Fast Inverse Halftoning

Zachi Karni, Daniel Freedman, Doron Shaked

HP Laboratories
HPL-2010-52

Keyword(s):
inverse halftoning

Abstract:

Printers use halftoning to render printed pages. This process is useful for many printing technologies which are binary in nature, as it allows the printer to deposit the ink as series of dots of constant darkness. Indeed, many of printing pipelines are based on this 1-bit framework; this unfortunately raises a critical problem when image processing operations that require the original 8-bit image must be performed. In this situation, what is required is the reconstruction of the 8-bit image from its halftoned version, a process referred to as "inverse halftoning". In this paper, we present a technique for fast inverse halftoning which given a dithered image together with the dithering mask that created it, approximates the original 8-bit image. The technique is elegant, and allows for generalizations to other inverse problems in which the exact details of the forward process are known. The algorithm is light computationally, and has been tested in practice. Results are shown, demonstrating the algorithm's promise.

External Posting Date: April 21, 2010 [Fulltext] Approved for External Publication
Internal Posting Date: April 21, 2010 [Fulltext]

To be published and presented at the 31st International Congress on Imaging Science (ICIS 2010), Beijing, China

Fast Inverse Halftoning

Zachi Karni, Daniel Freedman and Doron Shaked
HP Labs
Haifa, Israel

Abstract

Printers use halftoning to render printed pages. This process is useful for many printing technologies which are binary in nature, as it allows the printer to deposit the ink as series of dots of constant darkness. Indeed, many of printing pipelines are based on this 1-bit framework; this unfortunately raises a critical problem when image processing operations that require the original 8-bit image must be performed. In this situation, what is required is the reconstruction of the 8-bit image from its halftoned version, a process referred to as "inverse halftoning".

In this paper, we present a technique for fast inverse halftoning which given a dithered image together with the dithering mask that created it, approximates the original 8-bit image. The technique is elegant, and allows for generalizations to other inverse problems in which the exact details of the forward process are known. The algorithm is light computationally, and has been tested in practice. Results are shown, demonstrating the algorithm's promise.

Introduction

Printers use halftoning to render printed pages. In this process, a regular 8-bit image is converted into a 1-bit image in such a way that the human eye perceives the two images as close to the same. This process is useful for many printing technologies which are binary in nature, as it allows the printer to deposit the ink as a series of dots of constant darkness. Therefore, it is a common practice that the entire printing pipeline is based on this 1-bit framework. Unfortunately, a critical problem arises when image manipulations, which are usually 8-bit in nature, are required to be performed at the printer level.

To perform this compensation properly, it is necessary to reconstruct an 8-bit image from the given 1-bit image, a process referred to as "inverse halftoning". This is a highly challenging inverse problem; in this work, we provide an elegant but fast solution to this problem, and present some initial results showing the promise of this approach. We also note that our approach to this problem can be generalized to a larger class of inverse problems, in which one knows the exact details of the forward process. For halftoning, the advantages of this approach over existing techniques are twofold: speed and the ability to deal with dithered images (rather than error diffused images), which are most relevant for many printing applications. We are hopeful that these advantages may be extended to other such inverse problems, including compression and tomography.

Problem Statement

We are given a 1-bit image, I_1 which is the result of a dithering process, as follows. For each pixel, the value of the original 8-bit image I_8 is thresholded; the threshold value varies by pixel, and depends on the dither mask. Wherever the pixel value is bigger than the corresponding value in the dither mask, the resulting 1-bit value is set to 1. The resulting 1-bit value is set to 0 wherever the pixel value is lower or equal to the corresponding value in the dither mask. The dithering process is illustrated in Figure 2.

The problem of inverse halftoning is then to go in the reverse direction: given the 1-bit image as well as the threshold values (as the dithering mask), reconstruct an approximation $\hat{I}_8$ to the 8-bit image. Clearly, inverse halftoning is highly underdetermined, due to the extreme many-to-one nature of thresholding. The idea is to carefully use the redundancy of the image, i.e., that neighboring pixels tend to have similar gray values, to help solve the inverse problem.

Previous Work

In general, there are many proposed solutions to general inverse problems, which are quite powerful, such as graph cuts [1], belief propagation [2], and so on; however, these techniques are typically too slow for the intended application. On a more specific level, there have been a variety of efforts in the domain of inverse halftoning. Some of these [3] are aimed at halftones generated by error-diffusion rather than dithering, while others are computationally heavier [4] than is practical in most applications of interest.

Our Solution

Collecting Statistics: To compute the 8-bit reconstruction $\hat{I}_8(p)$, we begin by collecting statistics within a fixed window $W(p)$ around the pixel p of interest. We focus on statistics which are most related to the process of dithering or thresholding. Without a priori knowledge about the 8-bit image, the reconstructed 8-bit pixel value can be taken as a random variable with a uniform probability according to the threshold value. This means, for each 0 pixel in the 1-bit image, the reconstructed 8-bit value is uniformly distributed between 0 and the threshold value. On the other side, for each 1 pixel, the uniform distribution is between the threshold value and 255, as it is presented in Figure 1. We begin by computing a histogram of dither values $\theta(q)$ (from the dithering mask) within the window, as given by Equation (1). Now, due to the use of thresholding in forming the 1-bit image, the key statistic to examine is the *Conditional Expectation Function*, described in Equation (2). This function is simply the average or expectation of the dither values, conditioned on the fact that the dither values are less than a fixed value I; the function Q_1 records this expectation for each I. Note that it can be simply computed using the histogram, as in Equation (2).

$$h(x) = \left|\{q \in W(p) : \theta(q) = x\}\right| \tag{1}$$

$$Q_1(I) \equiv E[\Theta \mid \Theta \le I] = \frac{\sum_{x=0}^{I} x h(x)}{\sum_{x=0}^{I} h(x)} \tag{2}$$

Regularization: In order to solve the inverse problem, we need more information; this information comes in the form of the "regularization assumption" so common to inverse problems. Indeed, we use a rather extreme regularization, and assume that the image is constant within the window under consideration; let us refer to this constant value as $I_8(p)$. Now, suppose that we compute the empirical average of all dither values within the window, but only for those pixels whose 1-bit image value is 1; this quantity q_1^* is explicitly computed in Equation (3).

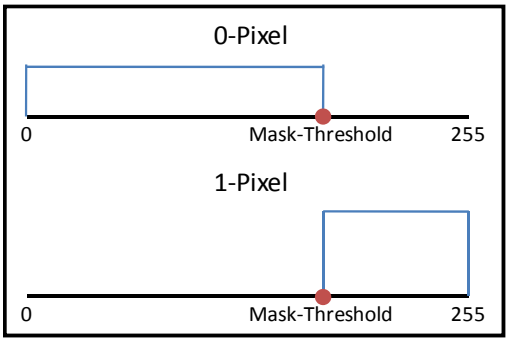

Figure 1: The reconstructed 8-bit value is taken to be a random variable with a uniform probability function according to the mask threshold level and the 1-bit value.

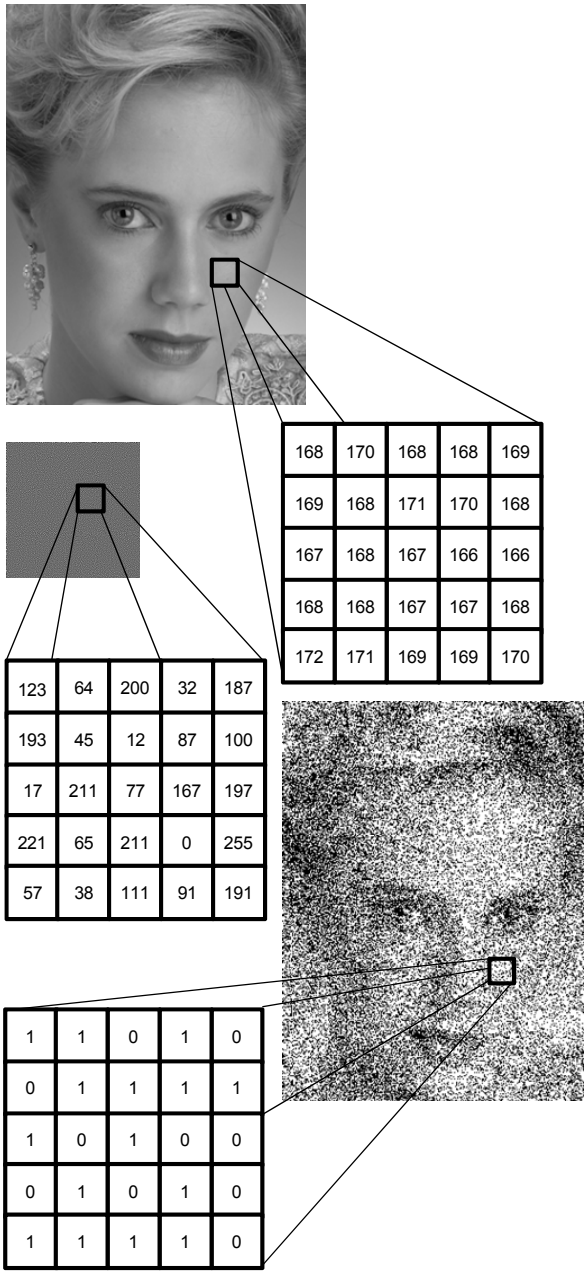

Figure 2. The dithering process. Top: Original 8-bit image. Middle: A dithering mask. Bottom: The 1-bit dithered image.

$$q_1^* = \frac{\sum_{q \in W(p)} I_1(q)\theta(q)}{\sum_{q \in W(p)} I_1(q)} \tag{3}$$

If the number of pixels is large enough, this quantity should be equal to the Conditional Expectation Func-

tion of Equation (2), evaluated at $I_8(p)$; as a result, we can invert to get an approximation of the 8-bit image, i.e. $\hat{I}_8^1(p) \equiv Q_1^{-1}(q_1^*)$, as in Equation (4). Figure 3 presents a histogram of a typical dithering cell and its corresponding Conditional Expectation Function.

$$q_1^* \approx Q_1(I_8(p)) \Rightarrow \hat{I}_8^1(p) \equiv Q_1^{-1}(q_1^*) \qquad (4)$$

Finally, note that we have only used the pixels whose 1-bit image values are 1. We may do the entire process again for those pixels with a 1-bit image value of 0. In this case, we let the Conditional Expectation Function be $Q_0(I) \equiv E[\Theta \mid \Theta > I]$, and so on. The equation analogous to (4) gives $\hat{I}_8^0(p)$. We combine the two estimates for the eight-bit image $\hat{I}_8^0(p)$ and $\hat{I}_8^1(p)$ as in Equation (5) through a simple weighted average, where f_0 is the fraction of pixels within the window whose 1-bit image values are 0.

$$\hat{I}_8(p) \equiv f_0 \hat{I}_8^0(p) + (1 - f_0)\hat{I}_8^1(p) \qquad (5)$$

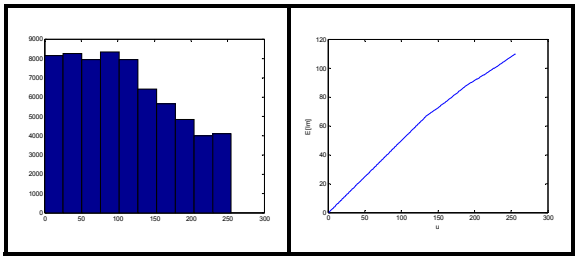

Figure 3. Left: A histogram of a typical dithering cell. Right: A conditional expectation function.

Window Selection: The key issue that arises from the previous analysis surrounds the regularization assumption. In particular, there is a tradeoff between the accuracy of the procedure and the accuracy of the assumption: for small window sizes, the assumption of constant 8-bit value is largely correct, but we do not collect enough statistics for the procedure to be accurate; and the reverse is true for large windows. The effects of varying window sizes may be seen in Figure 4. We therefore use the following adaptive window size algorithm, based on the fact that we *know* the details of the halftoning procedure. Suppose we have several possible window sizes $s_i \times s_i$, $i = 1 \ldots n$; for example, 3×3, 5×5, etc. For each size, we compute the 8-bit reconstruction at each point, now denoted as $\hat{I}_8^i(p)$; and for each such reconstruction, we recompute the halftoned image, $\hat{I}_1^i(p)$. We then gauge the correctness of a particular window size by comparing the closeness of $\hat{I}_1^i$ to the true halftoned image I_1 on a pixel-by-pixel basis. There are many ways to do this; we choose to compute, for each p, the number of points $g_i(p)$ in a fixed size window around p at which $\hat{I}_1^i(p) = I_1(p)$. We then take $\hat{I}_8(p)$ to be a weighted sum of the various reconstructions $\hat{I}_8^i(p)$, where the weights are proportional to the $g_i(p)$.

Generalization to Other Inverse Problems: The window selection procedure works because, as noted, we know the details of the halftoning procedure. This fact differentiates this inverse problem from many others, such as noise removal, where we only know a statistical characterization of the forward problem. Thus, halftoning is closer, in this sense, to certain problems such as decompression from a known compression scheme, deterministic deblurring, or tomography; in all of these cases, the forward process is understood exactly.

Figure 4. Effect of the varying window size. From left to right, top to bottom: windows size of 3×3, 9×9, 15×15 and 25×25. Smaller windows preserve artifacts of the dithering process, while larger windows lead to oversmoothing.

Experimental Results

The algorithm was implemented in MATLAB, which is acceptably fast given the number of matrix operations. To test the algorithm, we take an 8-bit image, halftone it, and then run our reconstruction algorithm. The results are shown in Figure 5 and Figure 6. In the latter, we focus on the woman's eye, to highlight the fact that our algorithm retains nearly all of the important details, such as the fineness of the eyelashes, and the skin texture. This is due to the use of the adaptive window sizes. For a quantitative comparison, we compute two quantities: the PSNR of the 8-bit reconstruction is 30.4 dB, while the fraction of mistakes in the rehalftoned image (i.e. when we halftone $\hat{I}_8$ and compare it with the halftoned version of I_8) is 0.3%.

In contradiction to our basic assumption that an image is constant within a small neighborhood, our method fails for pure constant images. The main reason for this is the existence of higher level moiré artifacts introduced during the forward dithering process. These artifacts are emphasized in the inverse approach. A possible solution for such a case is to use a large neighborhood which will smooth these artifacts out.

Figure 5: Inverse halftoning results. Top to bottom: The original 8-bit image, the inverse halftone result is very close to the original.

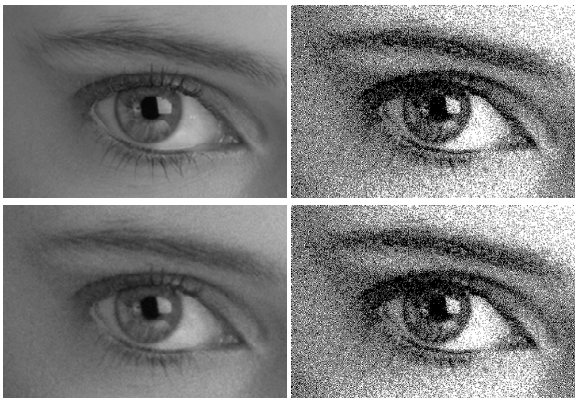

Figure 6: Close-up on the eye shows that the important features such as eyelashes, eyebrow and skin texture are retained. Top to bottom: original image, reconstructed image. Left to right: 8-bit and 1-bit images,

Conclusions

We presented a fast and elegant approach for the inverse problem of image halftoning. The approach can be easily integrated into a 1-bit pipeline to allow image manipulation at the printer level. We showed that the quality of the reconstructed image is visually close to the original both on the 8-bit and 1-bit images.

References

1. V. Kolmogorov, R. Zabih. What energy function can be minimized via Graph Cuts? IEEE Transactions on Pattern Analysis and Machine Intelligence, pg. 147-159. (2004).
2. J. Yedidia, W. Freeman, Y. Weiss. Generalized Belief Propagation. Advances in Neural Information Processing Systems, pg. 689-695. (2001).
3. T. Kite, N. Damera-Venkata, B. Evans, A. Bovik. A fast, high-quality inverse halftoning algorithm for errordiffused halftones. IEEE Transactions on Image Processing, Vol 9(9), pg. 1583-1592. (2000).
4. Y. Kim, G. Arce, N. Grabowski. Inverse halftoning using binary permutation filters. IEEE Transactions on Image Processing, Vol 4(9), pg. 1296-1311. (1995).