Low-Memory Low-Complexity Inverse Dithering

Shiufun Cheung and Robert A. Ulichney

Compaq Computer Corporation, Cambridge Research Laboratory,
Cambridge, Massachusetts, U.S.A.

ABSTRACT

Dithering an image decreases the pixel bit depth and reduces the storage space required in the image buffer while largely preserving the perceptual quality. In some applications, it is desirable to reconstruct the original image; that is, restore the dithered image to its original bit-depth, for further processing or display. In this paper, we present a new color inverse dithering system designed for low-cost implementation. Our algorithm is based on edge-sensitive adaptive low-pass filtering. In order to prevent excessive blurring from low pass filtering, the system uses edge detection methods so that the filters are applied only to regions of constant color or gray level in the original image. One such method exploits the fact that pixel values are only one level away from each other in a constant color region of a dithered image. Another method exploits a priori knowledge of the dither masks. By limiting the number of possible filters, and by restricting the region of support of the filters to a single image line, tremendous implementation advantages can be gained. Our prototype system uses a set of five filters, including a pair that are asymmetric about the origin specifically for application to object edges. In our implementation, the need for multipliers is eliminated by using bit replication for up-multiplication, and by using lookup tables with relatively small numbers of entries for filtering. We have found that our inverse dithering system can restore to a significant degree a dithered image to its original form. It is especially effective for graphics and synthetic images.

Keywords: Inverse Dithering, Inverse Halftoning

1. INTRODUCTION

When images are presented on media or stored in devices that offer low amplitude resolution, dithering is often performed to maintain the illusion of continuous color or grayscale [1]. Artifacts associated with low amplitude resolution, such as false contours, are thereby mitigated.

When the dithered version of images is used for storage or transmission purposes, restoration of the dithered images to an approximation of their original amplitude resolution can enhance their perceptual quality for printing or display. Towards this goal, this paper presents a color inverse-dithering system by which the effects of dithering are partially reversed. Previously suggested approaches to the problem of inverse dithering include wavelet decomposition [2], MAP estimation [3], and other adaptive algorithms [4,5]. Most of these works are targeted towards the special case of inverse dithering in which bitonal images are restored to full grayscale.

Our inverse-dithering system was designed for implementation on a high-end graphics accelerator chip for which space is at a premium. The twin requirements of low memory and low complexity have led to the development of a generalized inverse-dithering scheme based on the simple but effective approach of low-pass filtering.

2. OVERVIEW

2.1. Inverse dithering as part of an image rendering system

Inverse dithering should be discussed in the larger context of an image rendering system. In the broadest terms, an image rendering system would comprise subsystems for dithering, scaling, filtering and color space conversion. However, for the purpose of our discussion, the diagram in Figure 1 showing just the dithering and inverse-dithering subsystems and the image buffer would be sufficient. Furthermore, although our technique is applicable to a full-color image, we will limit our initial discourse to monochromatic images. Discussion of inverse dithering of color images will be deferred to Section 4.

In this simplified representation, image data flows in the following fashion: The dithering system takes as input the pixel values of a monochromatic image $I(x, y)$. This original image will have an amplitude resolution of q' bits per pixel and the pixel values will be within the range

$$I(x, y) \in \{0,1,2,\ldots,2^{q'}-1\}. \quad (1)$$

Figure 1: Simplified block diagram of an image rendering system

The image $I(x,y)$ will be reduced in amplitude resolution to m bits per pixel by the dithering process. There are many known dithering algorithms [1] but the inverse-dithering scheme described in this document can be used to restore images that have been dithered by most algorithms. The dithered image $I_d(x,y)$ will have pixel values within the range

$$I_d(x,y) \in \{0,1,2,\ldots,2^m - 1\}. \tag{2}$$

The dithered image then goes through an image buffer, which, in computer display systems, would typically be the frame buffer. However, in certain cases, it can also represent a communication channel. The reason for amplitude reduction is the limited amount of memory in the buffer for storage or, alternatively, the narrow bandwidth of the channel.

The inverse-dithering system then takes the dithered image $I_d(x,y)$ as input and produces a reconstructed image $I_r(x,y)$ with a higher amplitude resolution, q'' bits per pixel. The amplitude resolutions of the original and reconstructed images are not restricted to be the same, but in a majority of the cases, it would make the most sense for them to be identical. So, in a typical scenario,

$$q' = q'' = q \tag{3}$$

and the reconstructed image will have pixel values within the range

$$I_r(x,y) \in \{0,1,2,\ldots,2^q - 1\}. \tag{4}$$

2.2. Inverse dithering by edge-sensitive low-pass filtering

As mentioned earlier, the inverse-dithering algorithm is based on low-pass filtering. In order to understand why low-pass filtering is suitable for inverse dithering, it is instructive to consider first the theoretical basis upon which the idea of dithering rests. The reason dithering is effective in preserving an illusion of continuous color or grayscale is that visual acuity decreases dramatically at high spatial frequencies, so dither patterns are actually perceived as their local macro averages. Given this known nature of dithering, an algorithm based on adaptive low-pass filtering is developed. This approach is particularly attractive because it is simple and can be easily optimized to achieve low complexity.

However, the application of a low-pass filter to a dithered image can produce the conflicting effects of removing the graininess caused by the dither and blurring the object edges. The former effect is useful; the latter is undesirable. The blurring effect is especially objectionable with synthetically generated images, such as those most commonly displayed on a computer screen.

One way to remedy this problem is to use an edge-sensitive adaptive mechanism. In such a scheme, processing for each pixel is based on a windowed portion of the image, which is typically the area immediately surrounding the pixel. Edge detection is performed on the windowed portion. Based on the result, an appropriate filter is selected. The chosen filter is then applied to produce the reconstructed pixel value.

In the following section, we will describe the adaptive inverse-dithering scheme in more detail.

3. GENERAL INVERSE-DITHERING ALGORITHM

Figure 2 is an overall block diagram of the inverse-dithering algorithm. The system, as described in the previous section, takes as input the pixel values of a monochromatic dithered image $I_d(x,y)$ and produces an inverse-dithered image $I_r(x,y)$ as output. The design of the filter set and the operation of the three modules, windowing, filter selection and filtering & bit-depth increase, will be discussed in detail in subsequent subsections.

Inverse dithering of an image is performed on a pixel-by-pixel basis. In most cases, the pixels will be processed in the order in which they were generated or read from memory. Typically this will be in a serial raster fashion; that is, pixel-by-pixel from the left of the image to the right and from the top to the bottom. In theory, the order in which the pixels are processed does not affect the result. However, in practice, it does have some bearing on various optimization issues of the process.

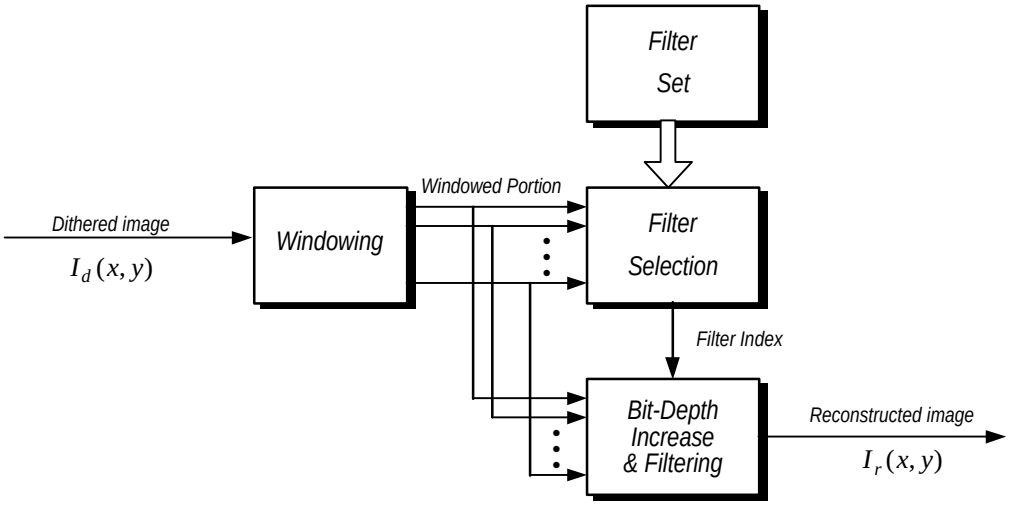


Figure 2: Block diagram of inverse-dithering system

3.1. Design of the filter set

In order to keep the complexity of the system low, the filters used are limited to a finite set. The design of the filter set is therefore crucial to the performance of the system. In the following we lay down some of the design constraints. For obvious reasons, the filters are normalized such that the coefficients sum to unity. They are also restricted to have finite impulse responses. For the sake of future discussion, we will introduce two notations here. The sets R_i represent the region of support of the filters.

$$R_i = \{(x, y) \quad | \quad h_i(x', y') \neq 0 \quad \forall \quad x' \leq x \text{ and } y' \leq y\} \tag{5}$$

where i is the filter index. The values E_i denote the number of non-zero coefficients within R_i.

3.1.1. Filter design—symmetric and asymmetric filters

For the adaptive mechanism to be effective, filters of varying regions of support are necessary. Special filters are also needed for pixels near object edges. In the proposed algorithm filters are restricted to be one of two kinds:

(a) Horizontally even symmetric filters where

$$h_i(x_0, y_0) = h_i(-x_0, y_0) \text{ for all } (x_0, y_0) \tag{6}$$

(b) Part of a horizontally asymmetric pair, $h_i(x, y)$ and $h_j(x, y)$, which are defined to be

$$h_i(x_0, y_0) = h_j(-x_0, y_0) \text{ for all } (x_0, y_0). \tag{7}$$

The asymmetric filters are typically designed to be one-sided, meaning that their region of support are either restricted to $x \geq 0$ or $x \leq 0$. In general, the processing of most pixels would call for the even symmetric filters. Asymmetric filters are applied only to pixels near an object edge.

3.1.2. Ordering of the filters

The filter index i represents the order in which the filters are considered in the selection process. Without loss of generality, numbering of i is assumed to start at 1 and increase without interruption in increments of 1. The index 0 is reserved for the choice of no filtering.

The ordering is subject to three constraints:

(a) For the filter selection process to make sense, filters of lower index numbers should extend at least in one direction beyond filters of higher index numbers. More formally, the region of support of a filter of index i cannot be a subset of the region of support of a filter of index $> i$, or

$$\forall \quad i < j \qquad R_i \not\subset R_j \,. \tag{8}$$

(b) Asymmetric filter pairs should have adjacent indices.

(c) Symmetric filters with the same region of support, which we will call *identical-support filters*, should have adjacent indices. They should also be ordered in terms of increasing cutoff frequencies

Section 3.3.1 will contain a more detailed description of how the selection process works.

3.2. Windowing

The first module of the inverse-dithering system performs a "windowing" of the image. This serves to highlight the neighborhood of the pixel currently under processing. Given that the pixel values are normally read into the inverse-dithering system in a raster fashion, the windowing module will have to contain enough memory to act as a buffer for storing the windowed portions of the image and all the pixels that would be read in between. Figure 3 shows the region of pixels that have to be stored for a given pixel. We observe that, by limiting the vertical extent of the window to one image line as in Figure 3 (b), the required memory can be minimized.

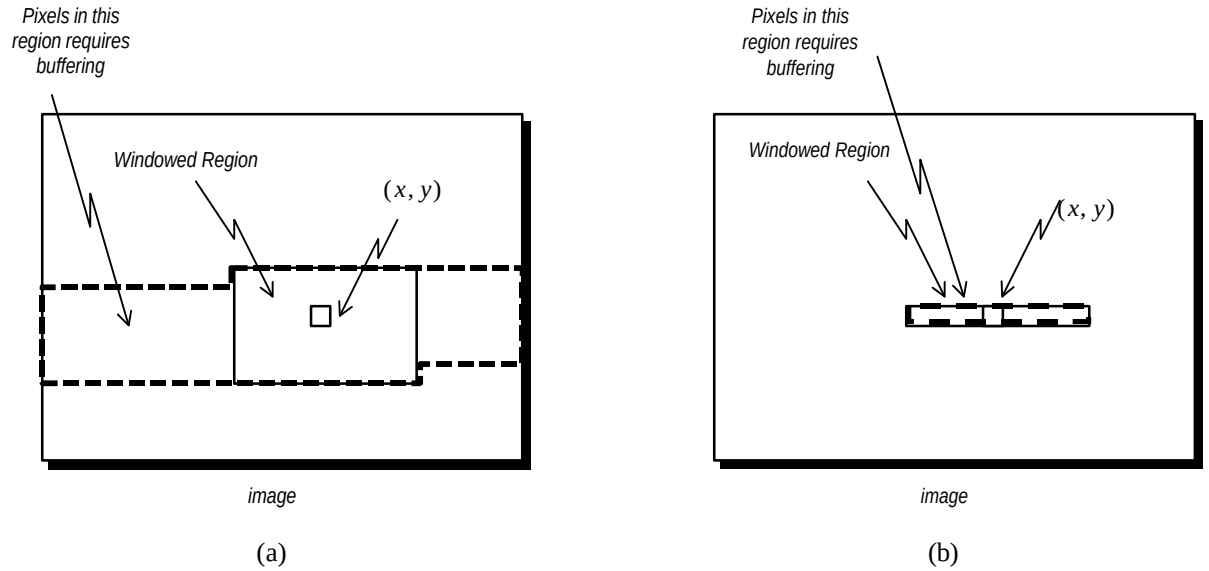


Figure 3: Diagram illustrating the buffer needed to inverse dither an image when (a) the windowed region is multi-line and (b) the windowed region is single-line.

In the case where the windowed region extends beyond the boundaries of the image, we have several options. Periodic extension is often used. Another simple method is to set the values outside of the defined image to zero.

3.3. Filter selection

The second module in the inverse-dithering system performs a filter selection based on the values of the pixels within the windowed region. The single constraint is that the filter selected should only perform low-pass filtering on a region which, at

the original amplitude resolution, would be at a constant color or constant gray level. In other words, the region of support of the chosen low-pass filter should not lead to the filtering of any object edges.

3.3.1. Selection process

Figure 4 is a flowchart depicting the filter selection process. Given that the filters have already been ordered, the selection process starts by setting the filter index i to 1. If an edge is detected in the region of support R_i of that particular filter, the index is incremented, and the edge-detection process is repeated until all the filters in the filter set is exhausted. If no edge is detected, the filter is selected and the index is returned unless it is one of an asymmetric pair or one of a group of identical-support filters.

When no edge is detected in R_i, the process checks to see if the filter is one of an asymmetric pair. If the filter is part of an asymmetric pair, edge detection is performed on the other filter in the asymmetric pair. If edges are absent in the regions of support of both asymmetric filters, the process returns 0, the "no-filtering" option. Otherwise, the process returns the current filter index.

In the case where no edge is detected in R_i and the i^{th} filter is part of a group of identical-support filters, the process will first count the number of filters in the group. This number will be stored in j. The process then calculates the activity measure $A(R_i)$ of the region of support, which is given by

$$A(R_i) = \min\left[\left(\sum_{(u,v)\in R_i} \left|I_d(u,v) - I_d(x,y)\right|\right), \left(E_i - \sum_{(u,v)\in R_i} \left|I_d(u,v) - I_d(x,y)\right|\right)\right]. \tag{9}$$

Using the activity measure, we can then select the $(k+1)^{\text{th}}$ filter in the group, where k satisfies the following criterion:

$$\frac{k}{j} < \frac{2A(R_i)}{E_i} \le \frac{k+1}{j}. \tag{10}$$

This can be rewritten as

$$k = \text{ceiling}\left(\frac{2jA(R_i)}{E_i}\right) - 1. \tag{11}$$

Once k is calculated, the filter index $(i+k)$ is returned.

3.3.2. Edge detection

Edge detection is performed to ensure that low-pass filtering is done only on regions that are originally of constant color or gray level. In a dithered image, such regions are marked by the characteristic that the pixel values are always within one level of each other. In some cases, there are recognizable patterns in the dithered region. These special characteristics lead to edge-detection methods that are unique and simple to implement.

3.3.2.1. Difference maps

One simple edge-detection method examines the map of the differences between the pixels in the region and the center pixel that is under processing. For the region to be declared edge-free, the difference map has to consist of either all zeros and ones or all zeros and negative ones.

Table 1 shows examples of the detection process in action on different 5×1 regions.

Table 1: Examples of edge detection on different regions

Pixel Values (center in bold)	Difference Map	Edge Detection
5, 4, **5**, 5, 4	0, -1, 0, 0, -1	no edge
6, 7, **6**, 6, 5	0, 1, 0, 0, -1	edge
6, 7, **6**, 6, 6	0, 1, 0, 0, 0	no edge
5, 7, **4**, 5, 4	1, 2, 0, 1, 0	edge

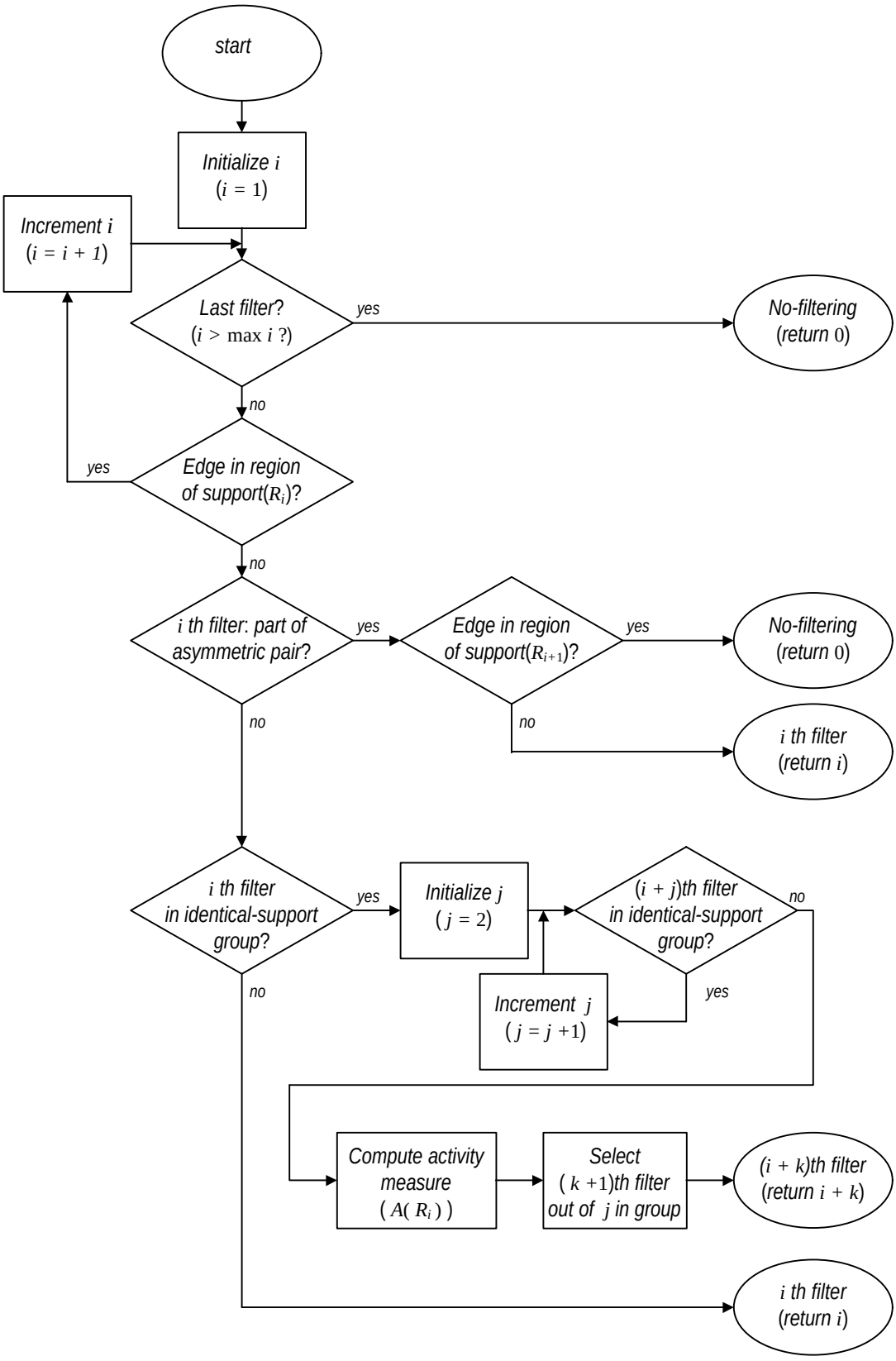


Figure 4: Flow chart showing the filter selection process

3.4. Pattern matching

It is easy to see that the edge-detection method of difference maps is not perfect. In many cases, false negatives are generated where the mechanism fails to detect edges in the image. This leads to undesirable blurring of the restored image. For certain dithering methods, we can improve the edge detection by observing that there are only a limited number of patterns that can occur in the difference maps of regions of constant color or gray level. For example, consider the dither patterns generated by a 8×1 one-dimensional recursive tessellation array such as

$$[0,4,2,6,1,5,3,7]$$

The valid dither patterns are shown in Table 2.

Table 2: Valid dither patterns if dithering is by an $\mathbf{8\times1}$ recursive tessellation array

Valid Dither Patterns	
0, 0, 0, 0, 0, 0, 0, 0	0, 0, 0, 0, 0, 0, 0, 1
0, 0, 0, 1, 0, 0, 0, 1	0, 0, 0, 1, 0, 1, 0, 1
0, 1, 0, 1, 0, 1, 0, 1	0, 1, 0, 1, 0, 1, 1, 1
0, 1, 1, 1, 0, 1, 1, 1	0, 1, 1, 1, 1, 1, 1, 1

If the region of support of interest is 5×1, it is easy to tell, just by simple observation, which 5×1 difference maps are possible. For example,

0, 1, 0, 1, 0, 1, 0, 1 ➔ 0, -1, 0, -1, 0

0, 1, 0, 1, 0, 1, 0, 1 ➔ 0, 1, 0, 1, 0

All valid 5×1 difference maps are shown in Table 3. A more sophisticated edge detection mechanism will therefore use pattern matching to reduce the number of false negatives.

Table 3: Valid $\mathbf{5\times1}$ difference maps if dithering is by an $\mathbf{8\times1}$ recursive tessellation array

Valid Difference Maps		
±1, 0, 0, 0, 0	0, 0, 0, 0, ±1	0, ±1, 0, ±1, ±1
0, ±1, 0, 0, 0	±1, 0, 0, 0, ±1	±1, ±1 , 0, ±1, 0
0, 0, 0, ±1, 0	0, ±1, 0, ±1, 0	±1, ±1, 0, ±1, ±1

If the phase by which the dither template was originally applied to the image is known, we can reduce the number of valid difference maps at each specific pixel location. Using the more restrictive table of difference maps at each pixel location for pattern matching, it is possible to further reduce the number of false negatives arising in edge detection.

3.5. Filtering and bit-depth increase

As shown in Figure 5, the last module in the inverse-dithering system increases the bit depth of the input pixel value to the correct amplitude resolution and then performs filtering. The choice of filter is determined by the filter-selection module.

3.5.1. Bit-depth increase

Input pixel values to the filter and bit-depth increase module have a resolution of m bits. Reconstructing them to q bits requires multiplication by a gain G

$$G = \frac{2^q - 1}{2^m - 1}. \tag{12}$$

Therefore, the up-multiplied pixel values $I_s(x, y)$ is given by

$$I_x(x, y) = G \cdot I_d(x, y). \tag{13}$$

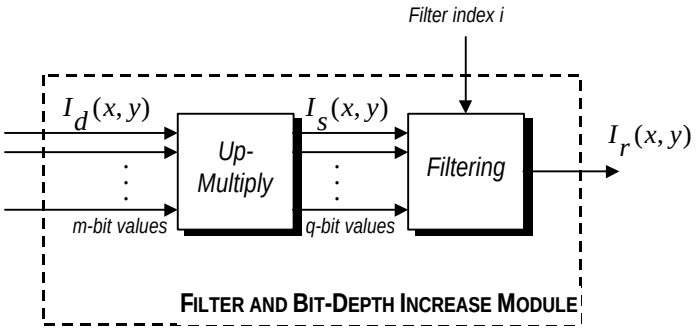


Figure 5: Block diagram of filter and bit-depth increase module

3.5.2. Filtering

Assuming that all the pixel values in the window have been similarly reconstructed to q bits and given the output filter index i from the filter selection module, filtering is performed on the scaled pixels in the usual fashion as follows:

u will contain $+ X_{i2} +)$ terms whereas the second summation over v will contain $(Y_{i1} + Y_{i2} + 1)$ terms.

Note that this filtering process may yield output values that are beyond the range of q bits. When this happens, values above $2^q - 1$ are clamped to $2^q - 1$ while values below 0 are clamped to 0.

4. COLOR IMAGES

In the case of color images, each pixel would be a three-dimensional vector which would be either representing the three channels - red, green, blue - as in

$$\mathbf{I}(x,y) = \big(I_R(x,y), I_G(x,y), I_B(x,y)\big), \tag{15}$$

or possibly representing the luminance and the two chrominance channels as in

$$\mathbf{I}(x,y) = \big(I_Y(x,y), I_U(x,y), I_V(x,y)\big). \tag{16}$$

Color images are handled by separating them into their components before inverse dithering as shown in Figure 6.

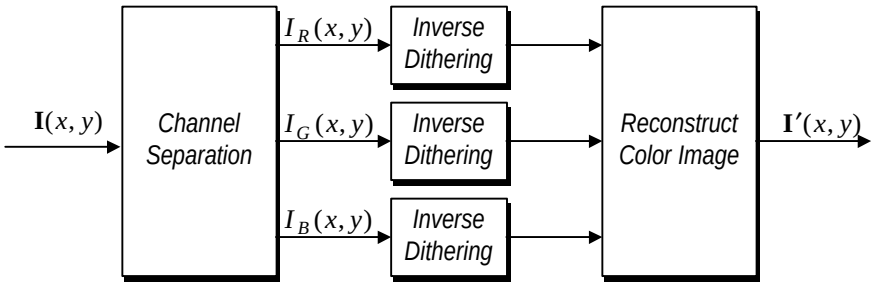


Figure 6: Block diagram for the inverse dithering of color images

There is the possibility of exploiting correlation between the color channels, especially for the step of edge detection. However, our studies indicate that linking the color channels adds considerable complexity to our system without significant quality enhancement.[8]

5. PROTOTYPE SYSTEM AND ITS IMPLEMENTATION

The above description of the inverse-dithering system clearly shows that there is a certain amount of flexibility in its implementation. The filter set, for example, can vary widely from implementation to implementation. Aside from the few aforementioned constraints, the shape and support of the filters are entirely at the discretion of the designer. In the following, we describe the design choices made in our prototype implementation. One of our major discoveries is that limiting the region of support of the filters to a single image line can yield satisfactory results. Another proud achievement is that all multipliers are eliminated from the final system.

5.1. The use of single-line filtering

It is easy to see how the size of the processing window and the filter sizes are related. The window extent must be large enough to cover the region of support of the largest filter. As mentioned earlier, it is highly advantageous to limit the vertical extent of the window to a single image line. That would mean that the filters all have to be one-dimensional. Our studies indicate that while multi-line filtering does yield better results than single-line filtering, the most objectionable artifact caused by using only one-dimensional filters can be rectified by the use of one-dimensional dithering [7]. A further discussion of this can be found in [8].

In the prototype implementation, a set of five filters is used. Each filter has an impulse response of one pixel in height. The horizontal extent was also kept to a minimum. The largest filter has a region of support of 9×1 pixels, followed by a filter with a region of support of 5×1 pixels. We then have two 4×1 asymmetric filters. The last filter is 3×1. As will be shown later, this limited set of one-dimensional filters will yield surprisingly good results.

5.2. Edge detection by pattern matching

The use of one-dimensional dithering and the fact that the filter regions of support are small offer the opportunity to use pattern matching for our edge detection without having to create huge lookup tables. It turns out difference map tables for the 9×1, 5×1 and 4×1 regions have 29, 11 and 5 entries respectively.

5.3. Implementation shortcuts

In Section 3.5, a theoretical description of the bit-depth increase and filtering process is given. For practical implementation, there are many ways to improve efficiency.

5.3.1. Filtering

We can rewrite the filtering process in the following manner.

$$\begin{aligned} I_r(x,y) &= \sum_u \sum_v h_i(u,v) I_s(x-u, y-v) \\ &= \sum_u \sum_v h_i(u,v) \cdot G \cdot I_d(x-u, y-v) \\ &= \sum_u \sum_v h_i(u,v) \cdot G \cdot I_d(x,y) + \sum_u \sum_v h_i(u,v) \cdot G \cdot \big(I_d(x-u, y-v) - I_d(x,y)\big) \\ &= G \cdot I_d(x,y) + G \cdot \sum_u \sum_v h_i(u,v) \cdot G \cdot \big(I_d(x-u, y-v) - I_d(x,y)\big) \end{aligned} . \tag{17}$$

As a matter of notation, we can define the correction term, $C_i(x,y)$, as follows

$$C_i(x,y) = \sum_u \sum_v h_i(u,v) \cdot G \cdot \big(I_d(x-u, y-v) - I_d(x,y)\big). \tag{18}$$

Note that the terms $\big(I(x-u, y-v) - I(x,y)\big)$ are the same as the entries of the difference maps described in section 3.3.2 and are always 0 or ±1. This greatly simplifies the computation of the correction term. In fact, a lookup table can be used for its

generation. Take, for example, the case with the 5×1 region in Table 3. There are only 18 possible different cases. The lookup table for the 5×1 filter would only contain 18 entries.

The bit-depth increase and filtering module can now be described as

$$I_r(x,y) = G \cdot I_d(x,y) + C_i(x,y)\,. \tag{19}$$

This is shown in Figure 7.

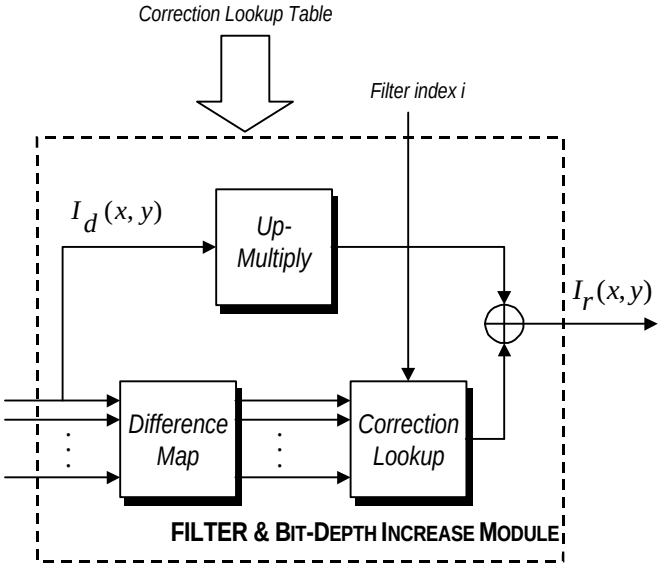


Figure 7: More efficient implementation of filter and bit-depth increase module

5.3.2. Bit-depth increase by bit replication

Using the aforementioned implementation of the filter and bit-depth increase module, only one multiplication per pixel is necessary to reconstruct the q-bit integer value from a m-bit integer value. This up-multiplication step can be approximated by the bit replication process [7]. This eliminates the need for an expensive implementation of a multiplier. The bit-replication process is illustrated in Figure 8 where a 3-bit integer value is scaled to 8 bits.

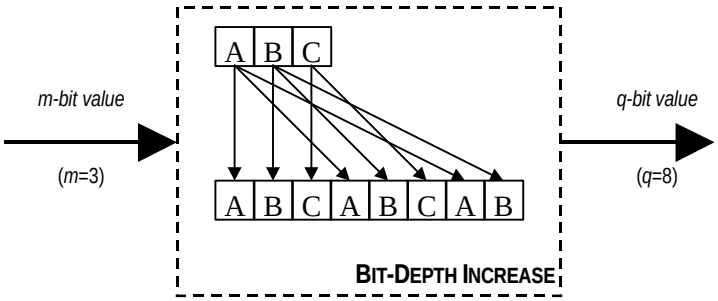


Figure 8: The bit-replication process for bit-depth increase

5.4. Performance

When the inverse-dithering system was applied in conjunction with one-dimensional dithering to 24-bit color images, the perceptual quality of the reconstructed images is very close to the original. This is especially true in cases when the amplitude

reduction is not severe, such as cases where the images are dithered to 12 bits, and when the original image is graphics or synthetic. Figure 9 shows the performance of the inverse-dithering system when the images are dithered to 9 bits (3 bits per color channel). We can see that the reconstruction, while not perfect, is successful to a large degree.

6. CONCLUSION

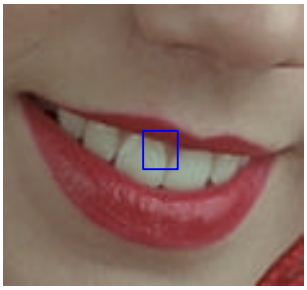
In this paper, we have introduced a color inverse-dithering system based on edge-sensitive adaptive low-pass filtering. By limiting the processing window to one image line, we have minimized the memory requirement. Moreover, by implementing the filtering process with lookup tables and the bit-depth-increase process by bit replication, the system requires no multipliers and complexity is kept low. Performance of this single-image-line no-multiplier implementation of our inverse-dithering system is surprisingly good.

REFERENCES

1. R. Ulichney, *Digital Halftoning*, MIT Press, 1987.
2. Z. Xiong, M. T. Orchard and K. Ramchandran, "Wavelet-Based Approach to Inverse Halftoning," *Proc. of IS&T/SPIE Symposium on Electronic Imaging Science and Technology*, 1997.
3. R. L. Stevenson, "Inverse Halftoning via MAP Estimation," *IEEE Transactions on Image Processing*, vol. 5, no. 4, pp. 574—583, April 1997.
4. C. M. Miceli and K. J. Parker, "Inverse Halftoning," *Journal of Electronic Imaging*, vol. 1, no. 2, pp. 143—151, 1992.
5. A. C. Barkans, "Color Recovery: True-Color 8-Bit Interactive Graphics," *IEEE Computer Graphics and Applications*, vol. 17, no. 1, January/February 1997.
6. R. Ulichney and S. Cheung, "Pixel Bit-Depth Increase by Bit Replication," *Color Imaging: Device-Independent Color, Color Hardcopy, and Graphic Arts III*, Proc. SPIE vol. 3300, pp. 232-241, 1998.
7. R. A. Ulichney, "One-Dimensional Dithering," *Int. Symposium on Electronic Image Capture and Publishing (EICP'98)*, Zurich, Switzerland, May 1998.
8. S. Cheung and R. Ulichney, "Window-Extent Tradeoffs in Inverse Dithering," *The Sixth IS&T/SID Color Imaging Conference: Color Science, Systems and Applications*, November 1998.

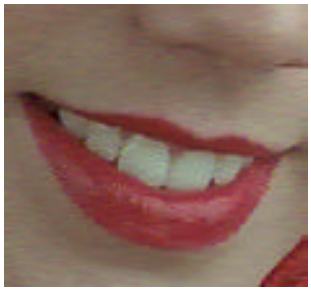
ACKNOWLEDGEMENTS

The authors would like to acknowledge the contributions of Giridharan Iyengar and Robert MacNamara to the development of the inverse dithering algorithm.



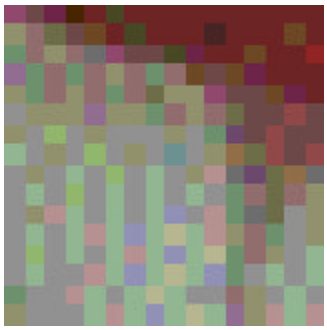
(a) 24-bit Original

(b) Dithered Image



(c) Reconstructed Image

(d)



(e)

(f)

Figure 9: Performance of our One-Dimensional Inverse-Dithering System. The image in (a) was first dithered (each color channel from 8 bits to 3 bits) to form (b) and then reconstructed using the proposed inverse dithering system to form (c). The Images in (d), (e) and (f) are blowups of (a), (b) and (c) respectively. The blowup area is indicated by the blue square in (a).