On Inverse Halftoning : Computational Complexity and Interval Computations

S. D. Cabrera and K. Iyer[1]
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[2]
Department of Computer Science
University of Texas at El Paso
El Paso, Texas 79968
e-mail: gxiang@utep.edu, vladik@cs.utep.edu

Abstract — **We analyze the problem of inverse halftoning. This problem 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.**

I. Introduction

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 cannot print the points of different intensity; at any pixel, it 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 form $b(n_1, n_2)$ in which at every pixel (n_1, n_2), we only have 0 or 1: 0 if we do not print a black dot at this location, and 1 if we do. This transformation from the original continuous image to the two-level ("halftone") image is called *halftoning*.

Crudely speaking, the level of intensity at a pixel is represented by the relative frequency of black spots around it:

- if the original image was black, then all the neighboring pixels are black;
- if the original image was white, then none the neighboring pixels are black, they are all white;
- any level between absolute black and absolute white means that some pixels in the neighborhood are black and some are white; the larger the intensity, the more black pixels there is.

Halftoning techniques: a brief reminder

There exist many halftoning algorithms; see, e.g., [23]. One of the most widely used halftoning algorithms is the *error diffusion* algorithm [23], in which we start with the original image $u(n_1, n_2) := f(n_1, n_2)$ and sequentially update the processed image $u(n_1, n_2)$ and quantize the processed value $u(n_1, n_2)$ into the halftone image $b(n_1, n_2) = Q(u(n_1, n_2))$, 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 values $u(n_1, n_2)$ at neighboring pixels, so that the processed value $u(n_1, n_2)$ eventually becomes equal to

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

Need for reverse halftoning

Visually, the halftone image printed on a high-quality laser printer looks practically identical to the original gray-scale image that we can see on the computer screen. So, visually, once we have a halftone image $b(n_1, n_2)$, we can tell which original multi-level image $f(n_1, n_2)$ it came from. However, this intuitive understanding is difficult to describe in precise terms.

Once we have a printed image, we can digitally scan it and get the halftone values $b(n_1, n_2)$ from this printed page. From these halftone values, we would like to reconstruct the original image. Our eyes can do it, but it is not so easy to describe this ability in algorithmic terms.

The need for such a representation also comes from the need to manipulate the original image, e.g., rotate it or zoom on it. These operations are easy to perform on the original image, but it is not clear how to perform them on a halftone image. So, if we want to go from a printed image to a printed zoomed and/or rotated image, we can do it in this way:

- first, we use the halftone image $b(n_1, n_2)$ to reconstruct the original image $f(n_1, n_2)$;
- then, we apply the appropriate zoom and/or rotation operations to the reconstructed image $f(n_1, n_2)$, resulting in a transformed image $f^*(n_1, n_2)$;
- finally, we halftone the transformed image $f^*(n_1, n_2)$, and print the resulting halftone image $b^*(n_1, n_2)$.

In all these cases, we must reverse the halftoning procedure.

[1]This work was supported by a scholarship from the Texas Instruments Foundation

[2]This work was supported in part 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

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 and is, therefore, a lossy compression.

Hence, there may be several different images that lead to the same halftoned image.

Existing inverse halftoning techniques: POCS

There exist several different techniques for inverse halftoning:

One class of such techniques is based on the iterative projection onto convex set (POCS); see, e.g., [7, 15]. Crudely speaking, the main idea behind these methods is that each value $b(n_1, n_2)$ of a halftone image represents a constraint on the original image $f(n_1, n_2)$. In most halftoning methods like error diffusion halftoning, the relation between the original image $f(n_2, n_2)$ and the halftone image $b(n_1, n_2)$ is described by convex transformations, so for each value $b(n_1, n_2)$, the set of all the images that lead to this particular value is a convex set.

Thus, the set of all the images $f(n_1, n_2)$ which are consistent with the halftone image $b(n_1, n_2)$ is also a convex set. Among these images, we want to find an image that satisfies certain reasonable properties, e.g., an image that is sufficiently smooth. For many properties, the set S of such images is convex. We would like to find, among all the images from the class S, the closest to $b(n_1, n_2)$ (e.g., in L^2 metric) that is consistent with the halftone image $b(n_1, n_2)$.

¿From the geometric viewpoint, we have a point $b(n_1, n_2)$ in the function space, and we want to find the closest element to this point in the convex set that is the intersection of the set S of all desirable images and the convex sets formed by all the images that lead to this very halftone image. It is known that to get this projection, we can:

- project our point onto the first of the intersected convex sets, then
- project the resulting point onto the second,
- etc.,

iterating the procedure if necessary. In terms of constraint propagation, we need to satisfy several constraints, so we:

- 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.

The resulting projection on convex sets method indeed leads to a good quality inverse halftoning.

Existing inverse halftoning techniques: wavelet techniques

Another class of techniques for inverse halftoning uses wavelet transform, a techniques that, as the experience of JPEG2000 has shown (see, e.g., [20]), best captures the visual quality of images uncompressed after a lossy compression (and halftoning is, as we have mentioned, an example of lossy compression).

Wavelet-based techniques for inverse halftoning are presented, e.g., in [15]. In accordance with the JPEG2000 experience, wavelet-based inverse halftoning techniques lead to visually the best reconstruction among all known inverse halftoning methods.

Existing inverse halftoning techniques: fast techniques based on adaptive filtering

As we have just mentioned, the wavelet-based inverse halftoning techniques lead to visually the best reconstruction among all known inverse halftoning methods. The only reason why these methods are not universally used is that these methods require a lot of computations.

In view of this fact, researchers have been trying to design faster inverse halftoning techniques that would still lead to similar quality image reconstruction.

The main idea behind such techniques is that, as we have mentioned, the intensity of the original image $f(n_1, n_2)$ can be reconstructed from the density of black pixels in the neighborhood of a given pixel (n_1, n_2). In engineering terms, this means that the original image $f(n_1, n_2)$ can be obtained from the halftone image $b(n_1, n_2)$ by low-pass filtering.

If the image consists of a single object, with intensity smoothly changing from pixel to pixel, then we can indeed apply a low-pass filter to the half-tone image and get a reasonable reconstruction. However, in real life, images have edges. When applied to an image with edges, a low-pass filter correctly reconstructs the intensity within each smooth zone, but blurs the edge.

A natural way to avoid this blurring is to detect the edges and to apply different filters (with different spatial radius) at different parts of the image, so that:

- a filter applied inside each zone would have a larger radius and thus, have a greater smoothing effect, while
- a filter applied closer to the edge would have a smaller radius, and thus, would preserved the edge.

Of course, ideally, instead of just two levels inside-edge, we should have a filter radius adjusted to the estimated gradient of intensity at the given pixel (n_1, n_2).

This idea has been successfully implemented in inverse halftoning; see, e.g., [12]. The resulting method is much faster than the wavelet-based reconstruction, while the visual quality of the reconstructed images is almost as good as for the wavelet-based reconstruction.

Inverse halftoning: remaining problem

The remaining problem is that the existing methods are still not optimal. They are optimized with respect to selection of parameters, but the consensus of researchers is that there is still room for improvement, especially when we are looking for methods of low computational intensity that can be easily implemented within the existing printing devices.

What we are planning to do

In this paper, 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.

II. Inverse Halftoning is a Particular Case of a General Class of Inverse Problems of Reconstructing Piecewise Smooth Images

Inverse halftoning is an ill-posed problem

We have mentioned that the inverse halftoning problem is ill-posed in the sense that, from the purely mathematical viewpoint, there are many different images that are consistent with the same observed data – in this case, with the given halftone image $b(n_1, n_2)$.

Most inverse problems in science and engineering are ill-posed

The above ill-posedness of the inverse halftoning problem is a common feature in applications: most inverse problems in science and engineering are ill-posed; see, e.g., [21].

Smoothness: traditional approach to solving ill-posed inverse problem

A typical way to solve an inverse problem is to find a natural physically meaningful property of actual solution, and use this *a priori* information to select a single most physically meaningful solution among many mathematically possible ones. This process is called *regularization.*

Typically, in inverse problems, this natural property is smoothness. Smoothness can be naturally described in precise mathematical terms. For example, when we reconstruct a 1-D signal $x(t)$, then the degree of smoothness can be defined as follows. At a given moment of time t, the larger the absolute value $|x'(t)|$ of the derivative $x'(t)$, the less smooth the signal is. Thus, at a given time t, the value $|x'(t)|$ is a natural degree of the signal's non-smoothness. Overall, a natural degree of non-smoothness can be defined as a mean square of these degrees corresponding to different moments t, i.e., as $J \stackrel{\text{def}}{=} \int (x'(t))^2\, dt$.

Most regularization techniques try to find, among many signals that are consistent with given observations, the smoothest signal, i.e., the signal with the smallest possible value of the degree of non-smoothness J.

Smoothness: discrete case

In real life, we only have the values $x(t_1)$, $x(t_2)$, ..., of the signal $x(t)$ at discrete moment of time t_1, $t_2 = t_1 + \Delta t$, ..., $t_{i+1} = t_i + \Delta t$, ... Based on this discrete data, we can approximate the derivative $x'(t)$ as a difference $\dfrac{x(t_{i+1}) - x(t_i)}{\Delta t}$, so minimizing the integral J is equivalent to minimizing the corresponding integral sum $J_{\text{discr}} \stackrel{\text{def}}{=} \sum\limits_i (x(t_{i+1}) - x(t_i))^2$.

Smoothness: 2-D case

For a 2-D image $f(n_1, n_2)$, similarly, a natural assumption is that this image is smooth. Similarly to the 1-D case, a natural way to describe the degree of smoothness of a given image is to use the integral sum

$$J \stackrel{\text{def}}{=} \sum_{n_1, n_2} [(f(n_1+1, n_2) - f(n_1, n_2))^2 + (f(n_1, n_2+1) - f(n_1, n_2))^2].$$

Alternatively, we can describe this criterion as the sum of the squares of the differences in intensity between all possible pairs (p, p') of neighboring pixels $p = (n_1, n_2)$ and $p' = (n_1', n_2')$:

$$J = \sum_{p, p' \text{ are neighbors}} (f(p) - f(p'))^2.$$

Smoothness makes problems computationally solvable

A practically useful property of the above degrees of non-smoothness J is that the expression J is a convex function of the signal $x(t_i)$ or $f(n_1, n_2)$. Thus, if the conditions describing the fact that the unknown images is consistent with the observations is also described by linear or, more generally, smooth inequalities, then the problem of finding the regularized solution can be reformulated as a problem of minimizing a convex function J on the convex set.

Similarly, if we fix the degree of non-smoothness and look, among all the solutions with a given degree of non-smoothness, for the one that is the closest to the original approximate solution, we also have a problem of minimizing a convex function (distance) on the convex set (of all functions that are consistent with the observations and have the desired degree of smoothness).

It is known that, in general, the problems of minimizing convex functions over convex domains are algorithmically solvable (see, e.g., [22]), and smoothness-based regularization has indeed been efficiently implemented; see, e.g., [21].

For image reconstruction, we only have piecewise smoothness

We have already mentioned that in images, we have a smooth change from pixel to pixel only within an object; between objects, we may have a sharp edge in which there is no smoothness at all.

Many other inverse problems can also be characterized by piecewise smoothness. For example, in the inverse problem of geophysics, we use the results of ultrasound waves passing through the earth to find out how the density change with depth and location. In geophysics, we have clear layers of different rocks with sharp edges between different layers, so we also face an inverse problem with only piecewise smoothness; see, e.g., [19].

Traditional smoothness measures are not adequate for piecewise smoothness

In the piecewise smooth case, the above measure of non-smoothness is not applicable, because it would include neighboring pixels on the different sides of the edge.

Appropriate smoothness measures for piecewise smoothness case

To avoid the above problem, we need to only take into account the pairs of neighboring pixels that belong to the same zone, i.e., consider the sum

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

where Z denotes the information about the zones. This measure makes computational sense only if we know beforehand

where the zones are – i.e., where is the border between the two zones.

However, in real life, finding the edge is a part of the problem. In this case, we can use the same smoothness criterion not only to reconstruct the original image, but also to find the edge location. Specifically, we want to look for the zone distribution *and* for the zone location for which the above criterion J takes the smallest possible value.

In terms of an image, we fix the number of zones, and we characterize the non-smoothness of an image by a criterion

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

The resulting problem is no longer convex

The resulting functional is no longer convex, because the division into zones is a discrete problem. It is known that non-convex problems are, in general, more computationally difficult than the corresponding convex ones (see, e.g., [11]), and adding discrete variables makes the problems even more computationally difficult; see, e.g., [18].

What we are planning to do

In the following section, we show that in general, the inverse problem for piecewise smooth case is computationally intractable (NP-hard) even when the relation expressing the consistency between the measured results and the desired image is linear.

This proof will follow the proof of NP-hardness of different image and signal processing problems described in our previous publications [14].

III. Complexity of Inverse Problems of Reconstructing Piecewise Smooth Images

Let us prove that in general, the inverse problem for piecewise smooth case is computationally intractable (NP-hard).

Main idea of the proof: reduction to a subset problem

To prove NP-hardness of our problem, we will reduce a known NP-hard problem to the problem whose NP-hardness we try to prove: namely, to the inverse problem for piecewise smooth images.

Specifically, we will reduce, to our problem, the following *subset sum* problem [14, 18] that is known to be NP-hard:

- 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.

For each i, we can take $x_i = 0$ if we do not include the i-th integer in the subset, and $x_i = 1$ if we do. Then the subset problem takes the following form: check whether there exist values $x_i \in \{0, 1\}$ for which $\sum s_i \cdot x_i = s$.

We will reduce each instance of this problem to the corresponding piecewise smooth inverse problem.

Reduction to a subset problem: details

Let us consider the following problem. 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 $n_2 > d$;
- $\sum_{i=1}^{m} s_i \cdot f(i, d) = s$; and
- $f(n_1, n_2) = 0$ for $n_2 < d$.

The problem that we consider is to find the solution with the smallest possible value of smoothness J^* among all the images that satisfy these linear constraints.

Let us show that the minimum of J^* is 0 if and only if the original instance of the subset problem has a solution. Indeed, if J^* is 0, this means that all the values within each zone must be the same. Since we have values 1 for $n_2 > d$ and values 0 for $n_2 < d$, we must therefore have every value to be equal either to 0 or to 1. Thus, if we have such a solution, the corresponding values $f(i, d) \in \{0, 1\}$ provide the solution to 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.

So, if we can solve the inverse problem for piecewise smooth images, we will thus be able to solve the subset sum problem.

This reduction proves that the inverse problem for piecewise smooth images is indeed NP-hard.

IV. Interval Computations and Inverse Halftoning

Every halftoning algorithm includes a thresholding step

For every halftoning algorithm, including the error diffusion algorithm, there is a thresholding step where we replace the original continuous value with the quantized value. Usually, to decide whether the halftoned value $b(n_1, n_2)$ at a pixel (n_1, n_2) will be 0 (white pixel) or 1 (black pixel), we do some processing on the original image $f(n_1, n_2)$, and then apply thresholding to the resulting value $u(n_1, n_2)$. For example, in the error diffusion halftoning, we compute the auxiliary value $u(n_1, n_2)$, and then compute the halftone as $b(n_1, n_2) = Q(u(n_1, n_2))$, where:

- $Q(u) = 0$ if $u < 0.5$ and
- $Q(u) = 1$ if $u \geq 0.5$.

New idea: instead of selecting a single original image, produce the whole range of possible original images

We have already mentioned that halftoning loses information and is, therefore, a lossy compression. Hence, there may be several different images that lead to the same halftoned image.

In the existing methods, we reverse the halftoning procedure by selecting one of such images. However, it may be beneficial to present not just a *single* possible original image, but the whole *range* of images that could lead to the given halftoned image.

For example, in the above halftoning procedure, if we know that $b(n_1, n_2) = 0$, this means that the signal $u(n_1, n_2)$ could

have any value from the interval $(0.0, 0.5)$, while if we know that $b(n_1, n_2) = 1$, this means that the signal $u(n_1, n_2)$ could have any value from the interval $[0.5, 1.0)$.

Result: interval-valued image

If we follow this idea, then instead of the image in which the intensity $f(n_1, n_2)$ at every pixel has an exact value, we come up with an "interval-valued" image in which, at each pixel (n_1, n_2), we only know the interval $[\underline{f}(n_1, n_2), \overline{f}(n_1, n_2)]$ of possible values of intensity.

A similar idea of interval-valued quantities has been successfully used in science and engineering

The idea of using interval-valued quantities to represent uncertainty in engineering and scientific applications is not new: it has been known since the 1950s when R. E. Moore from Lockheed used it to analyze trajectories of intercontinental missiles and spaceship under interval uncertainty; see, e.g., [16]. Since then, interval computations have been widely applied in different engineering problems; see, e.g., [8, 9, 10, 17].

Interval techniques have been actively used in robust control [9], and in image and data processing [4, 13]. We believe that interval techniques can be applied to the inversion of halftoning as well.

Interval methods may provide one more explanation for the efficiency of wavelets

In particular, there exists an interval-based justification of wavelet techniques in image processing [4].

This explanation makes us believe that interval methods may be able to explain why wavelet-based inverse halftoning techniques lead, at present, to the most accurate (albeit not the fastest) inverse halftoning.

V. Towards New Interval-Motivated Inverse Halftoning Techniques: First Algorithm

As we have mentioned, 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. However, the resulting lowpass filtered image is usually still different from the original image $f(n_1, n_2)$: $\ell(n_1, n_2) \neq f(n_1, n_2)$.

One reason for this difference is that, as we have mentioned, halftoning is a lossy compression. Due to this lossiness, several different images $f(n_1, n_2)$ lead to the same halftone image $b(n_1, n_2)$, so we cannot reconstruct the original image exactly.

However, there is another reason why $\ell(n_1, n_2) \neq f(n_1, n_2)$: namely, 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. In this sense, the lowpass filtered image is not the true inverse to the halftoning procedure.

It is therefore desirable to modify the lowpass filtered image so that the modified image wil be inverse to halftoning, in the sense that if we apply the halftoning procedure to the modified image $g(n_1, n_2)$, we will get exactly the halftone image $b(n_1, n_2)$.

An Interval Consistency algorithm: description

Let us show how we can use interval ideas to design a desired image modification procedure. We will apply these ideas to the most widely used halftoning algorithm – error diffusion. In error diffusion, in order to process a pixel (n_1, n_2), we must have the results of halftoning of pixels $(n_1 - m_1, n_2 - m_2)$ with smaller values of the coordinates. Thus, in this halftoning procedure, 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 the halftone image, we similarly start with the pixel (1,1). The result $b(1, 1)$ of halftoning this pixel depends only on the intensity $f(n_1, n_2)$ at this pixel: $b(n_1, n_2) = 0$ for $f(n_1, n_2) < 0.5$ and $b(n_1, n_2) = 1$ for $f(n_1, n_2) \geq 0.5$. So, 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 the result of this thresholding coincides with $b(1, 1)$, we keep the lowpass filtered value $\ell(1, 1)$, i.e., we set $g(1, 1) = \ell(1, 1)$.

On the other hand, if the result of thresholding $\ell(1, 1)$ is different from the halftone value $b(1, 1)$, then we would prefer to select $g(1, 1)$ from the corresponding interval $(0.0, 0.5)$ or $[0.5, 1.0)$ of values that lead to the correct $b(1, 1)$. As we have mentioned, ideally, we should keep the entire interval as an interval of possible values of the original image; however, in this first algorithm, we select a single point $g(1, 1)$ within this interval – the point which is the closest to the lowpass filtered value $\ell(1, 1)$.

In other words:

- if $b(1, 1) = 0$ and $\ell(n_1, n_2) \geq 0.5$, then we take $g(n_1, n_2) = 0.5 - \varepsilon$ (where ε is a small positive number, e.g., the smallest positive floating point number representable in the given computer), and
- if $b(1, 1) = 1$ and $\ell(n_1, n_2) < 0.5$, then we take $g(n_1, n_2) = 0.5$.

After producing $g(1, 1)$, we proceed to the next pixel, etc. Once we get to the pixel (n_1, n_2), this means that we have already processed the previous pixels. This means that we have already produced the values $g(n_1', n_2')$ for all the coordinates $n_1' < n_1$ and $n_2' < n_2$, and, correspondingly, the values $u(n_1', n_2')$ and $e(n_1', n_2')$ (see the description of error diffusion halftoning in Section I).

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 $\ell(n_1, n_2)$.

As we have mentioned in Section I, the value $b(n_1, n_2)$ of halftoning $g(n_1, n_2)$ is the result of thresholding the linear combination $u(n_1, n_2) = g(n_1, n_2) - g_0(n_1, n_2)$, where

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

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

When $b(n_1, n_2) = 1$, then the corresponding interval for $g(n_1, n_2) + g_0(n_1, n_2)$ is $[0.5, 1.0)$, so the interval of desired values of $g(n_1, n_2)$ is $[0.5 - g_0(n_1, n_2), 1.0)$. Thus, if the lowpass filtered value $\ell(n_1, n_2)$ is not in this interval, we select, as $g(n_1, n_2)$, the closest value from this interval, i.e., $g(n_1, n_2) = 0.5 + g_0(n_1, n_2)$.

Similarly, when $b(n_1, n_2) = 0$ and $\ell(n_1, n_2)$ does not belong to the corresponding interval $(0.0, 0.5 - g_0(n_1, n_2))$, we select, as $g(n_1, n_2)$, the closest value from this interval, i.e., $g(n_1, n_2) = 0.5 - g_0(n_1, n_2) - \varepsilon$.

A POCS iterative procedure

The interval consistency algorithm just described is evaluated in a POCS iterative procedure so that its impact can be evaluated using the Floyd-Steinberg algorithm for error diffusion based halftoning. The input halftoned image is first lowpass filtered using a 5x5 Gaussian lowpass mask. The low pass filtering removes the high frequency halftoning noise as well as all other high frequency information available in the halftoned original. The image as a result of lowpass filtering is not only blurred but is no longer a valid candidate image. That is, the difference between the original halftoned image and the rehalftoned version is not zero. The lowpass filtered image is then processed by the interval consistency algorithm that will make it a candidate image. This limiting step induces some large local differences in gray level which can be fused back into the image by a projection step that replaces the low-part of the frequency spectrum with that of the halftone input image. This step is called frequency swapping which is implemented using the two-dimensional DFT. The limiting and frequency swapping steps are then repeated a few times to produce the final output image.

Results and future work

The main advantage of the interval consistency based halftoning algorithm is that it is simple and should require less computations than other methods. For example, fast techniques from [12] requires several hundred arithmetic operations per pixel.

The related disadvantage of our algorithm is that, 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, e.g., for methods from [12].

It is therefore desirable to come up with intermediate techniques that may take a little bit longer than the above-described fast algorithm but provide images which are visually closer to the original ones.

References

[1] B. R. Barmish, *New Tools for Robustness of Linear Systems*, McMillan, New York, 1994.

[2] S. P. Bhattacharyya, H. Chapellat, and L. Keel, *Robust Control: The Parametric Approach*, Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

[3] G. Bozkurt-Unal and A. E. Cetin, "Restoration of Error-Diffused images using Projection onto Convex Sets", *IEEE Trans. Image Processing*, Vol. 10, No. 12, pp. 1836–1841, December 2001.

[4] A. E. Brito and O. Kosheleva, "Interval + Image = Wavelet: For Image Processing under Interval Uncertainty, Wavelets are Optimal", *Reliable Computing*, 1998, Vol. 4, No. 3, pp. 291–301.

[5] N. Damera-Venkata, T. D. Kite, M. Venkataraman, and B. L. Evans, "Fast Blind Inverse Halftoning", *Proc. IEEE Int. Conf. on Image Processing*, Oct. 4–7, 1998, Chicago, IL, vol. 2, pp. 64–68.

[6] N. Damera-Venkata, T. D. Kite, W. S. Geisler, B. L. Evans, and A. C. Bovik, "Image Quality Assessment Based on a Degradation Model", *IEEE Transactions on Image Processing*, Vol. 9, No. 4, pp. 636–650, Apr. 2000.

[7] S. Hein and A. Zakhor, "Halftone to Continuous Tone Conversion of Error-Diffusion Coded Images," *IEEE Transactions on Image Processing*, Vol. 4, No. 2, pp. 208–216, February 1995.

[8] Interval computations website `http://www.cs.utep.edu.interval-comp`

[9] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied interval analysis: with examples in parameter and state estimation, robust control and robotics*, Springer Verlag, London, 2001.

[10] R. B. Kearfott and V. Kreinovich (eds.), *Applications of interval computations*, Kluwer, Dordrecht, 1996.

[11] R. B. Kearfott and V. Kreinovich, "Beyond Convex? Global Optimization Is Feasible Only for Convex Objective Functions: A Theorem", *Journal of Global Optimization* (to appear).

[12] T. D. Kite, N. Damera-Venkata, B. L. Evans, and A. C. Bovik, "A Fast, High-Quality Inverse Halftoning Algorithm for Error Diffused Halftones", *IEEE Transactions on Image Processing*, Vol. 9, No. 9, pp. 1583–1592, Sep. 2000.

[13] O. Kosheleva, S. Cabrera, B. Usevitch, and E. Vidal, Jr., "Compressing 3D Measurement Data under Interval Uncertainty", *Proceedings of the Workshop on State-of-the-Art in Scientific Computing PARA'04*, Lyngby, Denmark, June 20–23, 2004, Vol. 1, pp. 79–85.

[14] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1997.

[15] M. Mese and P. P. Vaidyanathan, "Optimized halftoning using dot diffusion and methods for inverse halftoning", *IEEE Transactions on Image Processing*, Vol. 9, No. 4, pp. 691–709, Apr. 2000.

[16] R. E. Moore, *Automatic error analysis in digital computation*, Technical Report Space Div. Report LMSD84821, Lockheed Missiles and Space Co., 1959.

[17] R. Muhanna and R. Mullen (eds.), *Proceedings of the NSF Workshop on Reliable Engineering Computing*, Savannah, Georgia, September 15–17, 2004.

[18] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Dover Publications, Inc., Mineola, New York, 1998.

[19] R. L. Parker, *Geophysical Inverse Theory*, Princeton University Press, Princeton, New Jersey, 1994.

[20] D. S. Taubman and M. W. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice*, Kluwer, Boston, Dordrecht, London, 2002.

[21] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*, W. H. Whinston & Sons, Washington, D.C., 1977.

[22] S. A. Vavasis, *Nonlinear Optimization: Complexity Issues*, Oxford University Press, New York, 1991.

[23] P. W. Wong, "Image quantization, halftoning, and printing", In: A. Bovik (ed.), *Handbook of Image and Video Processing*, Academic Press, New York, 2000, pp. 657–667.