A Clustering Based Approach to Perceptual Image Hashing

Vishal Monga, *Student Member, IEEE*, Arindam Banerjee, and
Brian L. Evans, *Senior Member, IEEE*

Abstract—
A perceptual image hash function maps an image to a short binary string based on an image's appearance to the human eye. Perceptual image hashing is useful in image databases, watermarking, and authentication. In this paper, we decouple image hashing into feature extraction (intermediate hash) followed by data clustering (final hash). We show that the decision version of our clustering problem is NP complete. Then, for any perceptually significant feature extractor, we propose a polynomial-time heuristic clustering algorithm that automatically determines the final hash length needed to satisfy a specified distortion. Based on the proposed algorithm, we develop two variations to facilitate perceptual robustness vs. fragility trade-offs. We validate the perceptual significance of our hash by testing under Stirmark attacks. Finally, we develop randomized clustering algorithms for the purposes of secure image hashing.

Index terms— hashing, data clustering, image indexing, image authentication

Contact— Dr. Vishal Monga, 1 University Station C0803, The University of Texas, Austin, TX 78712 USA. Voice: +1-512-425-1306, Fax: +1-512-471-5907, vishal@ece.utexas.edu

V. Monga and B. L. Evans are with the Embedded Signal Processing Laboratory, and A. Banerjee is with the Laboratory for Artificial Neural Systems, in the Dept. of Electrical and Computer Engineering at The University of Texas at Austin, Austin, TX 78712 USA. {vishal,abanerje,bevans}@ece.utexas.edu. V. Monga and B. L. Evans were supported by a gift from the Xerox Foundation and A. Banerjee was supported by an IBM PhD Fellowship.

I. Introduction

An image hash function maps an image to a short binary string based on the image's appearance to the human eye. In particular, a perceptual image hash function should have the property that two images that look the same to the human eye map to the same hash value, even if the images have different digital representations; e.g., being separated by a large distance in mean squared error. This differentiates a perceptual hash from traditional cryptographic hashes, such as SHA-1 and MD-5 [1]. SHA-1 and MD-5 hashes are extremely sensitive to the input data, i.e., a one bit change in the input changes the output dramatically.

A perceptual image hash function would facilitate comparisons and searches of images in large databases in which several "perceptually identical" versions of an image may exist. Further need for such image descriptors arises for the purposes of *integrity verification*. Because of the easy-to-copy nature of digital media, digital data can be tampered with and hence, there exists a need to be able to verify the content of the media to ensure its authenticity. In the literature, the methods used for media verification can be classified into two categories: digital signature-based [2], [3], [4], [5], [6], [7] and watermark-based [8], [9], [10], [11], [12], [13]. A digital signature is a set of features extracted from the media that sufficiently represents the content of the original media. Watermarking, on the other hand, is a media authentication/protection technique that embeds invisible (or inaudible) information into the media. For content authentication, the embedded watermark can be extracted and used for verification purposes. The major difference between a watermark and a digital signature is that the embedding process of the former requires the content of the media to change. However, for content authentication, both the watermark-based approach and the digital signature-based approach are expected to be sensitive to any malicious modification of the media while being able to tolerate incidental modifications such as JPEG compression (with compression ratios that do not result in significant loss of perceptual quality) or image enhancement. In other words, the digital signature or hash is required to be robust to perceptually insignificant changes to the image. Another application of perceptual image hashing is content dependent key generation for video watermarking [14], [15].

Let $\mathcal{I}$ denote a set of images (e.g., all natural images of a particular size) with finite cardinality. Also, let $\mathcal{K}$ denote the space of *secret* keys.[1] Our hash function then

[1]The key space in general can be constructed in several ways. A necessary but not sufficient condition for secure hashing is that the

takes two inputs, an image $I \in \mathcal{I}$ and a *secret* key $K \in \mathcal{K}$, to produce a q-bit binary hash value $h = H(I, K)$. Let $I_{ident} \in \mathcal{I}$ denote an image such that I_{ident} looks the same as I. Likewise, an image in $\mathcal{I}$ that is perceptually distinct from I will be denoted by I_{diff}. Let $\mathcal{Q} = \{H(I, K) \mid I \in \mathcal{I},\ K \in \mathcal{K}\}$, i.e., the set of all possible realizations of the hash algorithm on the product space $\mathcal{I} \times \mathcal{K}$. Also, for a fixed $I_0 \in \mathcal{I}$ define $\mathcal{O}(I_0) = \{H(I_0, K) \mid K \in \mathcal{K}\}$. That is, for a fixed image I_0, $\mathcal{O}(I_0)$ is the set of all possible realizations of the hash algorithm over the key space $\mathcal{K}$. Then, for some θ_1, θ_2 satisfying $0 < \theta_1, \theta_2 < 1$, three desirable properties of a *perceptual* hash are identified as follows:

1. **Perceptual robustness**:
$Probability(H(I, K) = H(I_{ident}, K)) \geq 1 - \theta_1$, for a given θ_1
2. **Fragility to visually distinct images**:
$Probability(H(I, K) \neq H(I_{diff}, K)) \geq 1 - \theta_2$, for a given θ_2
3. **Unpredictability of the hash**:
$Probability(H(I, K) = v) \approx \frac{1}{2^q}, \quad \forall\ v \in \{0, 1\}^q$

Note that the probability measure in the first two properties is defined over the set $\mathcal{Q}$. For example, property 1 requires that for any pair of "perceptually identical" images in $\mathcal{I}$ and any $K \in \mathcal{K}$, the hash values must be identical with high probability. The probability measure in the third property, however, is defined on $\mathcal{O}(I_0)$. That is, the third property requires that as the secret key is varied over $\mathcal{K}$ for a fixed input image, the output hash value must be approximately uniformly distributed among all possible q-bit outputs.

Remark: The three desired properties as laid out above are those of an "ideal" hash algorithm. Whether or not such hash algorithms can even be constructed (especially in a computationally feasible time) remains an outstanding open problem in media hashing. We therefore do not claim to achieve these properties for arbitrary values of θ_1, θ_2 and q, but instead provide heuristic solutions that achieve these goals with an acceptably high probability.

Further, the three desirable hash properties conflict with one another. The first property amounts to robustness under small perturbations whereas the second one requires minimization of collision probabilities for perceptually distinct inputs. There is clearly a trade-off here, e.g., if very crude features were used, then they would be hard to change (i.e., robust), but it is likely that one is going to encounter collision of perceptually different images. Likewise for perfect randomization, a uniform distribution on the output hash values (over the key space) would be needed which in general, would deter achieving the first property. From a security viewpoint though, the second and third properties are very important; i.e., key based randomization is introduced to prevent against intentional attacks of guessing, and forgery. It is desirable for the hash algorithm to achieve these (conflicting) goals to some extent and/or facilitate trade-offs.

key space should be large enough to preclude exhaustive search. For this paper, unless specified otherwise we will assume the key space to be the Hamming space of 32-bit binary strings. Also, we choose 32-bits to facilitate simulations presented later in Section VI-C. In practice, larger key sizes, e.g. 512 bits or higher, may be used.

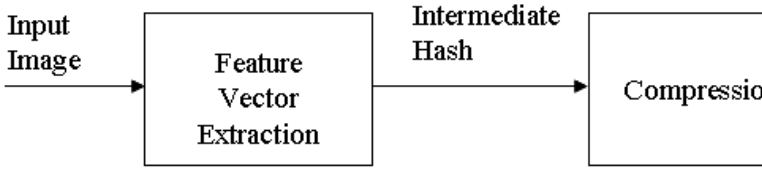

Fig. 1. Block diagram of the hash function.

We partition the problem of deriving an image hash into two steps, as illustrated in Fig. 1. The first step extracts a feature vector from the image, whereas the second stage compresses[2] this feature vector to a final hash value. In the feature extraction step, the two-dimensional image is mapped to a one-dimensional feature vector. This feature vector must capture the perceptual qualities of the image. That is, two images that appear identical to the human visual system should have feature vectors that are close in some distance metric. Likewise, two images that are clearly distinct in appearance must have feature vectors that differ by a large distance. For such feature vector extraction, many algorithms could be used, e.g. [16], [17], [18], [19], [20]. For the rest of the paper, we will refer to this visually robust feature vector as the "intermediate hash".

The second step then compresses this intermediate hash vector to a final hash value. In this paper, we assume the availability of such an intermediate hash vector that has been extracted from the image and a model on its distribution (detailed later in Section IV). The methods previously proposed for this step include using error correction decoding for compression of binary intermediate hash vectors [16] and dither-based compression via distributed source coding [21]. While compression is their primary goal [16], [21], no *explicit* attempt was made to ensure that perceptually identical images are compressed to the same hash value.

The second step will involve clustering of the intermediate hash vector of an input source (image) and the intermediate hash vectors of its perceptually identical versions. In this paper, we present a solution to the second step by developing such a clustering algorithm based on the distribution of intermediate hash vectors [22]. Another important issue is the length (or granularity) of the final hash required to cluster images within a specified distance. Underestimating this length can adversely affect the perceptual qualities of the hash. A significant contribution of our work is that this length is determined as a natural outcome of our proposed clustering algorithm.

Section II formally defines the problem for the feature vector compression step of the two-step hash function. For the second step, Section III brings out the limitations of

[2]In this paper, the term compression is used purely to imply a significant reduction in the dimensionality of the input feature space. Traditional compression involves the construction of both encoding and decoding modules. However, in hashing, only the first module is designed as irreversibility is desirable. This is also emphasized later in Section V-A.

traditional vector quantization (VQ) based compression approaches. Section IV formulates a novel cost function for feature vector (or intermediate hash) compression for the perceptual hashing application. Section V presents heuristic clustering algorithms for minimizing the cost function defined in Section IV. We first present a deterministic algorithm in Section V-A that attempts to retain the perceptual significance of the hash as best as possible. Next, a randomized clustering is proposed (based on a secret key) in Section V-B for the purposes of secure hashing. Randomization in the algorithm is of great importance in an adversarial scenario in which a malicious attacker may try to generate inputs that defeat the hash algorithm. Experimental results are presented in Sections VI-A through VI-C. In Section VI-A, we compare with traditional VQ as well as error correction decoding approaches [16] and test under Stirmark [23] attacks to show the efficacy of our clustering algorithm(s) for perceptual hash compression. Section VI-B presents a statistical analysis of our algorithm using precision-recall (or receiver operating characteristic (ROC)) curves. Section VI-C then presents results that demonstrate the security properties of our randomized clustering algorithm. Section VII concludes the paper with suggestions for future work.

II. Problem Statement

We first establish notation that will be used throughout the paper. Let V denote the metric space of intermediate hash vectors extracted at step 1 of the hash algorithm in Fig. 1. Let $\mathcal{L} \subseteq V$ represent a finite set of vectors $\{l_i\}_{i=1}^{n}$ on which the clustering/compression algorithm is applied. Let $D : V \times V \rightarrow \mathcal{R}_+$ be the distance *metric* defined on the product space. Finally, let $C : \mathcal{L} \rightarrow \{1, 2, ..., k\}$ denote the clustering map. Note, in a typical application $k << n$, re-emphasizing the fact that the clustering as well as the overall hash is a many-to-one mapping.

Our goal is to have all images that are visually indistinguishable map to the same hash value with high probability. In that sense an image hash function is similar to a *vector quantization* (VQ) or *clustering* scheme. We are attempting to cluster images whose intermediate hash vectors are close in a metric into the same cell. In particular, it is desired that with high probability

$$if\ D(l_i, l_j) < \epsilon \text{ then } C(l_i) = C(l_j) \tag{1}$$

$$if\ D(l_i, l_j) > \delta \text{ then } C(l_i) \neq C(l_j) \tag{2}$$

where $0 < \epsilon < \delta$, and l_i, l_j denote random vectors in $\mathcal{L}$ (following the distribution of the intermediate hash) and let $C(l_i)$, $C(l_j)$ represent the clusters to which these vectors map after applying the clustering algorithm.

III. Conventional VQ based Compression Approaches

The goal of the compression step as discussed above is to achieve a clustering of the intermediate hash vector of an image I and the intermediate hash vectors of images that are identical in appearance to I with high probability. In that respect, it is useful to think of perceptually insignificant modifications or attacks on an image as "distortions" to the image. We may then look to compress the intermediate hash vectors while tolerating a specified distortion. The design problem for a vector quantization or compression scheme that minimizes an average distortion is to obtain a K partioning of the space V by designing codevectors $\{l^k\}_{k=0}^{K-1}$ in V such that

$$\sum_{k=0}^{K-1} \sum_{l \in S_k} P(l) D(l, l^k) < \epsilon \tag{3}$$

Here, $P(l)$ denotes the probability of occurrence of vector l and S_k denotes the k^{th} cluster. Average distance minimization is a well known problem in the VQ literature and a large number of algorithms [24], [25], [26] have been proposed to solve it.

However, an average distance type cost function as in (3) is inherently not well suited for the hashing application. First, while the design of the codebook in (3) ensures that the average distortion is less than ϵ, there is no guarantee that perceptually distinct vectors, i.e., intermediate hash vectors that are separated by more than δ, indeed map to different clusters. In some applications, such as image authentication where the goal is to detect content changes, such guarantees may indeed be required because mapping perceptually distinct vectors to the same final hash value would be extremely undesirable. More generally, the nature of the cost function in (3) does not allow trade-offs between desired properties (1) and (2) of the hash algorithm.

Secondly, the cost in (3) increases linearly as a function of the distance between the intermediate hash vector(s) and the codebook vector(s). Intuitively though, it is desirable to penalize some errors much more strongly than others, e.g., if vectors really close are not clustered together, or vectors very far apart are compressed to the same final hash value. A linear cost function does not reflect this behavior.

Based on these observations, we propose a new cost function for the perceptual hashing application that does not suffer from the limitations of average distance measures.

IV. Formulation of the Cost Function

In this section, we formulate the cost function to be minimized by the proposed clustering algorithm. First, we analyze several fundamental properties of our requirements of (1), (2), and the intermediate hash.

We say that an error is encountered when either (1) and/or (2) is not satisfied for any pair of vectors (l_i, l_j). Intuitively then, we must ensure that errors occur for vectors that are less likely or that the clustering must necessarily be dictated by the probability mass function of the vectors in $\mathcal{L}$.

We now describe the construction of our clustering cost function. Let $\mathbf{P} : \mathcal{L} \times \mathcal{L} \rightarrow [0, 1]$ be the joint distribution

matrix of intermediate hash pairs

$$\mathbf{P} = \begin{bmatrix} p(1,1) & p(1,2) & \cdots & p(1,n) \\ p(2,1) & p(2,2) & \cdots & p(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ p(n,1) & p(n,2) & \cdots & p(n,n) \end{bmatrix} \quad (4)$$

where $\mathbf{P}_{ij} = p(i,j) = p(i)p(j)$, i.e. pairwise independence of (l_i, l_j) is assumed. $p(i)$, $p(j)$ respectively denote the probability of occurrence of vectors l_i, l_j and n is the total number of vectors to be clustered.

To estimate the probability measure introduced above, we employ a statistical model on the intermediate hash/image feature vectors. The fundamental underlying principle is to define rectangular blocks (or sub-images) in an image as a real two dimensional homogenous Markov random field (RF) $X(m_1, m_2)$ on a finite lattice $(m_1, m_2) \in L \subset Z^2$. The basis for connecting such a statistical definition to perception is the hypothesis first stated by Julesz [27] and reformulated by several other authors; e.g., [28], [29] : there exists a set of functions $\phi_k(X), k = 1, 2, \ldots N$ such that samples drawn from any two RFs that are equal in expectation over this set are visually indistinguishable.

In particular, we employ a universal parametric statistical model for natural images developed by Portilla and Simoncelli [30] that works with a complex overcomplete wavelet representation of the image. The image features that we extract are also based on such a representation and are described in [20]. Our Markov statistical descriptors, i.e. ϕ_ks, are then based on pairs of wavelet coefficients at adjacent spatial locations, orientation and scales. In particular, we measure the expected product of the raw coefficient pairs (i.e., correlation), and the expected product of their magnitudes. For pairs of coefficients at adjacent scales, we also include the expected product of the fine scaled coefficient with the phase-doubled coarse coefficient. Finally, we include a small number of marginal statistics of the image pixels and lowpass wavelet coefficients at different scales.

There is no inherent structure to the probability mass functions associated with these random fields (except the Markovian property due to spatial correlation in images). A mathematically attractive choice is a maximum entropy density [31] of the form

$$\mathcal{P}(\vec{x}) \propto \prod_k e^{-\lambda_k \phi_k(\vec{x})} \quad (5)$$

where $\vec{x} \in \mathcal{R}^{|L|}$ corresponds to a vectorized sub-image, and λ_ks are the Lagrange multipliers. The maximum entropy density is optimal in the sense that it does not introduce any new constraints on the RF beyond those of perceptual equivalence under expected values of ϕ_ks. The density in (5) is defined on RFs that are portions of natural images. Since features are functions of RFs a probability density is in turn induced on the feature vectors.

Our choice of a statistical model vs. using an empirical distribution on image features is based on the robustness of model parameters as more samples (images) are added. By the weak law of large numbers, it can be argued that the model parameters become nearly invariant once a sufficiently large sample set is considered. In our experiments, we worked with a set of roughly 2500 natural images. More details on the model parameters and the typical distributions on image feature vectors may be found in [30].

Next, we define $\mathbf{C_1}$ as the joint cost matrix for violating (1); i.e., the cost paid if $D(l_i, l_j) < \epsilon$, yet $C(l_i) \neq C(l_j)$. In particular, $\forall\ i, j = 1, 2, ..., n$

$$c_1(i,j) = \begin{cases} \Gamma_1^{-\alpha_1 D(l_i,l_j)} & \text{if } D(l_i,l_j) < \epsilon,\ C(l_i) \neq C(l_j) \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where $\alpha_1 > 0$ and $\Gamma_1 > 1$ are algorithm parameters. This construction follows intuitively because the cost for violating (1) must be greater for smaller distances, i.e. if the vectors are really close and not clustered together.

Similarly, $\mathbf{C_2}$ is defined as the joint cost matrix for violating (2)

$$c_2(i,j) = \begin{cases} \Gamma_2^{\alpha_2 D(l_i,l_j)} & \text{if } D(l_i,l_j) > \delta,\ C(l_i) = C(l_j) \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

As before, $\alpha_2 > 0$ and $\Gamma_2 > 1$. In this case however, the cost is an increasing function of the distance between $(l_i,\ l_j)$. This is also natural as we would like to penalize more if vectors far apart (and hence perceptually distinct) are clustered together. An exponential cost as opposed to linear in an average distance VQ, ensures that errors associated with large distances are penalized severely.
Further, let matrices $\mathbf{S_1}$ and $\mathbf{S_2}$ be defined as

$$s_1(i,j) = \begin{cases} \Gamma_1^{-\alpha_1 D(l_i,l_j)} & \text{if } D(l_i,l_j) < \epsilon \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

$$s_2(i,j) = \begin{cases} \Gamma_2^{\alpha_2 D(l_i,l_j)} & \text{if } D(l_i,l_j) > \delta \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

Note, that $\mathbf{S_1}$ is different from $\mathbf{C_1}$ in the sense that the entries of $\mathbf{S_1}$ include the cost for all possible errors that can be committed due to (1), while $\mathbf{C_1}$ is the cost matrix due to (1) for the errors actually made by the clustering algorithm. The same holds for $\mathbf{S_2}$ and $\mathbf{C_2}$. We normalize the entries in $\mathbf{C_1}$ and $\mathbf{C_2}$ to define normalized cost matrices $\tilde{\mathbf{C}}_\mathbf{1}$ and $\tilde{\mathbf{C}}_\mathbf{2}$ such that

$$\tilde{c}_1(i,j) = \frac{c_1(i,j)}{\sum_i \sum_j s_1(i,j)} \quad (10)$$

$$\tilde{c}_2(i,j) = \frac{c_2(i,j)}{\sum_i \sum_j s_2(i,j)} \quad (11)$$

This normalization ensures that $\tilde{c}_1(i,j), \tilde{c}_2(i,j) \in [0,1]$.

Finally, we define the total cost function given by

$$P_{err} = E[\tilde{\mathbf{C}}_\mathbf{1} + \tilde{\mathbf{C}}_\mathbf{2}] \quad (12)$$

The expectation is taken over the joint distribution of (l_i, l_j). Assuming pairwise independence of (l_i, l_j), (12) may be rewritten as

$$P_{err} = \sum_i \sum_j p(i)p(j)\left(\tilde{c}_1(i,j) + \tilde{c}_2(i,j)\right) \quad (13)$$

At this point it is worth re-emphasizing that the distance function $D(l_i, l_j)$ can be any function of l_i and l_j that satisfies metric properties, i.e., non-negativity, symmetry and triangle inequality. In particular, we are not restricting ourselves to any class of functions other than requiring $D(\cdot,\cdot)$ to be a metric. In practice, the choice of $D(\cdot,\cdot)$ is motivated by the nature of features extracted in Stage 1 of the hash algorithm.

The two additive terms in (12), $E[\tilde{\mathbf{C_1}}]$ and $E[\tilde{\mathbf{C_2}}]$ quantify the errors resulting from violating (1) and (2) respectively. In particular, $E[\tilde{\mathbf{C_1}}]$ can be interpreted as the expected cost of violating (1). Similarly, $E[\tilde{\mathbf{C_2}}]$ signifies the expected cost incurred by violating (2). It is this structure of the cost function in (12) that our proposed clustering algorithm exploits to facilitate trade-offs between goals (1) and (2) of the hash algorithm. Note in the special case that $\alpha_1, \alpha_2 = 0$, $E[\mathbf{C_1}]$ and $E[\mathbf{C_2}]$ represent the total probability of violating (1) and (2) respectively.

Indyk *et al.* [32], [33] have addressed a problem similar to the one we present in Section II. They introduce the notion of *locally sensitive hashing* (LSH) [32] and use it to develop sublinear time algorithms for the *approximate nearest neighbor search* (NNS) problem [34] in high dimensional spaces. The key idea in their work is to use hash functions [35], [36] such that the probability of collision is much higher for vectors that are close to each other than for those that are far apart. However, while they prove the existence of certain parametrized LSH families [32] they do not concern themselves with the problem of codebook design for specific cost functions. Instead, their work focuses on developing fast algorithms for the NNS problem based on the availability of such hash codebooks. Our objective here is to develop a clustering algorithm or equivalently design a codebook to minimize the cost function in (12) that is well suited for the perceptual image (or media) hashing application.

V. Proposed Clustering Algorithms

Finding the optimum clustering that would achieve a global minimum for the cost function in (12) is a hard problem. The decision version of the problem: "for a fixed number of clusters k, is there a clustering with a cost less than a constant?" is NP-complete. We sketch a proof of NP completeness in the appendix. Hardness results for the search version, that actually finds the minimum cost solution, can be similarly shown. In this paper, we present a polynomial-time *greedy heuristic* for solving the problem.

A. Deterministic Clustering

For the following discussion, vectors in $\mathcal{L}$ will be referred to as "data points". Fig. 2 describes the basic clustering algorithm. A visualization of the same is shown in Fig. 5. The data points in the input space are covered to a large extent by hyperspheres (clusters) of radius $\frac{\epsilon}{2}$. For each pair of points $(l_i, l_j) \in S_k$ and cluster center l^k, we have

$$D(l_i, l_j) < D(l_i, l^k) + D(l^k, l_j) \tag{14}$$

This is true because $D(\cdot,\cdot)$ defines a metric. By virtue of Steps 3 and 5 of the basic clustering algorithm, $D(l_i, l^k) < \frac{\epsilon}{2}$, $D(l^k, l_j) < \frac{\epsilon}{2}$ and hence $D(l_i, l_j) < \epsilon$. The algorithm therefore attempts to cluster data points within ϵ of each other and in addition the cluster centers are chosen based on the strength of their probability mass function. This ensures that "perceptually close" data points are clustered together with high probability.

We make the following observations about the basic clustering algorithm:

- The minimum distance between any two members of two different clusters has a lower bound of ϵ and hence there are no errors from violating (1), which is guaranteed by Step 4 of the algorithm.
- Within each cluster the maximum distance between any two points is at most ϵ, and because $0 < \epsilon < \delta$, there are no violations of (2).
- The data points that are left unclustered are less than $\frac{3}{2}\epsilon$ from some member of at least one cluster.

For perceptual robustness, i.e. achieving (1), we would like to minimize $E[\tilde{\mathbf{C_1}}]$. Likewise, in order to maintain fragility to visually distinct inputs, we would like $E[\tilde{\mathbf{C_2}}]$ to be as small as possible (ideally zero). Exclusive minimization of one would compromise the other. Next, we present two different approaches to handle the unclustered data points so that trade-offs may be facilitated between achieving properties (1) and (2).

A.1 *Approach 1*

Fig. 3 describes Approach 1 for handling the unclustered data points. Step 2 of the algorithm in Fig. 3 looks for the set of clusters $\mathcal{S}_\delta$, such that every point in each of the clusters is less than δ away from the unclustered data point l^* under consideration. Step 3 then computes the minimum cost cluster to which to assign l^*. In essence, this approach tries to minimize the cost in (12) conditioned on the fact that there are no errors from violating (2). This could be useful in authentication applications in which mapping

1: Obtain user defined parameters ϵ and δ. Set the number of clusters $k = 1$.
2: Select the data point associated with the highest probability mass, and label it l^1
3: Make the first cluster by including all data points l_j such that $D(l^1, l_j) < \frac{\epsilon}{2}$
4: $k = k + 1$. Select the highest probability data point l^k among the unclustered points such that $\min_{S\in\mathcal{C}} D(l^k, S) \geq \frac{3}{2}\epsilon$ where S is any cluster and $\mathcal{C}$ denotes the set of clusters formed till this step of the algorithm. $D(l^k, S)$ is calculated using the notion of distance from a set given by: $D(x, S) = \min_{y\in S} D(x, y)$
5: Form the k^{th} cluster S_k by including all unclustered data points l_j such that $D(l^k, l_j) < \frac{\epsilon}{2}$
6: Repeat steps 4–5 until no more clusters can be formed.

Fig. 2. Basic clustering algorithm.

perceptually distinct inputs to the same hash may be extremely undesirable.

A.2 *Approach 2*

Approach 1 clusters the remaining data points to ensure that $E[\tilde{\mathbf{C_2}}] = 0$. The goal in Approach 2 is to effectively trade-off the minimization of $E[\tilde{\mathbf{C_1}}]$ at the expense of increasing $E[\tilde{\mathbf{C_2}}]$ via a tuning parameter[3] β (see Fig. 4). This can be readily observed by considering extreme values of β. For $\beta = \frac{1}{2}$ a joint minimization is performed. The other extreme $\beta = 1$ corresponds to the case when the unclustered data points are assigned, so as to exclusively minimize $E[\tilde{\mathbf{C_1}}]$. For $\delta \geq \frac{5}{2}\epsilon$, Approaches 1 and 2 coincide because all of the unclustered points are then necessarily within δ of the existing clusters. Finally, note that a meaningful dual of Approach 1 does not exist. This is because requiring $E[\tilde{\mathbf{C_1}}] = 0$ leads to the trivial solution that all data points are collected in one big cluster.

In traditional VQ based compression approaches, the number of codebook vectors or the *rate* of the vector quantizer [24] is decided in advance and an optimization is carried out to select the best codebook vectors. In our algorithm, the length of the hash (given by $\lceil \log_2(k) \rceil$ bits) is determined adaptively for a given ϵ, δ and source distribution. Note however, that we do not claim for this to be the minimum possible number of clusters that achieves a particular value of the cost function in (12). Nevertheless, the length of the hash in bits (or alternatively the number of clusters) as determined by our proposed clustering is enough so that the perceptual significance of the hash is not compromised.

Remark: Note that another difference from compression applications is the fact that compression entails the design of reconstruction values as well (in addition to quantization bins/clusters or Vornoi regions). In the hashing application, however, these may be chosen for convenience (e.g., a straightforward enumeration using $\lceil \log_2(k) \rceil$ bits for k clusters) as long as the notion of closeness is preserved.

[3] $\beta \in [\frac{1}{2}, 1]$ as opposed to $[0, 1]$. This is because values of $\beta \in [0, \frac{1}{2})$ do not lead to meaningful clusterings. For example, $\beta = 0$ ignores the minimization of $E[\tilde{\mathbf{C_1}}]$ which is the primary objective of the algorithm.

1: Given the k clusters formed by running the basic clustering algorithm, select the data point l^* among the unclustered points that has the highest probability mass
2: For each existing cluster S_i, $i = 1, 2...k$ compute $d_i = \max_{x \in S_i} D(l^*, x)$
Let $\mathcal{S}_\delta = \{S_i \text{ such that } d_i \leq \delta\}$
3: IF $\mathcal{S}_\delta = \phi$ THEN $k = k + 1$ and $S_k = l^*$ is a cluster of its own
ELSE for each $S_i \in \mathcal{S}_\delta$ define $F(S_i) = \sum_{l \in \bar{S}_i} p(l)p(l^*)c_1(l, l^*)$
where $\bar{S}_i$ denotes the complement of S_i; i.e., all clusters in $\mathcal{S}_\delta$ except S_i. Then, l^* is assigned to the cluster $S^* = \arg\min_{S_i} F(S_i)$
4: Repeat steps 1–3.

Fig. 3. Approach 1 locally minimizes $E[\tilde{\mathbf{C_1}}]$ conditioned on $E[\tilde{\mathbf{C_2}}] = 0$ where $\tilde{\mathbf{C_2}}$ is defined by (11).

1: Given the k clusters formed by running the basic clustering algorithm, select the data point l^* among the unclustered points that has the highest probability mass
2: For each existing cluster S_i, $i = 1, 2, ...k$ define $F(S_i) = \beta \sum_{l \in \bar{S}_i} p(l)p(l^*)c_1(l, l^*) + (1 - \beta) \sum_{l \in S_i} p(l)p(l^*)c_2(l, l^*)$
where $\beta \in [\frac{1}{2}, 1]$, and $\bar{S}_i$ denotes the complement of S_i. Then, l^* is assigned to the cluster $S^* = \arg\min_{S_i} F(S_i)$. Analogous to Approach 1, this includes the case that l^* is a cluster by itself; in that case, we increment k.
3: Repeat steps 1–2.

Fig. 4. Approach 2 enables trade-offs between goals (1) and (2) by varying the real-valued parameter β.

B. Randomized Clustering

The clustering algorithm as presented in the previous subsection is a perfectly deterministic map; i.e., a particular input intermediate hash vector always maps to the same output hash value. We now present a randomization scheme to enhance the security properties of our hash algorithm and minimize its vulnerability to malicious inputs generated by an adversary.

Recall that the heuristic employed in the deterministic algorithm (both for Approaches 1 and 2) was to select the vector or data point with the highest probability mass among the candidate unclustered data points as the cluster center. In other words, we select the data point that has the highest probability mass as the cluster center with probability one. The randomization rule that we propose modifies this heuristic to select cluster centers in a probabilistic manner. That is, there is a non-zero probability of selecting each candidate unclustered data point as the next cluster center. This probability in turn is determined as a function of the original probability mass associated with the data points.

Consider the clustering algorithm with $m \geq 0$ clusters already formed and $i < n$ points clustered. Let $\mathcal{X} \subset \mathcal{L}$ denote the set of unclustered data points that can be chosen as cluster centers. Note that $|\mathcal{X}|$ is not necessarily $n - i$. As described in the basic clustering algorithm (Fig. 2) the set $\mathcal{X}$ consists of all data points $l \in \mathcal{L}$ such that $\min_{S \in \mathcal{C}} D(l, S) \geq \frac{3}{2}\epsilon$ where S is any cluster and $\mathcal{C}$ denotes the set of clusters formed prior to this step of the algorithm. When no more cluster centers can be identified in this manner, the set $\mathcal{X}$ indeed consists of all unclustered data points.

Then, we define a probability measure on the elements

of $\mathcal{X}$ as

$$\pi_i^{(s)} = \frac{(p_i)^s}{\sum_{j\in\mathcal{X}}(p_j)^s} \tag{15}$$

where $s \in \mathcal{R}^+$ is an algorithm parameter and p_i denotes the probability mass associated with data point $l_i \in \mathcal{X}$. The data point $l_i \in \mathcal{X}$ is then chosen as a cluster center with a probability equal to $\pi_i^{(s)}$ [37].

Example: A hypothetical example is presented in Fig. 6. In the example, the set $\mathcal{X}$ consists of four data points $\{l_1, l_2, l_3, l_4\}$ with probability mass values of 0.4, 0.2, 0.1 and 0.1, respectively. The normalized probabilities $\{\pi_i^{(s)}\}_{i=1}^4$ using $s = 1$ are given by $\pi_1^{(1)} = 0.5$, $\pi_2^{(1)} = 0.25$, $\pi_3^{(1)} = 0.125$, and $\pi_4^{(1)} = 0.125$. A secret key K_1 is used to serve as a seed to a pseudorandom number generator that generates a uniformly distributed number a in [0,1] which in turn is used to select one of the data points as the cluster center. Note that the probability that $a \in [0, 0.5]$ is 0.5 and hence the data point l_1 is selected with a probability of 0.5. In general, any data point l_i is selected with probability $\pi_i^{(s)}$. This is indeed the classical approach of sampling from a distribution.

The randomization scheme can be summarized by considering extreme values of s. Note

$$\lim_{s\to\infty} \pi_i^{(s)} = \begin{cases} 1 & \text{for the highest probability data point} \\ 0 & \text{for all other } l_i \in \mathcal{X} \end{cases}$$

In other words, $s \to \infty$ corresponds to the deterministic clustering algorithm. Similarly, the other extreme, i.e. $s = 0$, implies that $\pi_i^{(0)}$ is a uniform distribution or that any data point in $\mathcal{X}$ is selected as a cluster center with the same probability equal to $\frac{1}{|\mathcal{X}|}$. To enhance security, the parameter s may also be generated in random fashion using a second secret key K_2.

In general, in the absence of the secret keys K_1 and K_2, it is not possible to determine the mapping achieved by the randomized clustering algorithm. We demonstrate the hardness of generating malicious inputs by means of experimental results in Section VI-C.

VI. Experimental Results

An intermediate hash (or feature) vector extracted from an image I will be referred to as $\mathbf{fv}(I)$. In the presented experiments, the intermediate hash was generated using the method by Monga *et al.* in [20]. They obtain a *binary* intermediate hash vector from a set of visually robust image feature points. Normalized Hamming distance was used as the distance metric. In particular, they determine empirically

$$D(\mathbf{fv}(I), \mathbf{fv}(I_{ident})) < 0.2. \tag{16}$$

$$D(\mathbf{fv}(I), \mathbf{fv}(I_{diff})) > 0.3. \tag{17}$$

In our clustering framework, the two equations above yield $\epsilon = 0.2$ and $\delta = 0.3$.

A. Deterministic Clustering Results

A.1 Comparison with Error Correction Decoding and Conventional VQ

In our experiments, we extract a binary intermediate hash vector of length $L = 240$ bits from the image. V is, therefore, the Hamming space of dimension L. Further, we set $\mathcal{L} = V$ and hence the total number of vectors to be clustered i.e., $n = 2^{240}$. Because of space and complexity constraints it is clearly impractical to apply the clustering algorithm to that large of a data set. Hence, we take the approach commonly employed in space constrained VQ problems [24]; i.e., we divide the intermediate hash vector into segments of length $M = \frac{L}{m}$ (where m is an integer) and apply the clustering on each segment separately. The resulting binary strings are concatenated to form the final hash. A similar approach for an irreversible compression of binary hash values was used by Venkatesan *et al.* in [16]. They employ error control decoding using Reed-Muller codes [38]. In particular, they break the hash vector to be compressed into segments of length as close as possible to the length of codevectors in a Reed-Muller error correcting code. Decoding is then performed by mapping the segments of the hash vector to the nearest codeword using the exponential pseudo norm [16].

Tables I, II and III respectively, show values of the cost function in (12) by compressing the intermediate hash vector using 1.) our proposed clustering scheme, 2.) the error control decoding scheme as described in [16] and 3.) an average distance VQ approach [24]. We generated the results in Table I by using Approach 2 with $\beta = \frac{1}{2}$. For the error control decoding scheme, we use (8,4), (16,5) and (16,11) Reed-Muller codes. Our proposed clustering algorithm as well as the average distance VQ compression were also employed on segments of the same length to yield a meaningful comparison[4]. Note that VQ compression [24] based on descent methods that gradually improve the codebook by iteratively computing centroids cannot be applied here since the vectors to be compressed are themselves binary (i.e., codebook vector components cannot assume intermediate values between 0 and 1). For the results in Table III, we used the binary VQ compression based on "soft-centroids" proposed by Franti *et al.* [26].

The results in Tables I, II and III clearly reveal that the values for the expected cost of violating (1) and (2) i.e. $E[\tilde{\mathbf{C}}_1]$ and $E[\tilde{\mathbf{C}}_2]$, are orders of magnitude lower when using our clustering algorithm (even as we achieve better compression than error correction decoding approaches). Hence, we show that the codebook as obtained from using error correcting codes and/or conventional VQ based compression approaches does not fare well for perceptual hash compression.

Remark: The proposed clustering algorithm can be used

[4]For an average distance VQ the rate or the number of codebook vectors is to be decided in advance. We decided on this number by determining first the number of clusters (or equivalently the hash length in bits) that result from the application of our proposed clustering and then using a rate slightly higher than that for the average distance VQ. This ensures a fair comparison across the two methods.

to compress feature vectors as long as the distance measure defined on the product space $V \times V$ satisfies metric properties. For example, if the features were to be real valued, the number of data points n or equivalently the set $\mathcal{L}$ should be chosen large enough to sufficiently represent source (feature vector) statistics. A codebook can then be derived from the set $\mathcal{L}$ using the proposed clustering and feature vectors can be mapped to the nearest vector in the codebook based on a minimum cost decoding rule [24].

A.2 *Perceptual Robustness Vs. Fragility Trade-offs*

Table IV compares the value of the cost function in (12) for the two different clustering approaches. For Approach 2 (rows 2 and 3 of Table IV) the value of $E[\tilde{\mathbf{C}}_\mathbf{1}]$ is lower than that for Approach 1. In particular, it can be shown that (via our clustering algorithm) the lowest value of the cost function is obtained using Approach 2 with $\beta = \frac{1}{2}$. Trade-offs are facilitated in favor of (1) by minimizing $E[\tilde{\mathbf{C}}_\mathbf{1}]$ using Approach 2 with $\beta \in (\frac{1}{2}, 1]$. This trade-off is illustrated in greater detail in Fig. 8.

Likewise, (2) is favored by employing Approach 1. For the results in Table IV and Fig. 8, the clustering algorithm was applied to segments of length $M = 20$ bits.

A.3 *Validating the Perceptual Significance*

We applied the two-stage hash algorithm (using Approach 2 with $\beta = \frac{1}{2}$) on a database of 50 images. The final hash length obtained was 46 bits. For each image 20 perceptually identical images were generated using the Stirmark software [23], [39]. The attacks implemented on the images included JPEG compression with quality factors varying from 10 to 80, adding white Gaussian noise (AWGN), enhancing contrast, non-linear (e.g., median) filtering, scaling and random shearing, and small rotation and cropping. The resulting hash values for the original image and its perceptually identical versions were the same in over 95% cases. Then, we compared hash values for all possible pairings of the 50 distinct images (1225 pairs). One collision case was observed[5]. For all other cases the hash values (on a pairwise basis) were very far off. In general, the performance of our hash function is limited by the robustness of the feature detector.

For the same set of images, using an average distance VQ for feature vector compression resulted in about a 70% success rate of mapping perceptually identical versions to the same hash value. In addition, 40 collision cases (same hash value for perceptually distinct images) were observed.

B. *Precision Recall or ROC Analysis*

We now present a detailed statistical comparison of our proposed clustering with the average distance VQ and error correcting decoding using precision-recall (or ROC) curves [40].

[5]The results for the randomized clustering algorithm by appropriately choosing s (detailed in Section VI-C) were very similar to the ones reported here. In particular, we observed the same trend over several different choices of the secret key K_1.

The precision-recall terminology comes from document retrieval, where precision quantifies (for a given query) what fraction of the returned documents are correct. Recall, on the other hand, measures what fraction of the correct documents were returned. Fig. 7 illustrates this scenario. In this case, recall can be improved by simply returning as large a set as possible. This, however, will heavily compromise the precision of the search.

A precision-recall curve illustrates this trade-off and provides valuable insight especially for problems in which absolute maximization of precision and/or recall is possible only via trivial solutions. For our problem in Section II, we employ the notion of pairwise precision [40] in the following manner

$$Prec_\epsilon = \frac{|X_S \cap X_A|}{|X_A|} \tag{18}$$

where $X_S = \{(l_i, l_j) \mid D(l_i, l_j) < \epsilon\}$ is the set of all pairs that should be in the same cluster. X_A then denotes the set of pairs that a given algorithm A puts in the same cluster.

Similarly, pairwise recall

$$Rec_\epsilon = \frac{|X_S \cap X_A|}{|X_S|} \tag{19}$$

Clearly, $0 \leq Prec_\epsilon \leq 1$, $0 \leq Rec_\epsilon \leq 1$ (recall may trivially be made 1 by putting all vectors in the same cluster). Fig. 9 shows an analysis of three algorithms: 1.) average distance VQ, 2.) error correction decoding (ECD) and 3.) the proposed clustering via precision-recall curves. Each point on the curve(s) in Fig. 9 is a precision-recall pair for a particular value of ϵ; i.e., the precision and recall values computed using (18) and (19) when the algorithm is run for that ϵ. As indicated in Fig. 9 we varied ϵ in the range $[0.1, 0.5]$.

Comparing the precison-recall curves for the average distance VQ and ECD we observe that the average distance VQ affords a better recall rate at the cost of loosing precision which is higher for ECD. This explains partially the higher number of collisions in the hash values for perceptually distinct images using the average distance VQ. Note that both the precision as well as recall values are much higher using our proposed clustering algorithm.

Note also that there are three different curves for our proposed clustering algorithm. These correspond to different choices of δ (as a function of ϵ) in our algorithm. The average distance VQ and ECD do not have a δ parameter; hence, we present results of our clustering for different δ to ensure a fair comparison between the three schemes. This also provides insight on how δ may be chosen for a given ϵ (which is typically determined empirically from the feature space) to attain greater flexibility in the precision-recall trade-offs.

In the classical detection theoretic framework, ROC curves are reported as: probability of miss P_M; i.e., probability that perceptually identical images map to different hash values vs. probability of false alarm P_F; i.e., probability that perceptually distinct images map to the same hash value. Note that, $P_M = 1 - Rec_\epsilon$. However, there is no straightforward relationship between P_F

and $(Prec_\epsilon, Rec_\epsilon)$. We presented a precision-recall analysis then for two reasons: 1.) in our experiments the probability of false alarm P_F was too small to be noticed on the ROC curves, and 2.) $Prec_\epsilon$ is a more informative quantity than P_F from the viewpoint of image database search.

C. Security Experiments

An important observation underlying the need for randomization is the fact that feature extraction is seldom perfect. That is by means of thorough analysis it may be possible for an adversary to manipulate image content and yet generate vectors over the feature space that are close. The goal of randomization is hence to make the job of defeating the hash algorithm significantly harder.

A malicious adversary may try to accomplish the same in one of two ways.
1. The adversary may try to generate perceptually identical inputs for which hash algorithm generates different hash values, or
2. The adversary may attempt to tamper with the content so as to cause significant perceptual changes such that the hash algorithm generates the same hash value.
We assume here, that the adversary has complete knowledge of the intermediate hash (or feature) vector extraction as well as the deterministic clustering algorithm for intermediate hash vector compression. Hence, the adversary is capable of analyzing the algorithm and would attempt to generate inputs over the set $E \subset U$, where U represents the set of all possible pairs of intermediate hash vectors and E is the set of intermediate hash vector pairs over which the deterministic clustering algorithm makes errors.

C.1 *Security Via Randomization*

For the results presented next, the randomized clustering algorithm in Section V-B was employed with Approach 2 and $\beta = \frac{1}{2}$. Fig. 10 shows a plot of the cost in (12) computed over the set E against values of s decreasing from ∞ to 0. It can be seen that the cost decreases with s (though not monotonically) and is reduced by orders of magnitude for values of $s < 1000$. Decreasing s is tantamount to increasing randomness. Hence, the plot in Fig. 10 reveals that as randomness is increased beyond a certain level, the adversary meets with very little success by generating input intermediate hash pairs over the set E.

C.2 *Randomness vs. Perceptual Significance Trade-offs*

Let $\bar{E}$ denote the complement set of E; i.e., the set of all intermediate hash vector pairs over which no errors are made by the deterministic clustering algorithm. Fig. 11 then shows the plot of the clustering cost function against decreasing s as before. In this case, the cost increases with decreasing s (again not monotonically). As $s \to \infty$ the cost is zero since the deterministic clustering algorithm makes no errors over the set $\bar{E}$.

Fig. 12 shows a sum of the cost in the two plots in Figs. 10 and 11. This plot therefore shows the total cost computed over the set U as a function of s. Figs. 13 and 14 respectively show the same cost function plots as in Figs. 11 and 12 but with the $y-$axis in log-scale. As s approaches 0, the value of the cost is increased significantly over the cost incurred by the deterministic algorithm. The cost achieved by the deterministic algorithm is the value of the cost function in Fig. 12 (or Fig. 14) as $s \to \infty$ and equal to 7.43×10^{-9}. At $s = 0$, the total cost is 6.12×10^{-5}. This increase is intuitive as complete randomness (i.e., $s = 0$) would affect the perceptual qualities of the hash.

It is of interest to observe the values of the cost function in Fig. 12 for $40 < s < 1000$. This region is zoomed in and plotted in Fig. 15. It can be observed from Fig. 15 that the total cost is of the order of the cost incurred by the deterministic algorithm. Further, we know from Fig. 10 that the cost over the set E for $s < 1000$ decreases to the extent that the adversary cannot gain anything by generating input pairs on this set. By choosing a value of s in this range we can largely retain the perceptual qualities of the hash and also reduce the vulnerability of the hash algorithm to malicious inputs generated by the adversary.

C.3 *Distribution of Final Hash Values*

Finally, we evaluate our success in meeting the third desired property of the hash, i.e. the closeness to uniform distribution. We employ the widely used Kullback Leibler (KL) distance [31] given by

$$D(h||u) = \sum_{x \in C} h(x) \log \frac{h(x)}{u(x)} \tag{20}$$

where $C = \{x : h(x) > 0\}$ represents the support set of $h(x)$. Here $h(x)$ denotes the distribution of hash values generated by our algorithm and $u(x)$ denotes the uniform distribution over the set C. The set C was obtained by generating the hash values for a given image used in our experiments over the key space (of K_1).

Fig. 16 shows the plot of the KL measure against values of s decreasing from ∞ to 0. Even as $s \to \infty$ this value is pretty low (≈ 0.2) and for $s < 1000$ i.e., the desired range for secure hashing we achieve a near uniform distribution. Very similar results were observed for all of the 50 images in our experiments.

VII. Discussions & Conclusion

This paper presents a two-stage framework for perceptually-based image hashing. The framework facilitates trade-offs between robustness and fragility of the hash. Within the framework, we present a randomization scheme for secure hashing.

In the framework, the first step extracts visually significant features from an image to produce an intermediate hash. A variety of feature detectors may be applied. The second step clusters perceptually identical inputs while minimizing (in a rigorous sense) the likelihood of collision for perceptually distinct inputs to compress the intermediate hash to a final hash. The clusters are invariant for perceptually identical images with a very high probability.

One possible future direction is in audio hashing. Since the second step is media independent, an appropriate feature detector may be applied in the first step to make the

framework applicable to other media data sets. Another possible future direction is to analyze the secure perceptual media (image/audio) hashing problem formally in a game theoretic setting.

References

[1] A. Menezes, V. Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1998.

[2] M. Schneider and S. F. Chang, "A robust content based digital signature for image authentication," *Proc. IEEE Conf. on Image Processing*, vol. 3, pp. 227–230, Sept. 1996.

[3] C. Kailasanathan and R. S. Naini, "Image authentication surviving acceptable modifications using statistical measures and k-mean segmentation," *IEEE-EURASIP Work. Nonlinear Sig. and Image Processing*, vol. 1, June 2001.

[4] C. Y. Lin and S. F. Chang, "Generating robust digital signature for image/video authentication," *Proc. ACM Multimedia and Security Workshop*, Sept. 1998.

[5] C. Y. Lin and S. F. Chang, "A robust image authentication system distingushing JPEG compression from malicious manipulation," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, pp. 153–168, Feb. 2001.

[6] S. Bhatacherjee and M. Kutter, "Compression tolerant image authentication," *Proc. IEEE Conf. on Image Processing*, vol. 1, pp. 435–439, 1998.

[7] J. Dittman, A. Steinmetz, and R. Steinmetz, "Content based digital signature for motion picture authentication and content-fragile watermarking," *Proc. IEEE Int. Conf. on Multimedia Computing and Systems*, vol. 2, pp. 209–213, 1999.

[8] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *IEEE Trans. on Image Processing*, vol. 6, no. 12, pp. 243–246, Dec. 1996.

[9] E. T. Lin and E. J. Delp, "A review of fragile image watermarks," *Proc. ACM Multimedia and Security Workshop*, vol. 1, pp. 25–29, Oct. 1999.

[10] M. M. Yeung and F. Mintzer, "An invisible watermarking scheme for image verification," *Proc. IEEE Conf. on Image Processing*, vol. 1, pp. 680–683, Oct. 1997.

[11] M. Wu and B. Liu, "Watermarking for image authentication," *Proc. IEEE Conf. on Image Processing*, vol. 2, pp. 437–441, Oct. 1998.

[12] R. B. Wolfgang and E. J. Delp, "Fragile watermarking using the VW2D watermark," *Proc. SPIE/IS&T Int. Conf. Security and Watermarking of Multimedia Contents*, pp. 204–213, Jan. 1999.

[13] L. Xie and G. R. Arce, "A class of authentication digital watermarks for secure multimedia communication," *IEEE Trans. on Image Processing*, vol. 10, no. 11, pp. 1754–1764, Nov. 2001.

[14] G. L. Friedman, "The trustworthy digital camera: restoring credibility to the photographic image," *IEEE Trans. on Consumer Electronics*, vol. 39, pp. 905–910, Nov. 1993.

[15] M. K. Mihcak and R. Venkatesan, "Video watermarking using image hashing," *Microsoft Research Technical Report*, Jan. 2001.

[16] R. Venkatesan, S. M. Koon, M. H. Jakubowski, and P. Moulin, "Robust image hashing," *Proc. IEEE Conf. on Image Processing*, vol. 3, pp. 664–666, Sept. 2000.

[17] J. Fridrich and M. Goljan, "Robust hash functions for digital watermarking," *Proc. IEEE Int. Conf. on Information Technology: Coding and Computing*, pp. 178–183, Mar. 2000.

[18] K. Mihcak and R. Venkatesan, "New iterative geometric techniques for robust image hashing," *Proc. ACM Workshop on Security and Privacy in Digital Rights Management*, Nov. 2001.

[19] C.-S. Lu and H.-Y. M. Liao, "Structural digital signature for image authentication," *IEEE Trans. on Multimedia*, vol. 5, pp. 161–173, June 2003.

[20] V. Monga and B. L. Evans, "Robust perceptual image hashing using feature points," *Proc. IEEE Conf. on Image Processing*, vol. 1, pp. 677–680, Oct. 2004.

[21] M. Johnson and K. Ramachandran, "Dither-based secure image hashing using distributed coding," *Proc. IEEE Conf. on Image Processing*, vol. 3, pp. 14–17, Sept. 2003.

[22] V. Monga, A. Banerjee, and B. L. Evans, "Clustering algorithms for perceptual image hashing," *Proc. IEEE Digital Sig. Processing Workshop*, pp. 283–287, Aug. 2004.

[23] "Fair evaluation procedures for watermarking systems," http://www.petitcolas.net/fabien/watermarking/stirmark, 2000.

[24] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic, 1991.

[25] X. Wu, "Adaptive binary vector quantization using hamming codes," *Proc. IEEE Conf. on Image Processing*, vol. 3, pp. 93–96, Oct. 1995.

[26] P. Franti and T. Kaukoranta, "Binary vector quantizer design using soft-centroids," *Signal Processing: Image Communication*, vol. 14, pp. 677–681, Sept. 1999.
[27] B. Julesz, "Visual pattern discrimination," *IEEE Trans. on Information Theory*, vol. 8, pp. 84–92, Feb. 1962.
[28] J. I. Yellott, "Images, statistics and textures: Implications of triple correlation uniqueness for texture statistics and the julesz conjecture," *Journal of Optical Society of America*, vol. 10, pp. 777–793, Oct. 1993.
[29] S. Zhu, Y. N. Wu, and D. Mudford, "Filters, random fields and maximum entropy (frame) - towards the unified theory for texture modeling," *ACM Int. Journal of Computer Vision*, vol. 27, pp. 107–126, Mar. 1998.
[30] J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *Kluwer Int. Journal of Computer Vision*, vol. 40, pp. 49–71, Jan. 2000.
[31] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley Interscience, 1998.
[32] P. Indyk and R. Motwani, "Approximate nearest neighbor: towards removing the curse of dimensionality," *Proc.* 30^{th} *Annu. ACM Symp. Comput. Geometry*, pp. 604–613, May 1998.
[33] P. Indyk, *High-dimensional computational geometry*, PhD Thesis, Stanford University, 2001.
[34] J. E. Goodman and J. O'Rourke, *Handbook of Discrete and Computational Geometry*, CRC Press, 1997.
[35] M. L. Fredman, J. Komlos, and E. Szemeredi, "Storing a sparse table with O(1) worst case access time," *Journal of the ACM*, vol. 31, pp. 538–544, June 1984.
[36] M. L. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. on Mathematical Software*, pp. 209–226, Sept. 1977.
[37] A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes*, McGraw Hill College Series, 2000.
[38] R. E. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley Publishing Company, 1983.
[39] F. A. P. Petitcolas and R. J. Anderson, "Evaluation of copyright marking systems," *Proc. IEEE Int. Conf. on Multimedia Systems*, pp. 574–579, June 1999.
[40] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*, Springer-Verlag, 2001.
[41] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., 1979.

Appendix

I. Proof of NP-completeness

In this section, we prove that a decision version of the clustering problem that asks if it is possible to have a k-clustering such that the cost function in (13) is below a certain constant is NP-complete. We achieve this by a reduction (details skipped for brevity) to the decision version of the k-way graph-cut problem [41].

Proof. (Sketch) Let $G = (V, E)$ be a graph where V is the set of vertices and E is the set of edges. It is useful to think of V as the set of points to be clustered, and the edge e_{ij} between v_i and v_j denoting unit distance between the points v_i and v_j. The k-way graph-cut problem asks if there is a subset $A \subseteq E$ of edges with $|A| \leq K_0$, where K_0 is a constant, such that the graph $G' = (V, E \setminus A)$ has k pairwise disjoint subgraphs. We sketch a reduction of (the decision version) of the clustering problem in (13) for a fixed k to the graph-cut problem, i.e., we show that the input to the graph-cut problem can be converted into the input to the clustering problem and that solving the clustering problem yields a solution to the graph-cut problem. Since the graph-cut problem in NP-complete, it will follow that the clustering problem in (13) is NP-complete as well. The reduction is quite simple: Consider the clustering problem with $D(l_i, l_j) \leq \delta$ for a suitably large choice of δ. Further, the distance function is such that $D(l_i, l_j) < \epsilon$ if $e_{ij} \in E$. Then, choosing either $\Gamma_1 = 1$ or $\alpha_1 = 0$, we have $s1(i, j) = |E|$. Hence, for any clustering C, the normalized cost matrix $\tilde{C}_1$ is given by $\tilde{c}_1(i, j) = 1/|E|$ if $e_{ij} \in E$ and $C(l_i) = C(l_j)$, and 0 otherwise. Note that $\tilde{C}_2$ is a zero matrix since all points are within δ of one another. Assuming the intermediate hash distribution p to be uniform, i.e., $p(i) = \frac{1}{n}$ where n is the total number of intermediate hash vectors, the cost associated with the clustering as in (13) is proportional to the cost of the k-way cut with a proportionality constant of $\frac{1}{n^2|E|}$. Hence, the question whether there is a k-way cut A with $|A| \leq K_0$ exists is same as asking if there is a clustering C whose cost is less than $\frac{K_0}{n^2|E|}$, and the solution of the latter gives a solution of the former. That completes the reduction.

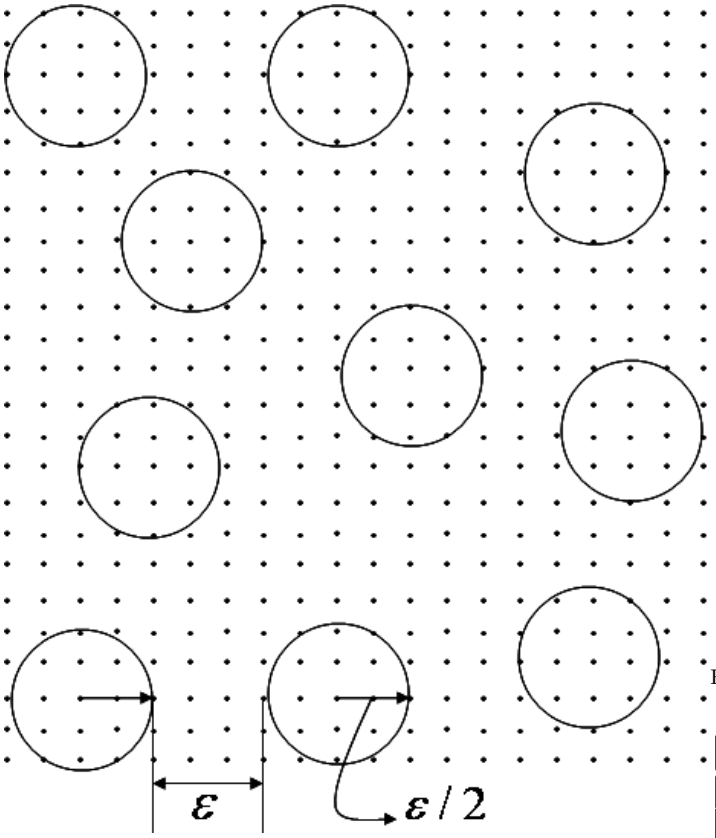

Fig. 5. Visualization of the Clustering Algorithm

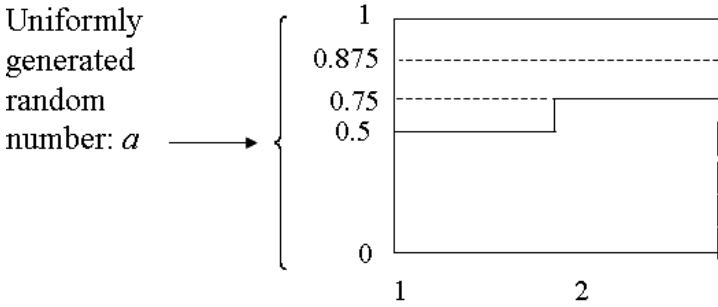

Fig. 6. Example: Selection of data points as cluster centers in a probabilistic sense

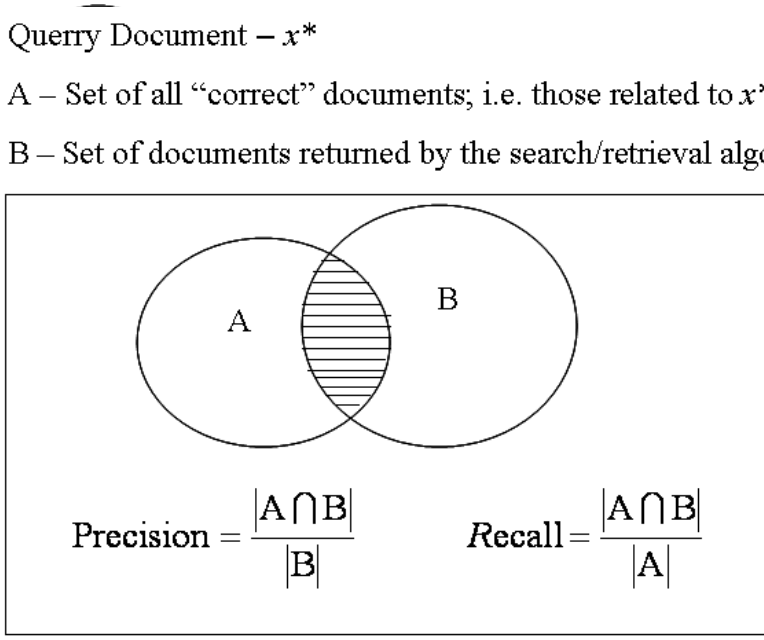

Fig. 7. Illustration of Precision and Recall in a document retrieval scenario

M	$E[\tilde{\mathbf{C}}_1]$	$E[\tilde{\mathbf{C}}_2]$	*Final Hash Length*
8	1.86×10^{-5}	2.372×10^{-7}	102 bits
16	1.219×10^{-7}	5.70×10^{-9}	54 bits

TABLE I

COMPRESSION OF INTERMEDIATE HASH VECTORS USING THE PROPOSED CLUSTERING. M IS THE SEGMENT LENGTH IN BITS. $\tilde{\mathbf{C}}_1$ AND $\tilde{\mathbf{C}}_2$ ARE DEFINED IN (10) AND (11) RESPECTIVELY. $E[\tilde{\mathbf{C}}_1]$ AND $E[\tilde{\mathbf{C}}_2]$ REPRESENT THE MEASURES OF VIOLATING DESIRABLE HASH PROPERTIES IN (1) AND (2) RESPECTIVELY.

M	$E[\tilde{\mathbf{C}}_1]$	$E[\tilde{\mathbf{C}}_2]$	*Final Hash Length*
8	1.526×10^{-3}	5.55×10^{-4}	120 bits
16	9.535×10^{-2}	6.127×10^{-3}	75 bits
16	5.96×10^{-4}	3.65×10^{-5}	165 bits

TABLE II

COMPRESSION OF INTERMEDIATE HASH VECTORS USING ERROR CONTROL DECODING. M IS THE SEGMENT LENGTH IN BITS. $\tilde{\mathbf{C}}_1$ AND $\tilde{\mathbf{C}}_2$ ARE DEFINED IN (10) AND (11) RESPECTIVELY. $E[\tilde{\mathbf{C}}_1]$ AND $E[\tilde{\mathbf{C}}_2]$ REPRESENT THE MEASURES OF VIOLATING DESIRABLE HASH PROPERTIES IN (1) AND (2) RESPECTIVELY.

M	$E[\tilde{\mathbf{C}}_1]$	$E[\tilde{\mathbf{C}}_2]$	*Final Hash Length*
8	1.44×10^{-3}	5.88×10^{-4}	120 bits
16	3.65×10^{-4}	7.77×10^{-5}	60 bits

TABLE III

COMPRESSION OF INTERMEDIATE HASH VECTORS USING A CONVENTIONAL AVERAGE DISTANCE VQ. M IS THE SEGMENT LENGTH IN BITS. $\tilde{\mathbf{C}}_1$ AND $\tilde{\mathbf{C}}_2$ ARE DEFINED IN (10) AND (11) RESPECTIVELY. $E[\tilde{\mathbf{C}}_1]$ AND $E[\tilde{\mathbf{C}}_2]$ REPRESENT THE MEASURES OF VIOLATING DESIRABLE HASH PROPERTIES IN (1) AND (2) RESPECTIVELY.

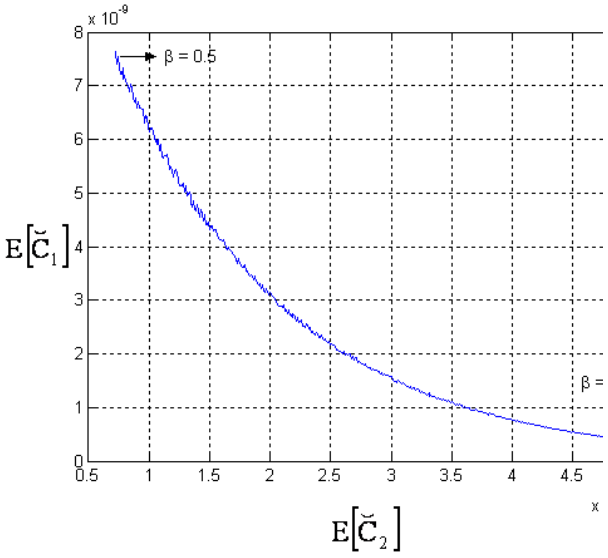

Fig. 8. Perceptual robustness vs. fragility trade-offs by varying $\beta \in [\frac{1}{2}, 1]$.

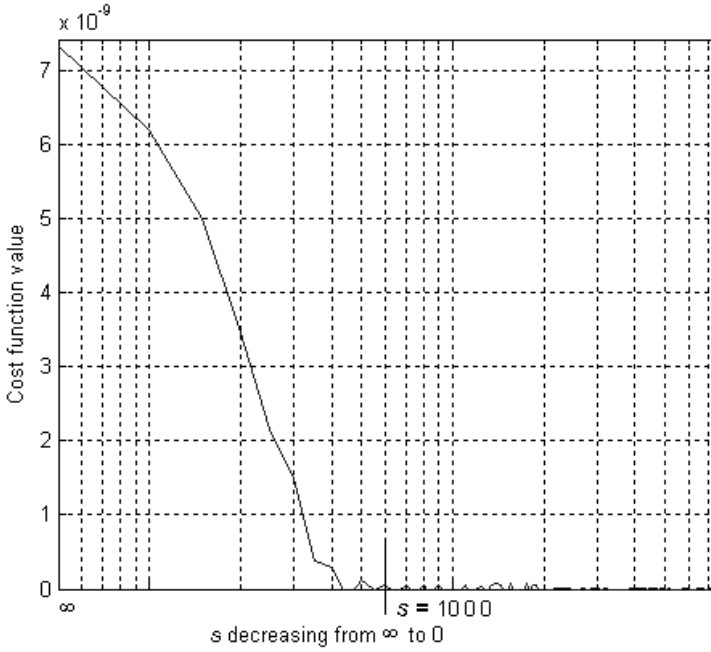

Fig. 10. Clustering cost function computed over the set E. E is the set of intermediate hash vector pairs over which the deterministic clustering makes errors and s is the randomization parameter.

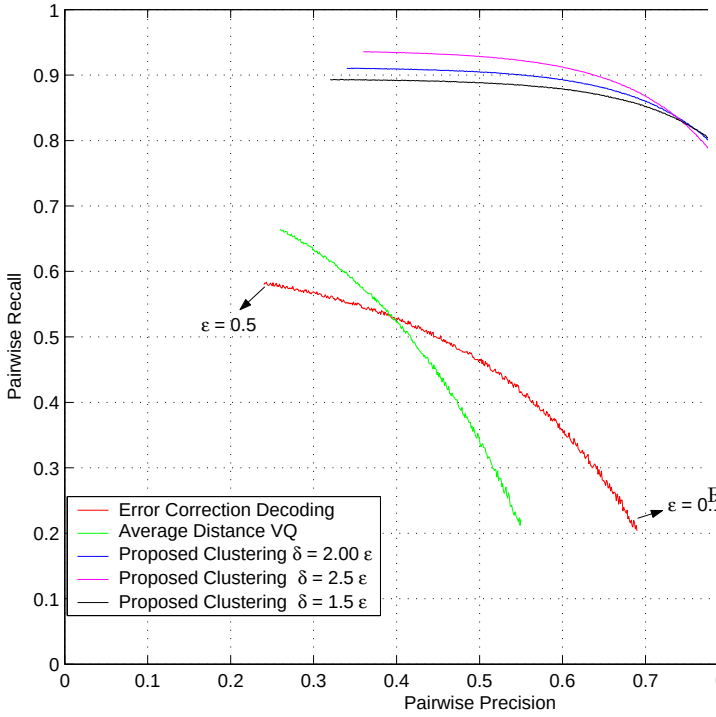

Fig. 9. Precision-recall curves for three compression approaches: traditional VQ, Error Correction Decoding, Proposed Clustering

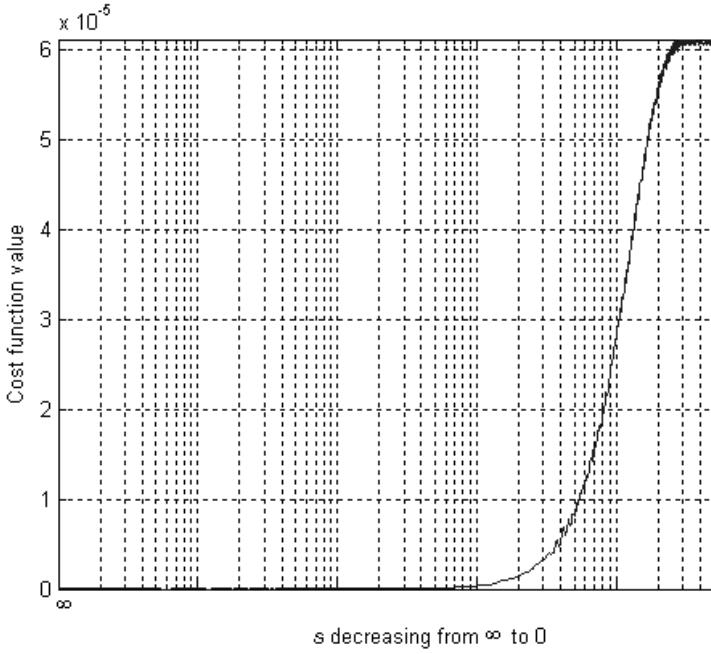

Fig. 11. Clustering cost function over the set $\bar{E}$. $\bar{E}$ denotes the complement set of E and s is the randomization parameter.

Clustering Algorithm	$E[\tilde{\mathbf{C}}_1]$	$E[\tilde{\mathbf{C}}_2]$
Approach 1	7.64×10^{-8}	0
Approach 2, $\beta = \frac{1}{2}$	7.43×10^{-9}	7.464×10^{-10}
Approach 2, $\beta = 1$	4.17×10^{-10}	4.87×10^{-8}

TABLE IV
COST FUNCTION VALUES USING APPROACHES 1 AND 2 WITH TRADE-OFFS NUMERICALLY QUANTIFIED.

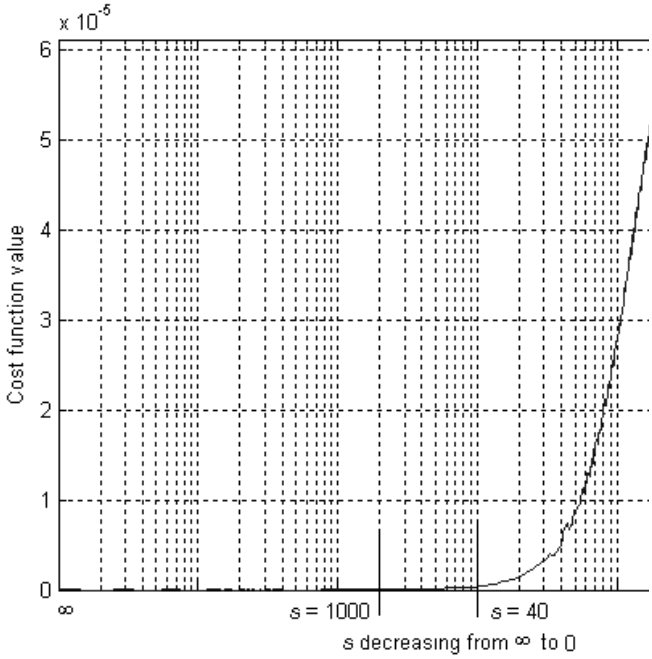

Fig. 12. Clustering cost function over the complete set U of intermediate hash pairs. $U = E \cup \bar{E}$ and s is the randomization parameter.

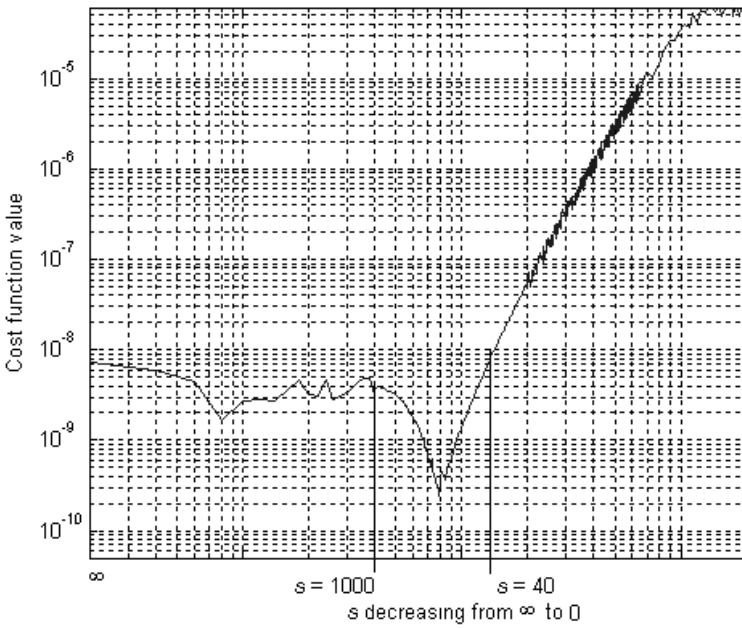

Fig. 14. Clustering cost function over the complete set U with the vertical axis on a log scale to show more detail of Fig. 12.

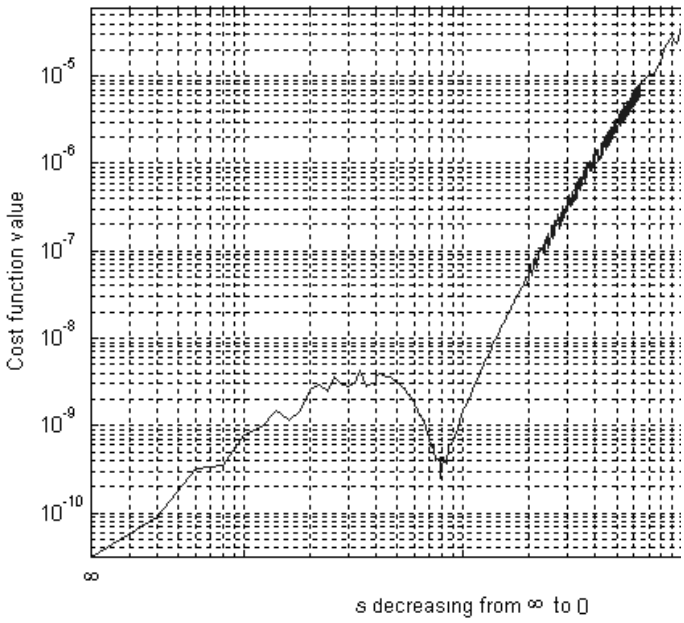

Fig. 13. Clustering cost function over the set $\bar{E}$ with the vertical axis on a log scale to show more detail of Fig. 11.

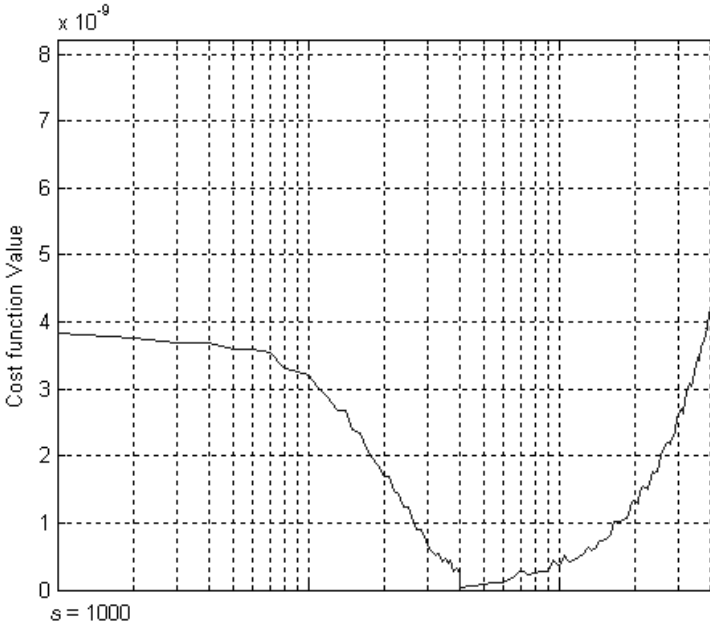

Fig. 15. Clustering cost function over the set U of intermediate hash pairs in the region $40 < s < 1000$

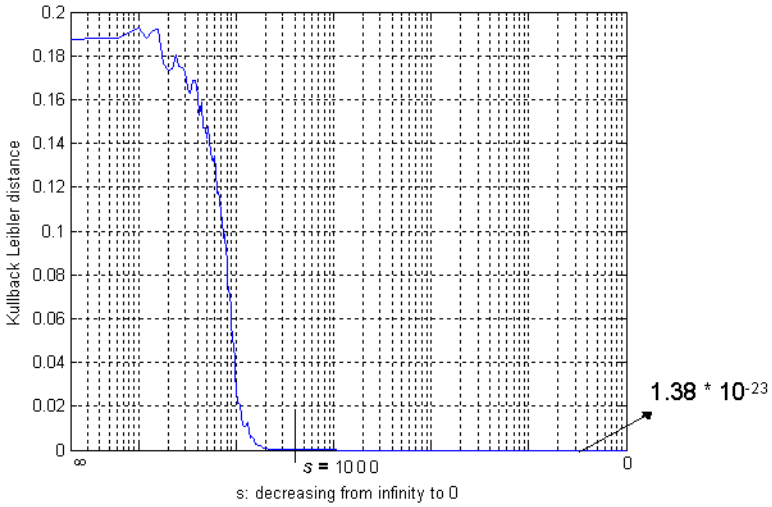

Fig. 16. Kullback-Leibler distance of the hash distribution measured with the uniform distribution as the reference. Here s is the randomization parameter.