On Inverse Halftoning: Computational Complexity and Interval Computations

S. D. Cabrera and K. Iyer
Department of Electrical and Computer Engineering
University of Texas at El Paso, El Paso, Texas 79968
e-mail: cabrera@ece.utep.edu, kish_199@yahoo.com

G. Xiang and V. Kreinovich
Department of Computer Science
University of Texas at El Paso, El Paso, Texas 79968
e-mail: gxiang@utep.edu, vladik@cs.utep.edu

March 17, 2005

1. Need for Halftoning

- Inside the computer, a gray-scale image is represented by assigning, to every pixel (n_1, n_2), the intensity $f(n_1, n_2)$ of the color at this pixel.
- Usually, 8 bits are used to store the intensity, so we have $2^8 = 256$ possible intensity levels for each pixel.
- For color images, we must represent the intensity of each color component.
- A laser printer either prints a black (or a colored) dot, or it does not print anything at all.
- Therefore, when we print an image, we must first transform it into the "halftone" form $b(n_1, n_2) \in \{0, 1\}$.
- Crudely speaking, the level of intensity at a pixel is represented by the relative frequency of black spots around it.

2. Halftoning Techniques: In Brief

- There exist many halftoning algorithms.
- Most widely used – *error diffusion*:
  - we start with the original image $u(n_1, n_2) := f(n_1, n_2)$
  - we sequentially update the processed image $u(n_1, n_2)$ and quantize the processed value $u(n_1, n_2)$ into $b(n_1, n_2) = Q(u(n_1, n_2))$, where:

    $Q(u) = 0$ for $u < 0.5$ and $Q(u) = 1$ for $u \geq 0.5$.
  - Once the pixel is quantized, the quantization error $e(n_1, n_2) \stackrel{\text{def}}{=} b(n_1, n_2) - u(n_1, n_2)$ is spread out ("diffused") to the neighboring pixels:

$$u(n_1, n_2) = f(n_1, n_2) - \sum_{m_1, m_2} h(m_1, m_2) \cdot e(n_1 - m_1, n_2 - m_2).$$

3. Need for Reverse Halftoning

- Visually, the printed halftone image $b(n_1, n_2)$ looks identical to the original gray-scale image $f(n_1, n_2)$.
- So, from the halftone values $b(n_1, n_2)$, it is possible to reconstruct the original image.
- Why we need it: we know how to rotate or zoom the original image but not the halftone image.
- So, to go from a printed image to a printed zoomed and/or rotated image, we can:
  - use $b(n_1, n_2)$ to reconstruct $f(n_1, n_2)$;
  - then, we apply the zoom and/or rotation to $f(n_1, n_2)$, resulting in $f^*(n_1, n_2)$;
  - finally, halftone $f^*(n_1, n_2)$, and print the resulting halftone image $b^*(n_1, n_2)$.
- For that, we must reverse the halftoning procedure.

4. Halftoning is an Ill-Posed Problem: A Reminder

- Our objective is to reverse the halftoning operation.
- By definition, halftoning transforms:
  - the original gray-scale image in which we stored at least 8 bits per pixel,
  - into a black-and-white image in which we store only one bit per pixel.
- Thus, halftoning loses information.
- Therefore, halftoning is a lossy compression.
- Hence, there may be several different images that lead to the same halftoned image.

5. Inverse Halftoning: POCS

- *Main idea:* each value $b(n_1, n_2)$ of a halftone image represents a (convex) constraint on the original image $f(n_1, n_2)$.
- *In geometric terms:*
  - we have a point $b(n_1, n_2)$ in the function space,
  - we want to find the closest element to this point in the convex set S.
- It is known that to get this closest element, we can:
  - first minimally modify the original halftone image so that it satisfies the first constraint,
  - then minimally modify the modification so that it satisfies the second constraint, etc.
- *Result:* we get a good quality inverse halftoning.

6. Wavelet Inverse Halftoning

- *Idea:* use wavelet transform.
- *Motivation:*
  - halftoning is an example of lossy compression;
  - the experience of JPEG2000 has shown that wavelets best captures the visual quality of images uncompressed after a lossy compression.
- *Results:* wavelet-based inverse halftoning techniques lead to visually the best reconstruction among all known inverse halftoning methods.
- *Comment:* this empirical result is in good accordance with the JPEG2000 experience,
- *Problem:* wavelet methods require a lot of computation time.

7. Fast Inverse Halftoning

- *Idea:* the value $f(n_1, n_2)$ can be reconstructed from the density of black pixels around (n_1, n_2).
- *In engineering terms:* $f(n_1, n_2)$ can be obtained from $b(n_1, n_2)$ by low-pass filtering.
- *Problem:* a low-pass filter blurs the edges.
- *Solution:*
  - detect the edges, and
  - apply different filters (with different spatial radius) at different parts of the image.
- This idea has been successfully implemented in inverse halftoning, by Kite et al.
- *Result:* the new method is much faster than the wavelet-based, while the visual quality is almost as good as for the wavelet-based reconstruction.

8. What We Are Planning to Do

- *Remaining problem:* the existing inverse haltoning methods are still not optimal,
- especially low-computations methods implementable within printing devices.
- In this talk:
  - we show that the problem of inverse half-toning is a particular case of a class of difficult-to-solve problems:
  - inverse problems for reconstructing piece-wise smooth images.
  - We show that this general problem is NP-hard.
  - We also propose a new idea for solving problems of this type, including the inverse halftoning problem.

9. A General Problem

- Inverse halftoning problem is ill-posed $\approx$ has many different solutions.
- Many inverse problems in science and engineering are ill-posed.
- *Regularization:* we select a solution with a certain property, e.g., a smooth one, $J \stackrel{\text{def}}{=} \int (x'(t))^2\,dt \to \min$.
- *Discrete case:* $J_{\text{discr}} \stackrel{\text{def}}{=} \sum_i (x(t_{i+1}) - x(t_i))^2$.
- *2-D case:* $J \stackrel{\text{def}}{=}$

  $$\sum_{n_1,n_2} [(f(n_1+1,n_2) - f(n_1,n_2))^2 + (f(n_1,n_2+1) - f(n_1,n_2))^2],$$

  or, equivalently, $J = \sum_{p,p' \text{ are neighbors}} (f(p) - f(p'))^2$.
- Smoothness leads to efficient POCS-type algorithms.
- *Problem:* for image reconstruction, we only have piecewise smoothness.

Title Page
◀◀ ▶▶
◀ ▶
Page 10 of 22
Go Back
Full Screen

10. General Problem: Precise Formulation

- *Idea:* we only take into account the pairs of neighboring pixels that belong to the same zone:

$$J(Z) = \sum_{p,p' \text{ are neighbors in the same zone}} (f(p) - f(p'))^2,$$

  where Z denotes the information about the zones.

- Often, we do not know where the edges are, i.e., we do not know Z.

- *Idea:* find Z for which the result inside each zone is the smoothest, i.e., minimize

$$J^* = \min_{\text{all possible divisions } Z \text{ into zones}} J(Z).$$

- *Problem:* the resulting problem is no longer convex.

- It is known that non-convex problems are, in general, more computationally difficult.

11. Our First Result: Reconstructing Piecewise Smooth Images is NP-Hard

- *Idea of the proof:* we reduce a known NP-hard problem (subset sum) to our problem.
- *Subset sum:*
  - given m positive integers $s_1, \ldots, s_m$ and an integer $s > 0$,
  - check whether it is possible to find a subset of this set of integers whose sum is equal to exactly s.
- *Alternative description:* check whether there exist $x_i \in \{0, 1\}$ for which

$$\sum s_i \cdot x_i = s.$$

12. Reduction

- We want to reconstruct an $m \times m$ image $f(n_1, n_2)$.
- Let $d = \lfloor m/2 \rfloor$. We want a piecewise smooth image $f(n_1, n_2)$ that consists of two zones.
- The following linear constraints describe the consistency between the observations and the desired image:
  - $f(n_1, n_2) = 1$ for $n_2 > d$;
  - $\sum\limits_{i=1}^{m} s_i \cdot f(i, d) = s$; and
  - $f(n_1, n_2) = 0$ for $n_2 < d$.
- *Problem:* among all the images that satisfy these contraints, find the one with the smallest non-smoothness J^*.

13. Proof

- Let us show that $\min J^* = 0 \leftrightarrow$ the original subset problem has a solution.
- If $J^* = 0$, then all the values within each zone must be the same.
- Since $f = 1$ for $n_2 > d$ and $f = 0$ for $n_2 < d$, every value $f(n_1, n_2)$ is $= 1$ or $= 0$.
- Thus, the values $x_i = f(i, d) \in \{0, 1\}$ solve the original subset problem

$$\sum s_i \cdot x_i = s.$$

- Vice versa:
  - if the selected instance of the original subset problem has a solution x_i,
  - then we can take $f(i, d) = x_i$ and get the solution of the inverse problem for which the degree of non-smoothness is exactly 0.

14. Interval Computations

- *At present:* we select one of the possible images.
- *Idea:* at each pixel (n_1, n_2), present the interval $[\underline{f}(n_1, n_2), \overline{f}(n_1, n_2)]$ of possible values of intensity.
- *Example:* $b(n_1, n_2) = Q(u(n_1, n_2))$, where $Q(u) = 0$ for $u < 0.5$ and $Q(u) = 1$ for $u \geq 0.5$.
- If $b(n_1, n_2) = 0$, then $u(n_1, n_2) \in (0.0, 0.5)$.
- If $b(n_1, n_2) = 1$, then $u(n_1, n_2) \in [0.5, 1.0)$.
- *Comments:*
  - Interval-valued quantities has been successfully used in science and engineering, e.g., in robust control.
  - There exists an interval-based justification of wavelet techniques in image processing (Brito et al.).
  - So, we may get one more justification of wavelets.

15. Towards New Interval-Motivated Inverse Halftoning Techniques

- *Good news:* low-pass filtering of the halftone (binary) image $b(n_1, n_2)$ provides a good first approximation $\ell(n_1, n_2)$ to the original image.
- *Bad news:* the resulting lowpass filtered image is usually still different from the original image $f(n_1, n_2)$: $\ell(n_1, n_2) \neq f(n_1, n_2)$.
- *Reason:* when we apply the original halftoning to the result $\ell(n_1, n_2)$ of applying the low-pass filter to the halftone image $b(n_1, n_2)$, we do not exactly get back the same halftone image.
- *Plan:* modify the lowpass filtered image so that the modified image wil be inverse to halftoning.

16. New Algorithm: Description

- *Case study:* error diffusion.
  - we start processing the image with the pixel (1,1),
  - and then we proceed with pixels (n_1, n_2) with increasing values of n_1 and n_2.
- To invert $b(n_1, n_2)$, we start with (1,1).
- To check whether halftoning of $\ell(n_1, n_2)$ produces the correct value of $b(1,1)$, it is sufficient to apply the above thresholding to the value $\ell(1,1)$:
  - if $b(1,1) = 0$ and $\ell(1,1) \geq 0.5$, then we take $g(1,1) = 0.5 - \varepsilon$ for some small $\varepsilon > 0$;
  - if $b(1,1) = 1$ and $\ell(1,1) < 0.5$, then we take $g(1,1) = 0.5$.

  If they match, we keep $g(1,1) = \ell(1,1)$.

17. New Algorithm: Cont-d

- Once we get to the pixel (n_1, n_2), this means that we have already processed the previous pixels.
- We want to select $g(n_1, n_2)$ at the pixel (n_1, n_2) in such a way that:
  - first, the result of halftoning $g(n_1, n_2)$ is exactly the value $b(n_1, n_2)$;
  - second, if there are several such values $g(n_1, n_2)$, then among these values, we would like to select the value that is the closest to the lowpass filtered image $\ell(n_1, n_2)$.

18. New Algorithm: Details

- So, if $g(n_1, n_2) = \ell(n_1, n_2)$ leads to the correct halftoning, i.e., if the thresholding of $u(n_1, n_2) = \ell(n_1, n_2) - g_0(n_1, n_2)$, where

$$g_0(n_1, n_2) \stackrel{\text{def}}{=} \sum_{m_1, m_2} h(m_1, m_2) \cdot e(n_1 - m_1, n_2 - m_2),$$

  leads to the desired value $b(n_1, n_2)$, then we select $g(n_1, n_2) = \ell(n_1, n_2)$.

- On the other hand, if the result of thresholding $g(n_1, n_2) = \ell(n_1, n_2) + g_0(n_1, n_2)$ is different from $b(n_1, n_2)$, then we take, as $g(n_1, n_2)$, the closest value from the corresponding interval:
  - if $b(n_1, n_2) = 1$, then $g(n_1, n_2) = 0.5 + g_0(n_1, n_2)$;
  - if $b(n_1, n_2) = 0$, then $g(n_1, n_2) = 0.5 - g_0(n_1, n_2) - \varepsilon$.

19. A POCS Iterative Procedure

- The result $\ell(n_1, n_2)$ of lowpass filtering is no longer a valid candidate image.
- $\ell(n_1, n_2)$ is then processed by the interval consistency algorithm – to make it a candidate image.
- This limiting step induces some large local differences in gray level.
- They can be fused back into the image by *frequency swapping:* a 2-D DFT-using projection step that replaces the low-part of the frequency spectrum with that of the halftone input image.
- The limiting and frequency swapping steps are then repeated a few times to produce the final output image.

20. Results and Future Work

- *Advantage of the interval consistency method:* it requires less computations than other methods.
- *Disadvantage:*
  - while it produces an image $g(n_1, n_2)$ whose halftoning produces the exact same result as the original image $f(n_1, n_2)$,
  - for standard benchmark images $f(n_1, n_2)$, the visual difference between $g(n_1, n_2)$ and $f(n_1, n_2)$ is higher than for other metjods.
- *Future work:* come up with intermediate techniques that:
  - may take a little bit longer
  - but provide images which are visually closer to the original ones.

21. Acknowledgments

- This work was supported:
  - by a scholarship from the Texas Instruments Foundation
  - by NASA under cooperative agreement NCC5-209,
  - by NSF grants EAR-0112968, EAR-0225670, and EIA-0321328,
  - by NIH grant 3T34GM008048-20S1, and
  - by Army Research Lab grant DATM-05-02-C-0046