image Collection

We have chosen an image collection[3] consisting about 10,000 general-purpose images. The main reason for choosing this image database is that a group of researchers defined 32 sets of similar images for this image collection [4]. These sets represent different categories and have different number of similar images in each set. Our experiments use these 32 sets of similar images as the ground truth in our evaluation.

We have excluded black and white images because our implementation is designed for color image retrieval. The result image database has 9877 images. The image segmentation tool we use is JSEG [9], and the tool's default parameter setting is used. In total, 70702 regions are generated from these images. The average number of regions per image is 7.16, with the minimum being 1 and the maximum being 57. Figure 3 shows the cumulative distribution (CDF) of the number of regions, where the x axis (in log scale) is the number of regions, and the y axis is the percentage of images with at most x regions.

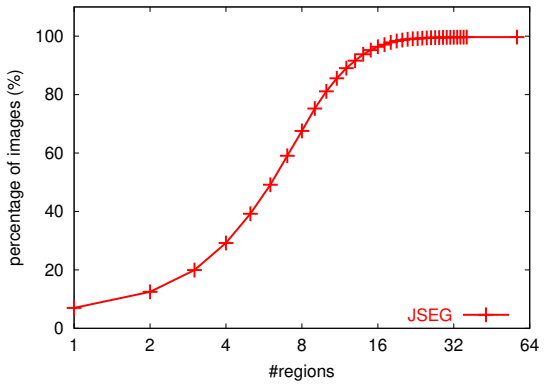

Figure 3: Cumulative distribution function (CDF) of #Regions in the segmented images.

Effectiveness Measure

The goal is to measure how effective a retrieval system is in finding similar images. We have chosen an effectiveness measure called *average precision*. Given a query q with k relevant items while query q is excluded from the relevant

[3]http://wang.ist.psu.edu/docs/related/.

[4]http://dbvis.inf.uni-konstanz.de/research/projects/SimSearch/effpics.html

set, let $rank_i$ be the rank of the ith retrieved relevant item ($1 \le i \le k$), then *average precision* is defined as follows:

$$avg.\ precision = \frac{1}{k}\sum_{i=1}^{k}\frac{i}{rank_i}$$

This measure incorporates information about the rankings of items in the returned results, which is more suitable for image retrieval. Also, it is a single-valued measure, making it much easier to compare the effectiveness of different systems.

This measure is more suitable than *Precision* and *recall* which are widely used as effectiveness measures in information retrieval. *Precision* and *recall* are defined as:

$$\begin{aligned} precision &= |A \cap R| \ / \ |A| \\ recall &= |A \cap R| \ / \ |R| \end{aligned}$$

where A is the set of items retrieved by the system (the actual answer), and R is the set of items relevant to this query (the ideal answer). The problem with *precision vs. recall curve* is that it does not directly consider the ranking of each relevant item in the returned results. In addition, the *Precision vs. recall curve* is not a single-valued measure, making it difficult to compare system effectiveness when we perform multiple queries with various relevant set sizes. The average precision measure we use can be interpreted as follows: we consider the smallest prefix of the returned ranked ordering that contains exactly i relevant results and compute the precision at this point. The average precision is the average of the precision values obtained thus for $i = 1, \ldots, k$.

Experimental Setup

All our experiments are done on a PC with a 933MHz Pentium III Coppermine CPU, 1GBytes of memory and two 60GB Maxtor IDE disks. For each experiment, we perform 32 queries and results are averaged over the 32 queries. Since our bit vector transformation algorithm and embedding algorithm is randomized, each experiment involving these two random algorithms is repeated 5-10 times.

7.2 Effectiveness of EMD* Match

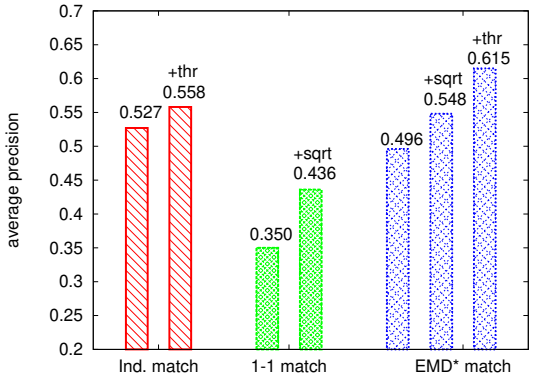

Figure 4: Effectiveness comparison of different region match techniques, using 14-D region feature vectors.

Figure 4 compares the effectiveness of different region matching techniques. All of the matching techniques use the same 14-D region feature vectors proposed in Section 4. When region distance thresholding is used, color moments distance is thresholded at 2.4 and bounding box distance is thresholded at 1.5.

The figure shows that the average precision of the independent best match ("Ind. match") is 0.527. With our proposed thresholding scheme, its average precision increases to 0.558.

To compare with the one-to-one match ("1-1 match"), we tested the image similarity measure proposed by Windsurf, and region matching is done using the Hungarian algorithm. Its average precision is only 0.350 when original region size is used as region weight, and the number goes up to 0.436 when square root of region size is used as region weight. Since Windsurf converts region distance d to region similarity e^{-d/δ_d}, in effect thresholds region distance, it does not benefit from our distance thresholding scheme.

With the original EMD match [5] , the average precision is 0.496. This number improves to 0.548 when we use square root region size as region weight. When region distance thresholding is added, we further increase the number to 0.615. As we can see, the EMD* match technique we proposed is 17% - 76% more effective than other region matching techniques. Both region distance thresholding and square root region size greatly improve the average precision, and other matching techniques also benefit a lot from these two schemes.

We also compare our EMD* match with SIMPLIcity[28]. We obtained the SIMPLIcity code and tested it on the same set of image databases. Since SIMPLIcity did not work for two of the query images, only 30 queries are used when comparing to SIMPLIcity. SIMPLIcity achieves an average precision of 0.331, as compared to EMD* match's 0.629 average precision.

System	Size	Ratio
Blobworld	900 bytes	72
NETRA	739 bytes	59.1
VisualSEEk	41 bytes	3.3
Windsurf	148 bytes	11.8
SIMPLIcity	36 bytes	2.9
Our System	100-bit	1

Table 2: Compactness of region representation: A comparison.

7.3 Compactness of Region Representation

Figure 5 shows the average precision using different sized region bit vectors, as compared to the original 14-D region feature vector. All of them use EMD* match. Compared with the 14-D (448-bit) region feature vector, a factor of more than 4 reduction in region representation size can be achieved without much loss in effectiveness. As shown in Table 2, our 100-bit region representation is 3 - 72 times more compact than the region representation used by other RBIR systems.

7.4 Effectiveness and Efficiency of Filtering

We test our approximate EMD embedding algorithm on various sized region bit vectors, using different M (number

[5]http://robotics.stanford.edu/~rubner/emd/default.htm

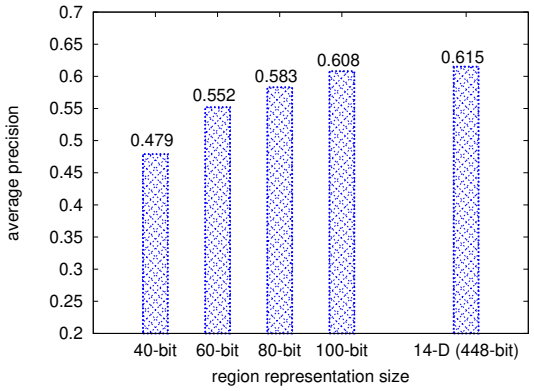

Figure 5: Effectiveness comparison of different region representations, using EMD* match.

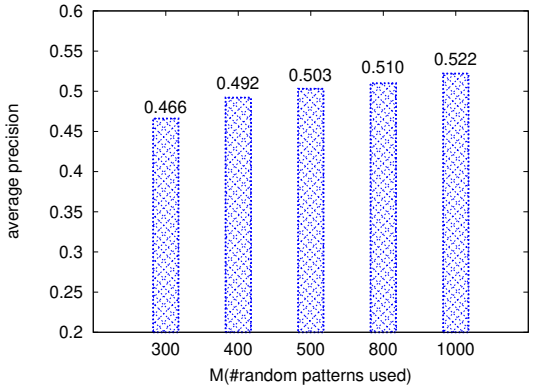

Figure 6: Effectiveness comparison of approximate EMD embedding using different number of random patterns. The region bit vectors used are 100-bit vectors. Each random pattern looks at $H = 3$ positions.

of random patterns) and different H (number of positions in each pattern) values. Figure 6 shows the effectiveness of our embedding algorithm using different number of random patterns (and $H = 3$) to embed images represented by 100-bit region vectors. For each set of embedded image vectors, effectiveness is calculated by ranking images based on the L_1 distance between two images' embedded feature vectors. ¿From the numbers, we can see that although the embedding introduces distortion to the original EMD, it still has fairly good average precision.

We pick the embedded vectors generated using 500 random patterns and convert the 500-dimensional embedded image feature vectors into 500-bit, 800-bit and 1000-bit vectors respectively, to test our filtering algorithm. Figure 7 and Figure 8 show the effectiveness and average query time using different sized bit vectors, different number of filtered images, and with or without exact EMD* calculation and reranking. We can see that filtering plus exact EMD* reranking works very well. Its effectiveness is almost as good as exact EMD* match, while greatly reducing query time.

As shown in Table 3, without embedding and filtering, EMD* match takes 0.698 seconds on average to finish a query; with embedding and filtering, only 0.130 seconds is

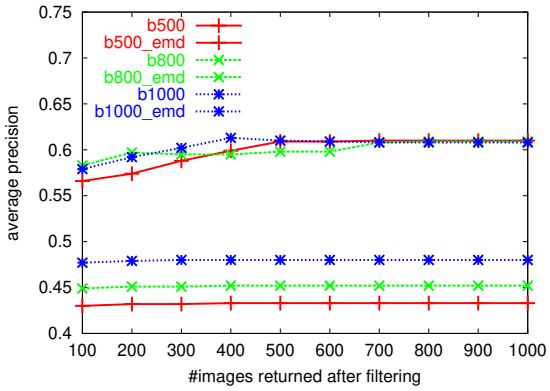

Figure 7: Effectiveness comparison using different image bit vector sizes and different number of images returned after filtering, with and without exact EMD* on the returned images. E.g. "b500_emd" uses 500-bit image vectors and exact EMD* is computed to rerank the filtered images.

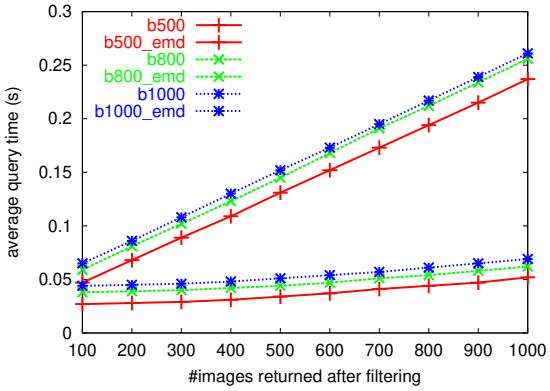

Figure 8: Comparison of average query time using different image bit vector sizes and different number of images returned after filtering, with and without exact EMD* on the returned images.

needed, a factor of more than 5 improvement in query speed. Table 3 also lists the speed of feature vector to bit vector transformation and EMD embeddings. The total amount of time spend on these operations is only 271 μs, which is negligible compared to the overall query time.

8. CONCLUSION

This paper reports our investigation on using compact data structures for content-based image retrieval. We have shown that our proposed method can achieve high-quality similarity searches with highly compact data structures.

We have evaluated the effectiveness of each component of our proposed method with our prototype system on an image database. Our results show that the proposed EMD* match is an effective measure, 17% - 76% more effective than previously proposed region-based image similarity measures.

The experimental results show that our proposed thresholding and transformation algorithm is an effective way to create compact metadata for similarity searches. Even when

query method	speed
SIMPLIcity	0.449 s/query
exact EMD* match	0.698 s/query
EMD* match w/ embedding and filtering	0.130 s/query
operation	speed
region vector transformation 14-D ⇒ 100-bit	16 μs
EMD embedding 100-bit rgn vec ⇒ 500-D img vec	157 μs
image vector transformation 500-D ⇒ 1000-bit	98 μs
subtotal	271 μs

Table 3: Comparison of average query time and operation speed.

such metadata for each region is 3 to 72 times smaller than the representations used by previous systems, the image retrieval system can achieve higher quality similarity searches.

Finally, our results show that approximate EMD embedding is an effective way to build sketches for filtering images for similarity searches. In our experiments, the filtering method can speed up the search time by about a factor of 5 with little loss in similarity search effectiveness.

In the near future, we plan to further evaluate our method with other image databases. SIMPLIcity uses wavelet transformation which does not work well with low resolution images. We plan to compare our method with SIMPLIcity using higher resolution images. Some theoretical analysis would be helpful to better understand the square-root region weighting function and region distance thresholding. We also plan to devise an index data structure for images which would enable fast query answering without examining the sketches of all images in the system. Since we produce sketches for images which are bit vectors, in principle, it should be possible to use known nearest neighbor search data structures for L_1 distance [13, 17, 8] in order to index these sketches.

Acknowledgments

This project is supported in part by NSF grants CCR-0205594 and CNS-0406415. Moses Charikar is supported in part by DOE Early Career Principal Investigator award DE-FG02-02ER25540, NSF CAREER award CCR-0237113 and an Alfred P. Sloan Fellowship.

The authors would like to thank James Z. Wang for providing the SIMPLIcity code and the test image collection. They would also like to thank Yining Deng and B.S. Manjunath for the JSEG tool; Yossi Rubner for the EMD implementation; and Martin Heczko, Alexander Hinneburg, Daniel Keim for the similar image sets.

9. REFERENCES

[1] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

[2] S. Ardizzoni, I. Bartolini, and M. Patella. Windsurf: Region-based image retrieval using wavelets. In *DEXA Workshop*, pages 167–173, 1999.

[3] I. Bartolini, P. Ciaccia, and M. Patella. A sound algorithm for region-based image retrieval using an index. In *DEXA Workshop*, pages 930–934, 2000.

[4] T. Batu *et al.* A sublinear algorithm for weakly approximating edit distance. In *Proc. of STOC'03*, pages 316–324, 2003.

[5] B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of ACM*, 13(7):422–426, 1970.

[6] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. In *Proc. of 6th Intl. World Wide Web Conf.*, pages 391–404, 1997.

[7] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. In *Proc. of 3rd Intl. Conf. on Visual Information and Information Systems*, pages 509–516, 1999.

[8] M. Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. of STOC'02*, pages 380–388, 2002.

[9] Y. Deng and B. S. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(8):800–810, 2001.

[10] J. P. Eakins and M. E. Graham. Content-based image retrieval: A report to the JISC Technology Applications Programme. Technical report, University of Northumbria at Newcastle, Institute for Image Data Research, 1999.

[11] H. Greenspan, G. Dvir, and Y. Rubner. Context-dependent segmentation and matching in image databases. *Computer Vision and Image Understanding*, 93:86–109, 2004.

[12] P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *Proc. of FOCS'00*, pages 189–197, 2000.

[13] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. of STOC'98*, pages 604–613, 1998.

[14] P. Indyk and N. Thaper. Fast image retrieval via embeddings. In *3rd Intl. Workshop on Statistical and Computational Theories of Vision*, 2003.

[15] F. Jing, M. Li, H. Zhang, and B. Zhang. An effective region-based image retrieval framework. In *Proc. of ACM Multimedia'02*, pages 456–465, 2002.

[16] M. H. Kryder. Future magnetic recording technologies. In *FAST'02, invited talk*, 2002.

[17] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000.

[18] W. Ma and B. S. Manjunath. NETRA: A toolbox for navigating large image databases. *Multimedia Systems*, 7(3):184–198, 1999.

[19] W. Ma and H. Zhang. Benchmarking of image features for content-based retrieval. In *Proc. of IEEE 32nd Asilomar Conf. on Signals, Systems, Computers*, volume 1, pages 253–257, 1998.

[20] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using brightness and texture. In *Proc. of NIPS*, pages 1255–1262, 2002.

[21] A. Natsev, R. Rastogi, and K. Shim. WALRUS: A similarity retrieval algorithm for image databases. In *Proc. of ACM SIGMOD'99*, pages 395–406, 1999.

[22] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[23] Y. Rui, T. S. Huang, and S.-F. Chang. Image retrieval: Current techniques, promising directions and open issues. *Journal of Visual Communication and Image Representation*, 10(4):39–62, 1999.

[24] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[25] J. R. Smith and S.-F. Chang. VisualSEEk: A fully automated content-based image query system. In *Proc. of ACM Multimedia'96*, pages 87–98, 1996.

[26] M. Stricker and M. Orengo. Similarity of color images. In *Proc. of SPIE Storage and Retrieval for Image and Video Databases*, volume 2420, pages 381–392, 1995.

[27] R. C. Veltkamp and M. Tanase. Content-based image retrieval systems: A survey. Technical Report UU-CS-2000-34, Utrecht University, Information and Computer Sciences, 2000.

[28] J. Z. Wang, J. Li, and G. Wiederhold. SIMPLIcity: Semantics-sensitive integrated matching for picture libraries. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(9):947–963, 2001.