INVERSE HALFTONING BY DECISION TREE LEARNING

Hae Yong Kim and Ricardo L. de Queiroz***
*Escola Politécnica, Universidade de São Paulo, Brazil. E-mail: hae@lps.usp.br.
**Universidade de Brasília, Brazil. E-mail: queiroz@ieee.org.

ABSTRACT

Inverse halftoning is the process to retrieve a (gray) continuous-tone image from a halftone. Recently, machine-learning-based inverse halftoning techniques have been proposed. Decision-tree learning has been applied with success to various machine-learning applications for quite some time. In this paper, we propose to use decision-tree learning to solve the inverse halftoning problem. This allows us to reuse a number of algorithms already developed. Especially, the maximization of entropy gain is a powerful idea that makes the learning algorithm to automatically select the ideal window as the decision-tree is constructed. The new technique has generated gray images with PSNR numbers, which are several dB above those previously reported in the literature. Moreover, it possesses very fast implementation, lending itself useful for real time applications.

1. INTRODUCTION

Most ink-jet and laser printers can print only tiny black dots in paper. Thus, any continuous-tone image has to be first converted into a binary image before the printing actually takes place. Halftoning simulates gray shades by scattering appropriately black and white pixels. Popular halftoning techniques are error diffusion, ordered dithering and blue noise masks [1],[2].

Halftone images can be either orthographic (perfect digital binary data before printing) or scanned. The later is typically a gray picture of a printed halftone.

Inverse halftoning (IH) or descreening is the process of retrieving a (gray) contone image from a halftone. Since halftoning is a many-to-one process, there is no unique contone image for a given halftone image. Hence, extra image properties have to be used in IH. The simplest IH method is to use a simple low-pass filter. Even though it would produce gray levels, it will also tend to blur edges and to destroy fine details. Many different methods were reported to improve descreening. See [3] for further discussion.

Mese and Vaidyanathan have recently proposed a machine-learning-based approach for inverse halftoning orthographic halftone images. They first proposed an algorithm that uses look-up-table (LUT) [4] and another that uses a tree structure [5]. In both methods, there is a training phase where sample images are used to construct the data structure.

Machine-learning has been used in various image-processing applications. In particular, one can use it to increase the resolution of halftone images using LUT [6] and a decision-tree (DT) [7]. These techniques can be straightforwardly adapted for the IH problem.

In [5], an “ad hoc” tree-structured machine-learning algorithm was employed. We, however, in this paper, propose to use a theoretic and algorithmic framework based on DT-learning towards IH. More specifically, we use a version of DT-learning called ID3. This approach presents advantages over [5]. The reason for this is twofold. First, in [5], the user has to carefully select the template (i.e., the window). They even present an algorithm to select a “good” window [4]. DT-learning exempts the user from explicitly choosing a window, because it will automatically use only those attributes (peepholes or pixels) that are most necessary to decide the output contone color. While they have used windows with at most 19 peepholes (which may be too small for many IH problems), using DT-learning, we opt for a much larger initial window (for example 8×8=64 pixels). The learning algorithm itself will automatically use only the appropriate peepholes in the appropriate order. The choice of peepholes is based on the maximization of the expected reduction in entropy and this policy has shown to produce short trees and good generalizations [8],[9]. Furthermore, they use an unorthodox (however interesting) data structure that combines a LUT and a binary tree. Although there is nothing wrong with this approach, the original decision-tree is clearly more elegant, and presents the equivalent computational performance.

We believe [5] to possess state-of-the-art performance and our algorithm to be a natural next-step improvement, since we can obtain images with PSNR that seem to be around 4 dB superior to those reported there. Programs and images used in this paper can be downloaded from:

http://www.lps.usp.br/~hae/software/invhalf

2. THE INVERSE HALFTONING PROBLEM

Binary and grayscale images are respectively defined as functions $Q^{\mathbf{x}}: \mathbb{Z}^2 \to \{0,1\}$ and $Q^{\mathbf{y}}: \mathbb{Z}^2 \to [0...255]$. The support of an image is a finite subset of $\mathbb{Z}^2$ where the image is actually defined.

A binary to grayscale windowed operator Ψ is a function that maps a binary image $Q^{\mathbf{x}}$ into a grayscale image $Q^{\mathbf{y}}$, defined via a set of w points called window $W=\{W_1,\ldots,W_w\}$, $W_i \in \mathbb{Z}^2$, and a characteristic function $\psi:\{0,1\}^w \to [0...255]$ as follows:

$$Q^{\mathbf{y}}(p)=\Psi(Q^{\mathbf{x}})(p)=\psi(Q^{\mathbf{x}}(W_1+p),\ldots,Q^{\mathbf{x}}(W_w+p)),$$

where $p\in\mathbb{Z}^2$. Each element W_i of window is called a peephole or a feature.

Let images $A^{\mathbf{x}}$, $A^{\mathbf{y}}$, $Q^{\mathbf{x}}$ and $Q^{\mathbf{y}}$ be, respectively, the training input-image, the training output-image, the image to be inverse halftoned and the (supposedly unknown) ideal output image. We can suppose that there is only one pair of training images ($A^{\mathbf{x}}$ and $A^{\mathbf{y}}$). If there are more pairs, they can be "glued" together to form a single pair.

Let us denote the content in $A^{\mathbf{x}}$ of the window W shifted to $p\in\mathbb{Z}^2$ as $a_p^{\mathbf{x}}$ and call it the training instance or the sample input-pattern at pixel p:

$$a_p^{\mathbf{x}}=\left(A^{\mathbf{x}}(W_1+p),A^{\mathbf{x}}(W_2+p),\ldots,A^{\mathbf{x}}(W_w+p)\right)\in\{0,1\}^w.$$

Each pattern $a_p^{\mathbf{x}}$ is associated with an output-color $A^{\mathbf{y}}(p)\in[0...255]$. Let us denote the data obtained when all pixels of $A^{\mathbf{x}}$ and $A^{\mathbf{y}}$ are scanned by

$$a=\left\{(a_{p_1}^{\mathbf{x}},A^{\mathbf{y}}(p_1)),\ldots,(a_{p_m}^{\mathbf{x}},A^{\mathbf{y}}(p_m))\right\}$$

and call it the sample set or the training set (m is the amount of pixels of images $A^{\mathbf{x}}$ and $A^{\mathbf{y}}$). Let us similarly construct the test set

$$q=\left\{(q_{p_1}^{\mathbf{x}},Q^{\mathbf{y}}(p_1)),\ldots,(q_{p_n}^{\mathbf{x}},Q^{\mathbf{y}}(p_n))\right\}$$

from $Q^{\mathbf{x}}$ and $Q^{\mathbf{y}}$ (n is the quantity of pixels of $Q^{\mathbf{x}}$ and $Q^{\mathbf{y}}$). Each $q_{p_i}^{\mathbf{x}}$ is called a query-pattern or an instance to-be-processed and the output $Q^{\mathbf{y}}(p_i)\in[0...255]$ is called the ideal output-color.

In IH-problem, the learning algorithm **A** constructs an operator $\hat{\Psi}$ based on $A^{\mathbf{x}}$ and $A^{\mathbf{y}}$ such that, when $\hat{\Psi}$ is applied to $Q^{\mathbf{x}}$, the resulting processed-image $\hat{Q}^{\mathbf{y}}=\hat{\Psi}(Q^{\mathbf{x}})$ is expected to be similar to the ideal output-image $Q^{\mathbf{y}}$.

To describe more precisely this process, let us define a loss (error) function l that will be used to measure the difference between the ideal and processed outputs. Restating the IH problem, the learner **A** should construct a characteristic function or hypothesis $\hat{\psi}$ based on the sample set a such that, when $\hat{\psi}$ is applied to a query-pattern $q_{p_i}^{\mathbf{x}}$ yielding the output-color $\hat{Q}^{\mathbf{y}}(p_i)=\hat{\psi}(q_{p_i}^{\mathbf{x}})$, the loss $l\left(Q^{\mathbf{y}}(p_i),\hat{Q}^{\mathbf{y}}(p_i)\right)$ have to be low with high probability.

3. DECISION-TREE LEARNING

There are many learning techniques that can be used to achieve the above goal. However, to be really useful, the technique has to present the following properties:

1. It must generalize beyond the training set. That is, it must output values with little loss not only for patterns from the training set but also for other (unseen) patterns as well.
2. It must have a fast implementation.
3. The memory space needed must be moderate.
4. The training phase must not take excessively long, although some delay can be tolerable.

Fortunately, DT-learning fulfills all these requirements. There are many versions of DT-learning. In this paper, we will use the ID3 algorithm [8],[9].

In the DT generating process, the input-pattern space $\{0,1\}^w$ is split into two halves, and all sample input-patterns with black color in the splitting attribute W_s will belong to one half-space and those with white color to another. The dimension of the half-spaces so obtained is one less than that of the original space, that is, $\{0,1\}^{w-1}$. For each one of the two half-spaces obtained, the splitting process continues recursively, generating smaller and smaller spaces. In each splitting, an internal node is created and the splitting attribute s is stored in it. This process stops when each space contains either only samples with the same output-color or only samples with the same input pattern but with two or more different output colors. In the first case, a terminal node is created and the output color is stored in it. The second case is called a conflict. In this case, a terminal node is created, but the average of output-values is evaluated and stored (if the square loss or PSNR is to be minimized). The averaging must be replaced by median if the absolute loss is to be minimized.

If there is no conflict (or noise), the above algorithm should perfectly classify the training set. However, machine-learning researchers have observed that this strategy can lead to "overfitting". We say that a hypothesis overfits the training examples if some other hypothesis that fits the training examples more poorly nevertheless performs better over the entire distribution of instances. In the literature, there are a few sophisticated strategies to avoid overfitting. However, using them, the training phase may take too long. Thus, we use the following simplifying strategy: the average (or median) is computed whenever there are k or fewer samples in a pattern subspace. When $k = 1$, we have the original algorithm. We will denote this strategy as k-ID3, for example, 1-ID3 or 10-ID3.

The central choice in the ID3 algorithm is to select which attribute will be used to split the input pattern space at each node of the tree. Different choices for splitting the axis will generate different DTs. Given a choice between two DTs, it is widely accepted that the simpler (shorter) one is to be preferred. This choice is known as "Occam's

Razor" and many studies point to its superiority, including our own experiences with the IH problem. One way to conform to "Occam's Razor" is to generate all possible DTs and to select the simplest of them. Clearly, this approach is impractical because would take too long to compute.

The ID3 algorithm uses the following criterion that closely follows "Occam's Razor". It consists on placing high information gain attributes close to the root. Let us define the entropy of a sample set a where each sample can take one of c different output values:

$$\text{Entropy}(a) \equiv \sum_{i=1}^{c} - p_i \log_2 p_i$$

where p_i is the proportion of examples in a belonging to the class i. For the IH problem, we discretized (quantized) 256 possible output values into $c = 16$ categories: [0...15], [16...31], etc. This discretization is used only to compute the entropy. The precise output value is still stored in the DT leaves. The information gain, defined below, is the expected reduction in entropy caused by partitioning the examples according to the attribute s:

$$\text{Gain}(a,s) \equiv \text{Entropy}(a) - \sum_{v\in\{0,1\}} \frac{|a_v|}{|a|} \text{Entropy}(a_v)$$

where a_v is the subset of a for which attribute s has value v. In each internal node, the learning algorithm chooses the attribute that maximizes the information gain. The DT-learning algorithm that does not use the maximization of information gain will be called sequential, because it chooses peepholes sequentially in a pre-determined order.

Once the decision-tree has been constructed, its application is quite straightforward: given a query pattern $q_p^{\mathbf{x}}$, DT is traversed from top to bottom, until a terminal node is reached. The information contained in this leaf is then chosen as the output-value $\hat{Q}^{\mathbf{y}}(p_i)$.

4. EXPERIMENTAL RESULTS

The proposed technique has been implemented and tested. We used three pairs of 1050×1050 in-out images, glued together, for the training. We applied the resulting IH system to three completely independent 1050×1050 images. We used an 8×8 window for all cases. The tests were repeated for four different types of halftones:

1. Error diffusion (Floyd-Steinberg's algorithm).
2. Dispersed-dot ordered dithering (Bayer).
3. Clustered-dot ordered dithering.
4. Halftone provided by HP's LaserJet driver for MS-Windows, option "coarse dot."

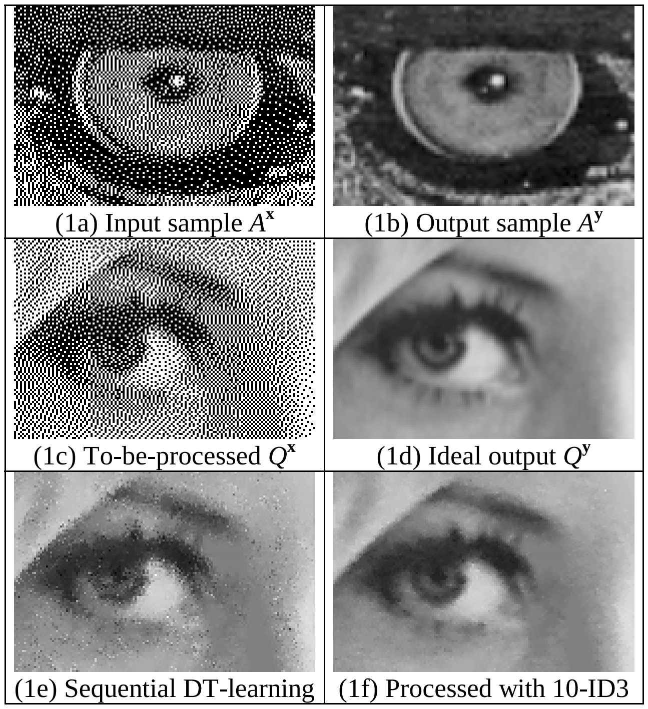
(1a) Input sample $A^{\mathbf{x}}$
(1b) Output sample $A^{\mathbf{y}}$
(1c) To-be-processed $Q^{\mathbf{x}}$
(1d) Ideal output $Q^{\mathbf{y}}$
(1e) Sequential DT-learning
(1f) Processed with 10-ID3

Fig. 1: Decision-tree inverse halftoning error diffused images. (a, b) Sample input $A^{\mathbf{x}}$ and sample output $A^{\mathbf{y}}$ (Mandrill). (c, d) Image to-be-inverse-halftoned $Q^{\mathbf{x}}$ and supposedly unknown ideal output-image $Q^{\mathbf{y}}$ (Lena). (e) Inverse-halftoned image using sequential DT-learning (PSNR 26.75 dB). (f) Inverse-halftoned image using 10-ID3 algorithm (PSNR 34.75 dB).

Figure 1 depicts the application of the proposed technique to halftones obtained by error diffusion. Part of one of the training images is shown in Figs. 1(a) and 1(b). Part of one of the images to be inverse halftoned is depicted in figure 1(c) and the corresponding (supposedly unknown) ideal output in figure 1(d). Figures 1(e) and 1(f) are the inverse halftoned images obtained using sequential and 10-ID3 algorithms, respectively. Their PSNRs are 26.75 and 34.75 dB for Lena image. This demonstrates that the maximization of information gain plays an important role in improving the quality of our IH system. Applying IH to 3 different test images (one of which was Lena), we can see that PSNR varies considerably as shown in Table 1. Image Lena seems to be "good" for inverse halftoning, probably because of its large smooth areas.

Figure 2 depicts parts of halftone images obtained using different algorithms and their respective inverse halftoned images. Their PSNR can be found in Table 1.

In [5], the PSNR reported was 27.08 dB for the inverse halftoned error-diffused set of images. Our technique yields 31.85 dB for a set of images, which, although different, contains images with similar characteristics. Testing only Lena image, [5] reports 30.95 dB while our technique yields 34.75 dB. For dispersed-dot and clustered-dot ordered dithering our method yields 30.22 dB and 28.95 dB, respectively. To trace some comparison, the same numbers in [5] are 25.82 dB and 24.26 dB. This seems to point to a gain in excess of 4 dB compared to previously reported results. In order to certify that the results are not biased, 10-ID3 algorithm was tested again using 6 completely different 1280×960 images, three for the training and three for the test. The results are depicted in table 1, rows labeled "3 images B."

The descreening phase takes only 5 seconds in Pentium-1GHz, for a 1050×1050 test image. The training phase takes approximately 20 minutes for ID3 and 1 minute for sequential DT-learning (using images with 1050×3150 pixels). The DT structure occupies approximately 5 Mbytes.

		10-ID3	1-ID3	Sequential
Error Diffusion	3 images A	31.85	31.02	25.46
	Lena only	34.75	33.20	26.75
	3 images B	32.98	-	-
Dispersed OD	3 images A	30.22	29.73	28.18
	Lena only	33.69	33.17	32.97
	3 images B	31.46	-	-
Clustered OD	3 images A	28.95	28.56	27.84
	Lena only	32.59	32.01	31.83
	3 images B	30.80	-	-
HP laser coarse dot	3 images	27.74	27.32	27.37
	Lena only	31.72	31.16	31.07
	3 images B	-	-	-

Tab. 1: PSNR obtained using different DT-learning and halftone algorithms (in dB). "3 images A" refers to PSNR obtained testing IH-system in 3 different test images, one of which was Lena. "Lena only" refers to PSNR of Lena image. "3 images B" refers to PSNR obtained using a completely different set of images.

5. CONCLUSIONS

In this paper, we presented a new ortographic inverse halftoning algorithm based on the decision-tree learning. The proposed technique has shown to yield high quality descreened images.

6. ACKNOWLEDGEMENT

One of authors, H. Y. Kim, would like to express his gratitude to FAPESP and CNPq for the partial financial support of this work under grants 2001/02400-9 and 300689/98-5, respectively.

7. REFERENCES

[1] P. Roetling and R. Loce, "Digital Halftoning", Chapter 10 in *Digital Image Processing Methods*, E. Dougherty Ed., Marcel Dekker, New York, NY, 1994.

[2] R. Ulichney, *Digital Halftoning*, The MIT Press, 1987.

[3] J. Luo, R. Queiroz and Z. Fan, "A Robust Technique for Image Descreening Based on the Wavelet Transform," *IEEE T. Image Processing*, vol. 46, no. 4, pp. 1179-1184, April 1998.

[4] M. Mese and P. P. Vaidyanathan, "Look-Up Table (LUT) Method for Inverse Halftoning," *IEEE T. Image Processing*, vol. 10, no. 10, pp. 1566-1578, October 2001.

[5] M. Mese and P. P. Vaidynathan, "Tree-Structured Method for LUT Inverse Halftoning and for Image Halftoning," *IEEE T. Image Processing*, vol. 11, no. 6, pp. 644-655, June 2002.

[6] H. Y. Kim, "Binary Operator Design by *k*-Nearest Neighbor Learning with Application to Image Resolution Increasing," *Int. J. Imaging Systems and Technology*, vol. 11, no. 5, pp. 331-339, 2000.

[7] H. Y. Kim, "Fast and Accurate Binary Halftone Image Resolution Increasing by Decision-Tree Learning," in *Proc. IEEE Int. Conf. on Image Processing* (Thessaloniki, Greece), vol. 2, pp. 1093-1096, 2001.

[8] T. M. Mitchell, *Machine Learning*, WCB/McGraw-Hill, 1997.

[9] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, 1, pp. 81-106, 1986.

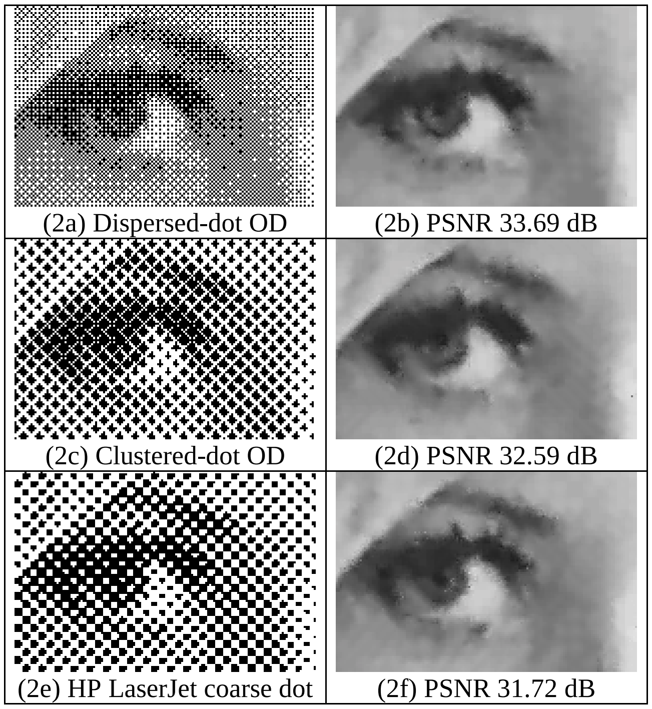

Fig. 2: Decision-tree inverse halftoning on different types of halftone images using 10-ID3 algorithm. (a, b) Bayer's dispersed-dot ordered dithered image and respective inverse-halftoned image. (c, d) Clustered-dot ordered dithered image. (e, f) Halftone image generated by HP LaserJet driver for MS-Windows, using option "coarse dot".