CLUSTERING ALGORITHMS FOR PERCEPTUAL IMAGE HASHING

Vishal Monga, Arindam Banerjee, and Brian L. Evans

Center for Perceptual Systems
The University of Texas at Austin, Austin, TX 78712
{vishal,abanerje,bevans}@ece.utexas.edu

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). 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. We prove that the decision version of our clustering problem is NP complete. Based on the proposed algorithm, we develop two variations to facilitate perceptual robustness vs. fragility trade-offs. We test the proposed algorithms against Stirmark attacks.

1. 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 hash function should have the property that two images that look the same map to the same hash value, even if the images have small bit-level differences. 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. Other applications lie in the area of authentication [2] and watermarking [3, 4].

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 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 the feature vector extraction, many algorithms could be used, e.g. [5, 6, 7, 8, 9]. For the rest of the paper, we will refer to this visually robust feature vector (or its quantized version) as the "intermediate hash".

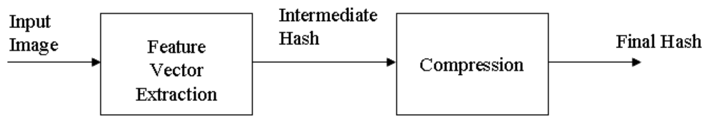

Figure 1: Block diagram of the Hash Function

The second step takes an intermediate hash extracted from the image and an estimate of the hash's distribution, and compresses the intermediate hash value to the final hash value. Many algorithms have been proposed for this second step, including those based on error correcting codes [7] and secure compression for authentication applications [10]. While compression is their primary goal [7], [10], no *explicit* attempt is made to ensure that perceptually identical images are compressed to the same hash value.

The second step will involve clustering between the intermediate hash vector of an input source (image) and the intermediate hash vectors of its perceptually identical versions. We develop a clustering algorithm based on the distribution of intermediate hash vectors to address exactly this problem. Another important issue is the length (or granularity) of the final hash required to cluster images within a specified distance. Underestimating this length can severely affect the perceptual qualities of the hash. A significant contribution of our work is that this length is determined naturally as an outcome of our proposed clustering algorithm.

2. PROBLEM STATEMENT

Let V denote the metric space of intermediate hash vectors extracted at stage 1 of the hash algorithm in Fig. 1 and (l_i, l_j) denote two arbitrary vectors in this space. Further, let $D(.,.)$ denote the distance metric applicable to the vectors in V. Our goal is to have all images that are visually indistinguishable map to the same hash value. In that sense an image hash function is similar to a *vector quantization* (VQ) or *clustering* scheme. We are attempting to cluster

Research was supported by a gift from the Xerox Foundation.

all images whose intermediate hash vectors are close in a metric into the same cell. In particular, it is desired that

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

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

where $0 < \epsilon < \delta$. $C(l_j)$ represent the clusters to which these vectors map after applying the clustering algorithm.

3. 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 $(l_i,\ l_j)$. The requirement in (1) is actually impossible to guarantee for every input pair. 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 V.

We now describe the construction of our clustering cost function. Let $\mathbf{P}$ be the joint distribution matrix of each pair $l_i, l_j)$ in the input space where $\mathbf{P}_{ij} = p(i,j) = p(i)p(j)$. $p(i)$, $p(j)$ respectively denote the probability of occurrence of vectors l_i, l_j and n is the total number of vectors to be clustered.

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, $\forall\ i, j = 1, 2, ..., n$

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

where $\alpha > 0$ and $\Gamma > 1$ 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)

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

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. The exponential cost ensures that errors associated with large distances are penalized severely.
Further, let matrices $\mathbf{S}_1$ and $\mathbf{S}_2$ be defined as

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

$$s2(i,j) = \begin{cases} \Gamma^{(\alpha D(l_i,l_j))} & \text{if } D(l_i,l_j) > \delta \\ 0 & \text{otherwise} \end{cases}$$

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, while $\mathbf{C}_1$ is the cost matrix for the errors actually made by the clustering algorithm. The same holds for $\mathbf{S}_2$ and $\mathbf{C}_2$. Then, we normalize the entries in $\mathbf{C}_1$ and $\mathbf{C}_2$ to define normalized cost matrices $\tilde{\mathbf{C}}_1$ and $\tilde{\mathbf{C}}_2$ such that

$$\tilde{c}_1(i,j) = \frac{c_1(i,j)}{\sum_i \sum_j s_1(i,j)} \tag{3}$$

$$\tilde{c}_2(i,j) = \frac{c_2(i,j)}{\sum_i \sum_j s2(i,j)} \tag{4}$$

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

$$P_{err} = E[\tilde{\mathbf{C}}_1 + \tilde{\mathbf{C}}_2] \tag{5}$$

The expectation is taken over the joint distribution of (l_i, l_j), i.e. (5) may be rewritten as

$$P_{err} = \sum_i \sum_j p(i)p(j)\left(\tilde{c}_1(i,j) + \tilde{c}_2(i,j)\right) \tag{6}$$

The two additive terms in (5) i.e. $E[\tilde{\mathbf{C}}_1]$, $E[\tilde{\mathbf{C}}_2]$ 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 violation of (2). Note for the particular case of $\alpha = 0$, $E[\mathbf{C}_1]$ and $E[\mathbf{C}_2]$ represent the total probability of violating (1) and (2) respectively.

4. PROPOSED CLUSTERING ALGORITHMS

Finding the optimum clustering that would achieve a global minimum for the cost function in (5) 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. NP 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.

4.1. Approach 0

For the following discussion vectors in the input space V will be referred to as "data points". A step by step description of the algorithm follows.

1. Obtain user defined parameters ϵ, δ. Set $k = 1$
2. Select the data point associated with the highest probability mass, 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 amongst 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.

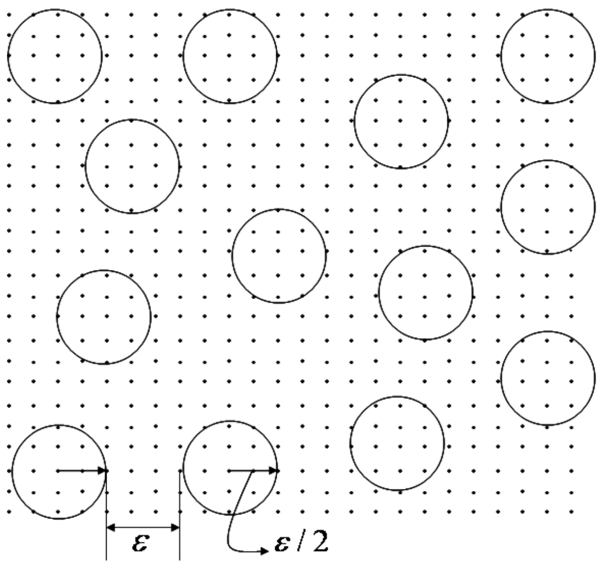

Figure 2: Visualization of the Clustering Algorithm

Fig. 2 shows a visualization of the clustering algorithm. 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

$$D(l_i, l_j) < D(l_i, l^k) + D(l^k, l_j) \qquad (7)$$

This is true because $D(,.,)$ defines a metric. By virtue of Steps 3 and 6 of the 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 "more likely" as well as "perceptually close" data points are clustered together. At this stage, we make the following observations:

- Each cluster is at least ϵ away from any other cluster and hence there are no errors by violating (1), Step 4. of the algorithm guarantees this.
- 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 each of the clusters

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 goals (1) and (2).

4.2. Approach 1

1. Select the data point l^* amongst 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 $k = k + 1$. $S^k = l^*$ 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. through 3.

Step 3 of the algorithm looks for the set of clusters $\mathcal{S}_\delta$, such that each point in the cluster is less than δ away from the unclustered data point l^* under consideration. Step 4 then computes the minimum cost cluster to which to assign l^*. In essence, this approach tries to minimize the cost in (5) conditioned on the fact that there are no errors by violating (2). This could be useful in authentication and security applications in which mapping perceptually distinct inputs to the same hash may be extremely undesirable.

4.3. Approach 2

1. Select the data point l^* amongst 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 and 2.

Approach 1 presents a clustering that ensures $E[\tilde{\mathbf{C}}_2] = 0$. The goal in Approach 2 is to effectively trade-off the minimization of $E[\tilde{\mathbf{C}}_1]$ at the expense of increasing $E[\tilde{\mathbf{C}}_2]$. 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 $\delta \geq \frac{5}{2}\epsilon$, the two approaches coincide because all the unclustered points are then necessarily within δ of the existing clusters. Hence, 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.

Traditional VQ or source coding based compression approaches [12, 13, 14] are not well suited because they tend

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 1: Cost Function values, compression of intermediate hash vectors using the proposed clustering

to minimize an average or maximum distance value which is not an appropriate objective function for the perceptual hashing application. In standard VQ based compression approaches, the size of the codebook (here, the length of the hash) is decided in advance and optimization is performed to select the best codevectors. 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.

5. EXPERIMENTAL RESULTS

An intermediate hash vector extracted from an image I will be referred to as $\mathbf{fv}(I)$. Let I_{sim} represent the class of images such that I_{sim} looks the same as I. Likewise, a perceptually distinct image will be denoted by I_{diff}. In the presented experiments, the intermediate hash vector was generated using the method by Monga *et. al* in [9]. 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 by empirical tests

$$D(\mathbf{fv}(I), \mathbf{fv}(I_{sim})) < 0.2 \qquad (8)$$

$$D_H(\mathbf{fv}(I), \mathbf{fv}(I_{diff})) > 0.3 \qquad (9)$$

In our clustering framework, $\epsilon = 0.2$ and $\delta = 0.3$.

In our experiments, we extract a binary intermediate hash vector of length $L = 240$ bits from the image and hence the total number of vectors, $n = 2^{240}$. Because of space and complexity constraints it is clearly impractical to apply the clustering algorithm to that large a data set. Hence, we take the approach commonly employed in space constrained VQ problems [12] i.e. we divide the intermediate hash vector into segments of a certain 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 [7]. They employ error control decoding using Reed-Muller codes [15]. 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 (EPN) [7].

Tables 1 and 2, respectively, show values of the cost function in (5) by compressing the intermediate hash vector using our proposed clustering scheme vs. the error control decoding scheme as described in [7]. We generated the results in Table 1 by using Approach 2 and $\beta = \frac{1}{2}$. For

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 2: Cost function values with compression of intermediate hash vectors using error control decoding.

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, $\beta = 1$	$7.17 * 10^{-9}$	$4.87 * 10^{-9}$

Table 3: Cost function values using Approaches 1 and 2 with trade-offs numerically quantified.

the error control decoding scheme, we use (8,4), (16,5) and (16,11) Reed-Muller codes. The clustering algorithm was also employed on segments of the same length to yield a meaningful comparison. Clearly, the values for expected cost by violating (1) and (2), i.e. $E[\tilde{\mathbf{C}}_1]$ and $E[\tilde{\mathbf{C}}_2]$, are orders of magnitude lower using our clustering algorithm (even as we achieve better compression). Hence, we show that the codebook as obtained from using error correcting codes does not fare well for perceptual hash compression.

Table 3 compares the value of the cost function in (5) for the two different clustering approaches. Note, for Approach 2 (rows 2 and 3 of Table 3) the value of $E[\tilde{\mathbf{C}}_1]$ 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) i.e. minimizing $E[\tilde{\mathbf{C}}_1]$ by using Approach 2 with $\beta \in (\frac{1}{2}, 1]$ and in favor of (2) by employing Approach 1. For these results, the clustering algorithm was applied to segments of length $M = 20$ bits.

We applied the two-stage hash algorithm 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 [16], [17]. The attacks implemented on the images included JPEG compression with quality factors varying from 10 to 80, additive white Gaussian noise (AWGN) addition, contrast enhancement, nonlinear (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 same in over 95% cases. For other cases, the maximum bit-level difference was 4 bits. Then we compared hash values for all possible pairings of the 50 distinct images (1225 pairs). One collision case was observed. 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.

In our experiments, we observed that some segments of the source vector exhibit very "skewed" distributions. In that case it is possible that trivial clusters be formed i.e. only very low probability points would be included in the cluster. For efficient compression, it is important that the number of clusters formed in our algorithm accurately reflects the statistics of the source. A slight modification to the basic algorithm as presented in Section 4 can be used to achieve this. In particular, consider the algorithm at a stage when m clusters are formed with i points already clustered. Note $i < n$ and $m < k$ where k is the number of clusters that results from the algorithm in Section 4. We can assign the remaining data points $i+1, ..., n$ to the existing clusters in the same fashion as the basic algorithm and then compute the cost function in (5) for this clustering. If the increase in cost, as compared to what is achieved by the algorithm in Section 4, is not significant, then the clustering algorithm terminates with the existing number of clusters.

Finally, the clustering algorithm can be used to compress feature vectors that are close in a arbitrary distance metric. For real valued feature vectors, the number of data points n should be chosen large enough to sufficiently represent source (feature vector) statistics. A codebook (comprising of cluster centers) can then be derived from these data points using the proposed clustering and feature vectors can be mapped to the nearest vector in the codebook based on a minimum cost decoding rule.

6. REFERENCES

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

[2] C. W. Wu, "On the design of content-based multimedia authentication systems," *IEEE Transactions on Multimedia*, pp. 385–393, 2002.

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

[4] M. K. Mihcak and R. Venkatesan, "Video watermarking using image hashing," *Preprint*, 2001.

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

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

[7] R. Venkatesan, S. M. Koon, M. H. Jakubowski, and P. Moulin, "Robust image hashing," *Proc. IEEE Conf. on Image Processing*, Sept. 2000.

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

[9] V. Monga and B. L. Evans, "Robust perceptual image hashing using feature points," *Proc. IEEE Conf. on Image Processing*, submitted.

[10] M. Johnson and K. Ramachandran, "Dither-based secure image hashing using distributed coding," *Proc. IEEE Conf. on Image Processing*, 2003.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of Np-Completeness*, W H Freeman & Co, 1979.

[12] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic, 1992.

[13] X. Wu, "Adaptive binary vector quantization using hamming codes," *Proc. IEEE Conf. on Image Processing*, 1995.

[14] P. Franti and T. Kaukoranta, "Binary vector quantizer design using soft-centroids," *Signal Processing: Image Communication*, pp. 677–681, 1999.

[15] R. E. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley Publishing Company, 1983.

[16] F. A. P. Petitcolas and R. J. Anderson, "Evaluation of copyright marking systems," *Proc. IEEE Int. Conf. on Multimedia Systems*, pp. 574–579, 1999.

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

A. 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 (6) is below a certain constant is NP-complete. We achieve this by a reduction (details skipped for brevity) from the decision version of the k-way weighted graph-cut problem [11].

Proof. (Sketch) Let $G = (V, W(E))$ be a weighted graph where V is the set of vertices, E is the set of edges and $W(E)$ denote the weights on the edges. It is useful to think of V as the set of points be clustered, and the weight $W(e_{ij})$ on the edge e_{ij} between v_i and v_j as the distance between the points v_i and v_j. The k-way weighted graph-cut problem asks if there is a subset $C \subseteq E$ of edges with $\sum_{e \in C} W(e) \leq K_0$, where K_0 is a constant, such that the graph $G' = (V, W(E \backslash C))$ has k pairwise disjoint subgraphs. We sketch a log-space reduction to the clustering problem in (6) for a fixed k. We construct a graph $\tilde{G} = (V, \tilde{W})$ from G as follows: Consider each possible vertex pair (v_i, v_j) with $i, j = 1, \ldots, n$. Denote $w_{ij} = W(e_{ij})$. If $w_{ij} < \epsilon$, $\tilde{w}_{ij} = K_1 c_1(i,j)$, where $c_1(i,j)$ is as defined in section 3 with $D(l_i, l_j) = w_{ij}$, and K_1 is a positive constant. If $w_{ij} > \delta$, $\tilde{w}_{ij} = -K_2 c_2(i,j)$, where $c_2(i,j)$ is as defined in section 3 with $D(l_i, l_j) = w_{ij}$, and K_2 is a positive constant. For $\epsilon \leq w_{ij} \leq \delta$, $\tilde{w}_{ij} = 0$. Consider the same k-way graph-cut problem on $\tilde{G}$. Let $\tilde{C}$ be a subset of the edges. For edges in $\tilde{C}$ with positive w_{ij}, the sum of the weights, say S_1, directly correspond to the sum of the $c_1(i,j)$ terms in (6). For edges in $\tilde{C}$ with negative weights, the sum of the weights, say S_2 is negative. Let $-N, N > 0$, denote the sum of all negative weights in $\tilde{W}$. Now, $N + S_2$ is the sum of the weights in $\tilde{W} \setminus \tilde{C}$, that exactly corresponds to the sum of the $c_2(i,j)$ terms in (6). Hence, $N + S_1 + S_2$ corresponds to the cost function in (6) up to an additive constant, when the $p(i)$ is uniform. Note that only constant number of indices of the vertices, which need $O(\log n)$ space, must be maintained to complete the reduction. Hence, the k-way weighted graph-cut reduces to the clustering problem in log-space. ■