Image Hashing

Ramarathnam Venkatesan and Mariusz H. Jakubowski
Microsoft Research

March 24, 2000

Abstract

We present new algorithms for hashing, or one-way image compression, and comparison of bitmapped images. Our methods are based on random multiscale subdivision of images into regions, randomized rounding of intensity averages in those regions, and robust compression of the resulting vectors by error correction. As hashes useful for robust identification and comparison of images, the compressed vectors can replace watermarks. Our schemes work with images subjected to common distortions, including scanning, resizing, and resampling. Additionally, image hashes withstand anti-watermark transformations performed by software such as StirMark and unZign.

1 Introduction

We describe several new algorithms for transforming bitmapped images into short hash vectors, which can be used to identify and compare images. This may be viewed as *one-way image compression*, where the image is compressed using randomly chosen, not necessarily invertible basis functions. The scheme allows us to compare two images and obtain an image-closeness measure satisfying the following criteria:

1. Every image should produce a randomly distributed value.
2. For two images that are "visually close enough," the values must be close.
3. For two not-so-close images, the outputs must be uncorrelated.

Our techniques allow comparisons even if an adversary maliciously transforms an image into another "visually equivalent" image with the hope of convincing the comparison algorithms that the images are different or unrelated. The schemes allow hierarchical comparisons where successive stages involve doubling the number of computations, but are performed only with much smaller probabilities. Unlike cryptographic hashes [8], our image hashes remain the same despite small changes to input images, and are thus useful even when images undergo compression, filtering, and other distortions.

Each of our hashing algorithms performs the following sequence of basic steps:

1. Subdivide an image into pseudorandom, possibly overlapping regions (typically rectangles) of various sizes.
2. Compute a summary (typically an average) of the pixel intensities in each region.
3. Randomly round resulting values to some randomly chosen *vertices*.
4. Shorten the resulting vectors by applying error correction.
5. Output the compressed vector as the image hash.

The schemes can iterate this sequence of steps on an image and combine all or parts of the resulting vectors. In addition, for comparison of two images, the schemes can subtract the two hash vectors and multiply the scalar differences by "weights" chosen to represent the relative significance of the differences at particular vector locations.

Our schemes are designed to withstand a wide variety of attacks intended to change hash values while preserving visual similarity. In particular, our hashes are invariant to arbitrary amounts of scaling, and withstand some amount of cropping. Additionally, the algorithms cope with software such as StirMark and unZign [9], which slightly alter images to render watermarks unreadable [4]. The schemes resist individual attacks as well as combinations.

The methods presented here can also be used to generate short random binary strings as image hashes, although more work is required to make such hashes truly robust and short. The paper [14] builds on the general approach described here, as well as on extensive empirical study, to generate highly resilient short hashes from data in the the wavelet domain.

2 Algorithms

We present several specific algorithms that use the above framework. Each scheme operates on an intensity plane of an image, and may be executed separately on the red, green, and blue planes of an RGB color image. Let k denote a secret key for generating a pseudorandom sequence of numbers, and I an image plane. Our first algorithm chooses many points, randomly spaced on the real line, within the range of image coordinates. We call these *vertices*. Rectangles can be replaced by arbitrary shapes.

2.1 Scheme A: Random rectangles

1. Use k to produce n random rectangles $r_1, r_2, ..., r_n$ in the space of the image.
2. Compute the intensity average μ_i of the pixels in each r_i.
3. Round each μ_i to one one of the closest vertices and produce d_i. The rounding may be randomly done so that expected value after the rounding is equal to the original quantity itself.
4. Apply an error-correction algorithm on the d_i.
5. Output the vector $D = d_1, d_2, ..., d_n$.

Random rounding of a value x in the interval $[0, 1]$ maps x to 0 with probability x, and to 1 with probability $1 - x$. This generalizes straightforwardly to other intervals.

A simple instance of this algorithm uses only two vertices corresponding to bits; specifically, each intensity average is mapped to either 0 or 1 based on a threshold. For error correction, groups of n bits are compressed into a single bit using majority-logic decoding; that is, a 0 is produced if there are more 0s than 1s in the group of n bits, and a 1 is produced otherwise. Typically, n is in the range 3...20, with higher n offering more robustness but fewer bits in the hash value.

2.2 Scheme B: Binary space partitioning

Let D denote an initially empty vector that will contain the image hash. Let s be the subdivision level (number of times image has been split, initially 0), and $s_{\max}$ the maximum subdivision level.

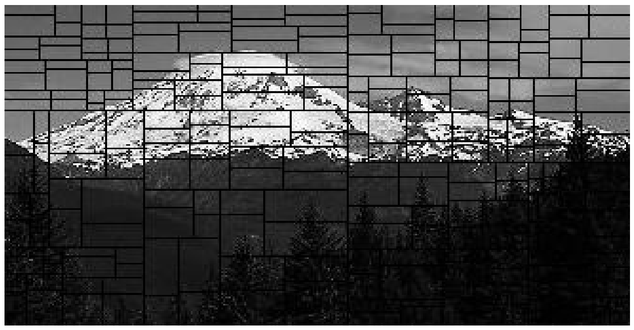

Figure 1: Test image before resizing and StirMark distortion, subdivided recursively into rectangles used to compute intensity averages in Scheme B.

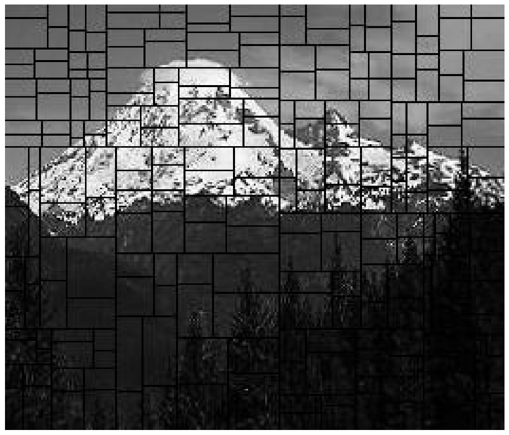

Figure 2: Test image after resizing and StirMark distortion, "visually similar" to the image in the preceding figure.

1. If I's dimensions are lesser than a predetermined minimum, or if $s \geq s_{\max}$,compute and randomly round the intensity average of pixels in I, insert this value into D, and return.

2. Pseudorandomly split I into two rectangular regions I_1 and I_2.

3. Compute, randomly round, and error-correct intensity averages of pixels in $I_1$and I_2. Insert these two values into D.

4. Recur this procedure on the subimages $I_1$and I_2, with s incremented by 1.

Pseudorandom splitting can be done in a number of ways. For example, we may alternate vertical and horizontal splitting between adjacent recursion levels. In addition, we can generate splits to ensure that subimages do not become too thin or too wide in any recursion step.

2.3 Scheme C: Hierarchical comparisons

We can generalize scheme B by splitting the image into more than two subimages in each recursion step, executing the entire procedure multiple times with different pseudorandom sequences determining the split points, and combining parts of the resulting hash vectors. As an example of such an algorithm, consider two executions of scheme B, each subdividing the image twice ($s_{\max} = 2$), first vertically and then horizontally. This generates two hash vectors, $[h_1, h_2]$ and $[g_1, g_2]$, where $h_k$and g_k are the vector components generated in the k-th recursion step, for $k = 1, 2$. To produce the final hash vector, we combine $h_1$and g_2 into one vector, $[h_1, g_2]$. For comparison of two images, we subtract their two hash vectors and multiply the scalar differences of the g_2 components by weights between 0 and 1 (to decrease the significance of these components, which correspond to image regions at finer resolution and are thus more error prone than the $h_1$components). We can then compute the norm (or some other function) of the difference vector as a measure of similarity between the two images.

A generalized version of the above method uses a sequence of hierarchical image *tesselations* to compute hash vectors. We define a *tessellation* of an image as follows: First, subdivide the horizontal axis into intervals having lengths chosen from a distribution that has expectation μ_H and finite variance; then independently subdivide each horizontal image region vertically so that the intervals have expected height μ_V and finite variance. Given a tessellation T_i, the next tessellaton T_{i+1} has parameters μ_H and μ_V half (or a constant factor smaller than) those of the previous tessellation. We compute the averages of pixel values in each of the cells in T_i, and construct a vector V_i of the averages. Our randomized rounding strategy then rounds the vectors V_i.

A hierarchical strategy can compare two images efficiently using the following method. Given an image, we compute T_1 and a vector V_1, and compare the image with another image using the vectors from the first tessellations only. If we observe sufficient closeness, we compare further using the second set of tessellations (which are at a finer resolution level). Given that an image is not likely to be close to another arbitrary image, it is clear that for uncorrelated images, successive comparison stages will be reached with lesser and lesser probability.

3 Applications

Hashes generated by the above algorithms have a variety of potential uses. For example, we can maintain a database of hashes for copyrighted images placed on the Web, and deploy a Web spider that searches for infringing copies of the images. The spider hashes each image it finds and attempts to locate the hash in its database. This technique can be used instead of image watermarks [2, 5, 10]; advantages include no image degradation due to embedded watermarks, as well as the potential for robustness impractical to achieve with watermarks. Additionally, image hashes can be used to index images for various types of searching [1, 11]; for example, an image's hash can serve as a key into a database of textual descriptions or other information [12].

The methods presented here can be extended to other media, such as audio and video [6, 7]. For streaming audio, we can use hashing to enhance watermarking: A window sliding over an audio signal can yield a hash useful for determining the start of watermark insertion and detection. Such hashes can also serve to identify the audio signal itself, particularly when combined with a perceptual model [3, 13] that determines which parts of the signal are aurally important. The sliding-window technique applies similarly to images, where a two-dimensional hashing window can help to locate watermarks and subimages; this can be useful for countering Web-based presentation attacks that break up or coalesce images onto Web pages [9]. We are currently investigating the uses of hashing in various domains and attack scenarios.

4 Conclusion

We presented several methods to compute image hashes, or compressed image representations useful for locating and identifying images. Such hashes are robust to small changes in input images, and thus do not fall under the cryptographic notion of hash values. However, images comprise relatively malleable data, and simple operations such as compression and rescaling are often done without intent to change basic visual appearance. Our methods also apply to other domains, such as audio and video, where both standard signal-processing operations and malicious attacks can change data slightly. Taking this into account, our hashes are useful in applications where traditional cryptographic hashes do not apply.

References

[1] S. Bhattacharjee and M. Kutter, "Compression tolerant image authentication," *Proc. IEEE-ICIP '98*, 1, Sep. 1998.

[2] I. J. Cox, J. Kilian, T. Leighton and T. Shamoon, "Secure spread spectrum watermarking for images, audio and video," *Proc. IEEE-ICIP '96*, Oct. 1996.

[3] I. J. Cox and M. L. Miller, "A review of watermarking and the importance of perceptual modeling," *Proc. Electronic Imaging*, Feb. 1997.

[4] I. J. Cox and J. P. Linnartz, "Some general methods for tampering with watermarks," *IEEE Journal on Selected Areas in Communications*, 16(4), May 1998.

[5] D. Gruhl and W. Bender, "Information hiding to foil the casual counterfeiter," *Second Workshop on Information Hiding*, Portland, Oregon (USA), April 1998.

[6] F. Hartung and B. Girod, "Fast public-key watermarking of compressed video," *Proc. IEEE-ICIP '97*, Oct. 1997.

[7] G. C. Langelaar, R. L. Lagendijk and J. Biemond, "Real-time labeling of MPEG-2 compressed video," *J. Visual Communication and Image Representation*, 9(4), Dec. 1998.

[8] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1998.

[9] F. A. P. Petitcolas, R. J. Anderson and M. G. Kuhn, "Attacks on copyright marking systems," *Second Workshop on Information Hiding*, Portland, Oregon (USA), April 1998.

[10] F. A. P. Petitcolas, R. J. Anderson and M. G. Kuhn, "Information hiding – a survey," *Proc. IEEE*, 87(7), July 1999.

[11] M. Schneider and S.-F. Chang, "A robust content based digital signature for image authentication," *Proc. IEEE-ICIP '96*, 3, Sep. 1996.

[12] M. D. Swanson, S. Hosur and A. H. Tewfik, "Coding for content-based retrieval," *Proc. ICASSP '96*, Atlanta, GA, May 1996.

[13] M. D. Swanson, B. Zhu, A. H. Tewfik and L. Boney, "Robust audio watermarking using perceptual masking," *Signal Processing*, 66(3), May 1998.

[14] R. Venkatesan, M. Jakubowski, S.-M. William Koon and P. Moulin, "Robust image hashing," December 1999.