Modeling and Simulation of a JBIG2 Compliant Color Printer Pipeline

Niranjan Damera-Venkata and Chase Krumpelman

Abstract

We develop a computer aided design (CAD) tool for the simulation and testing of a color image printing pipeline. True color images typically have three color planes (red, green and blue) with eight bits of resolution per pixel per image plane. Printing these images involves a binarization process called halftoning which converts each color plane into a binary image for rendering on binary devices such as printers. While the halftoning algorithm is the heart of the printer pipeline, several preprocessing methods are often required to perform basic image manipulations. These include printer dependent color transformations, image scaling/resizing, and image compression. Thus images at various stages in the printer pipeline are subjected to various types of distortions. Simulation of the printer pipeline is of key importance for end-to-end performance optimization. We model the printer pipeline using the synchronous dataflow (SDF) model of computation in the Ptolemy design environment. We simulate typical tradeoffs in a color printer pipeline using JBIG2 compliant image compression. We also provide an interactive high level interface to key pipeline parameters via MATLAB and/or the Ptolemy graphical user interface (GUI).

Index terms– image halftoning, color imaging, JBIG2, printer pipeline

The authors are with the Center of Vision and Image Sciences, Dept. of Electrical and Computer Eng., The University of Texas at Austin, Austin, TX 78712-1084. E-mail: {damera-v,krumpelm}@ece.utexas.edu.

I. Introduction

Real world images (e.g. photographs) are essentially continuous in tone and in resolution. Representing a real-world images on a computer requires converting a continuous image into a matrix of pixels where each pixel has a value within a finite set of values (color depth). Clearly, this process of sampling and quantization results in some degradation of the original image; however, modern computer monitors have sufficient resolution and color depth that the perceived image quality is very high. Printers, however, generally have signifigantly lower resolutions and color depths than monitors. Most printers are only able to place an ink dot or to not place an ink dot for a given pixel. Images stored in a computer must be converted into binary images for printing, however, straightforward subsampling and quantization of an already sampled and quantized image is generally not acceptable, as it causes severe image degradation and frequency artifacts. The process of "halftoning" attempts solve this problem by creating, through a clever distribution of dots, the illusion of a high-resolution, continuous tone image on a low resolution, finite tone device such as a printer.

Halftoning attempts to preserve as much information as possible in the frequencies where the human vision response is maximal. Halftoning hides information loss well; however, in practical printers, there are other processing stages which can introduce distortion and image quality degradation. Printer pipelines generally contain processing stages for image scaling [1], image compression [1], and color transformations. Image scaling and color transformations are generally printer-specific and based on memory size and available inks. Image compression is based on standards such as JPEG and JBIG [1].

In Fall 1999, the JBIG2 bi-level compression standard was adopted [2], [3]. JBIG2 is designed for lossy compression of halftones for printers and facsimile machines. JBIG2 is a lossy compression algorithm which compresses by coding an image into an array of indices and a dictionary of bitmaps. An image is reconstructed by replacing each index with its corresponding bitmap. The level of loss (and the level of compression) is controlled by the size of the bitmaps (mask size) and the number of entries the bitmap dictionary.

The designer of a typical printer pipeline like the one shown in Figure 1 is faced with preserving image quality through the cascaded image modifications of the pipeline. This

process is an optimization problem with numerous parameters (e.g. halftoning algorithm, JBIG2 compression level) and numerous constraints (e.g. available memory, printer resolution). In order to aid the designer in finding solutions to this problem, we have created a JBIG2 compliant printer pipeline simulator which allows tweaking of essential parameters and provides feedback about the resulting image quality and compression.

II. Modeling

We model the color printer pipeline in synchronous dataflow (SDF) [4] using the Ptolemy design environment. Figure 4 shows the Ptolemy schematic of the pipeline. We have not implemented the color transformation or image scaling blocks becase these blocks are printer specific and can be easily added using standard Ptolemy tool palettes. Without an image scaling block, we assume that the halftone is generated at the required printing resolution. In our modeling and implementation we focus on the two processing steps that have the greatest direct effect on overall printer pipeline performance: vector color error diffusion halftoning [5] and JBIG2 coding/decoding [2], [3], [6].

We take advantage of the feedforward chain-structured nature of the dataflow graph to reduce data memory on the arcs. Several of our stars write data to file (shared memory) and output integer-valued enable tokens instead of producing matrix-valued tokens. This is because we use the shared memory model for our arc buffers. This approach works in a chain structured graph where the minimum memory allocated can never be worse than than the arc buffer model and the buffers are easy to manage (there is only one writer and one reader at a time) [7]. By putting enable tokens on the arcs we are able to conceptually implement the shared buffer model while actually implementing the arc buffer model, and remaining fully compatible with SDF semantics. However the user may choose not to overwrite the shared buffer by explicitly providing filenames as input parameters to the actor.

Our color printer pipeline (shown in Figure 4) begins by reading in a 24-bit color image. The input color image is loaded by the *SDFReadRGB* star and split into color planes which are output as matrix-valued tokens on the red, blue, and green arcs. The vector color error diffusion star (*SDFVecColDiff*) consumes one matrix-valued token on each of its three input arcs and produces one matrix-valued color plane token on each of its three

output arcs. Each plane of the output color error diffused image is then packed into bytes and each color plane is written to file in bitmap (BMP) format, by the *SDFToBMP* utility star. After the conversion is accomplished the *SDFToBMP* star passes an enable token to the *SDFBMPtoPT* star. The *SDFBMPToPT* star converts a file from BMP format to a Ptolemy matrix and then places this matrix as a token on its output arc. Next, the *SDFgenIndPatts* star reads in the color plane image token and converts it to a JBIG2-compliant index array and bitmap dictionary. The *SDFgenIndPatts* star gives the user control over the mask size and shape and the number of bitmaps to be used to code the image. The *SDFgenIndPatts* star stores the index array and bitmap dictionary to shared memory and passes an enable token to the JBIG2 encoder. The JBIG2 encoder codes the index array and bitmap dictionary with context based arithmetic coding. The encoder provides detailed compression statistics about the bitplane being compressed with the given indices and patterns. The encoder then sends an enable token on its output arc to the JBIG2 decoder. The decoder reads in the compressed files, writes the reconstructed halftone in BMP format to a file, and outputs an enable token. The *SDFBMPToPT* star converts the reconstructed bitplane to a Ptolemy matrix token. The three reconstructed matrix tokens corresponding to the three bitplanes are passed to a display star that displays the reconstructed image. The original and reconstructed halftone bitplanes are fed into a MATLAB quality assessment star that reports the weighted signal to noise ratio (WSNR) of the output vs. the input image.

III. Implementation

A. Vector Color Error Diffusion

Fig. 2 shows a block diagram of vector color error diffusion halftoning [5]. When an image pixel is quantized using a bi-level quantizer $\mathbf{Q}(\cdot)$, the quantization error is diffused among the neighbouring pixels. The error filter $\mathcal{H}$ in the feedback loop has matrix valued coefficients. These coefficients have a significant impact on the quality of the resulting halftone. In our implementation the user has control over the error filter coefficients. Vector color error diffusion is implemented in the star *SDFVecColDiff*.

B. The JBIG2 Encoder

We obtained C code for a JBIG2 Encoder from the University of British Columbia (`http://spmg.ece.ubc.ca/jbig2/software/main.html`). This code reads a halftone image file, encodes it with JBIG2 using default index and pattern generation, writes the output bitstream to the file. This code, however, does not provide the user with direct control over the indices and bitmap dictionary. In order to give the user this capability, we rewrote some of the code. First we partitioned the encoder into the index/bitmap generator and the arithmetic coder *SDFJBIG2ModEncoder*, then we wrote the index/bitmap generator in C as its own star *SDFgenIndPatts*. We retained the arithmetic coder of the UBC code by disabling the default index and bitmap generation. Our C bitmap generator implements the index generation method described in [3]. User control over the averaging mask size and shape is provided via configuration files. In order to change the method of index/bitmap generation, a user must change the code that generates the indices and bitmaps. Since the file formats defined for reading and writing the files are fixed by (in accordance with the JBIG2 standard), the user is provided with a high level interface for encoder optimization. We also modified the encoder to generate detailed compression statistics. These are described in more detail in Section IV.

C. MATLAB stars

Blocks which may require sophisticated algorithms/tools were duplicated in MATLAB to provide the user with access to MATLAB's toolboxes. Thus we implemented the index and bitmap generation star (*SDFgenIndPatts*) and the quality assessment star (*colorWSNR.m*) in MATLAB. Thus the user can use any MATLAB function to come up with the indices/bitmaps, or can substitute his/her own customized quality assessment MATLAB function. Our quality assessment star weights the difference in the original halftone and the reconstructed halftone with the contrast sensitivity function and computes the ratio between the total signal power in the three image planes to the total weighted noise power. The result is reported in dB. This measure is called WSNR and is discussed in [8].

D. The JBIG2 Decoder

The JBIG2 decoder is not under user control. It simply interprets a valid JBIG2 bitstream and uses no other information in producing a reconstruction of the original halftone. The JBIG2 decoder is implemented in the *SDFJBIG2Decoder* star.

E. Utilities

The stars *SDFToBMP* and *SDFBMPToPT* are utility stars to convert between Ptolemy matrix tokens and a packed binary file format BMP. These utilities are generic and are suitable for general bi-level image processing tasks.

IV. Results

In this section we illustrate our pipeline simulator by encoding and decoding a sample color halftone. We use the error filter in [5] to generate the original color halftone. The halftone is then passed through the pipeline and the quality of the reconstructed image as compared with the original halftone is reported. The encoder generates compression statistics and reports the encoder parameters used in encoding the halftone. Table I tabulates the information generated by the encoder and the quality assessment star. The original and reconstructed halftones for this example are available for visual assessment at:

`http://www.ece.utexas.edu/ damera-v/halftones.html`

The encoder generated statistics give the user information about how his indices and bitmaps performed in a rate vs visual distortion sense. Individual coding rates of each bitplane and the percentage of bits used to code the pattern dictionary are reported. A 486×700 image requires 32 seconds to pass through the pipeline. The simulation was done on a Sun Ultrasparc 150MHz machine with a load average of 3.25. The worst case spool memory required is about 40Kb if the whole color image is encoded in a single frame. One could also encode/decode a few rows at a time, although this would incur a larger overall overhead in terms of header and synchronization information in the bitstream.

References

[1] R. A. V. Kam, P. W. Wong, and R. M. Gray, "JPEG-compliant perceptual coding for a grayscale image printing pipeline," *IEEE Trans. Image Processing*, vol. 8, pp. 1–14, Jan. 1999.

[2] P. G. Howard, F. Kossentini, B. Martins, S. Forchhammer, and W. J. Rucklidge, "The emerging JBIG2 standard," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 8, pp. 838–848, Nov. 1998.

[3] B. Martins and S. Forchhammer, "Halftone coding with JBIG2," *IEEE Trans. Image Processing*. to appear.

[4] E. A. Lee and D. G. Messerschmitt, "Synchronous data flow," *Proc. of the IEEE*, vol. 75, pp. 1567–1579, Sept. 1987.

[5] N. Damera-Venkata and B. L. Evans, "Design and analysis of vector color error diffusion halftoning systems," *IEEE Trans. Image Processing*, submitted.

[6] M. Valliappan, B. L. Evans, D. A. D. Tompkins, and F. Kossentini, "Lossy compression of stochastic halftones with JBIG2," *Proc. IEEE Conf. Image Processing*, vol. 1, pp. 214–218, Oct. 1999.

[7] B. L. Evans, *Class Notes for EE 382C: Embedded Software Systems*. Spring 2000. The Univerity of Texas at Austin.

[8] 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 Trans. Image Processing*, vol. 9, pp. 636–651, Apr. 2000.

Parameters	Compression statistics	Quality
no: of patterns = 5	CR - red plane = 3.1768	
mask size = 2 x 2	CR - green plane = 3.1546	
size of index array = 243 x 350	CR - blue plane = 3.3748	24.65 dB
pattern dictionary:	overall CR = 3.2325	
$\begin{pmatrix} 0000 & 0001 & 0101 & 0111 & 1111 \end{pmatrix}$	pattern dictionary overhead = 0.12%	

TABLE I

STATISTICS GENERATED BY OUR SIMULATOR.

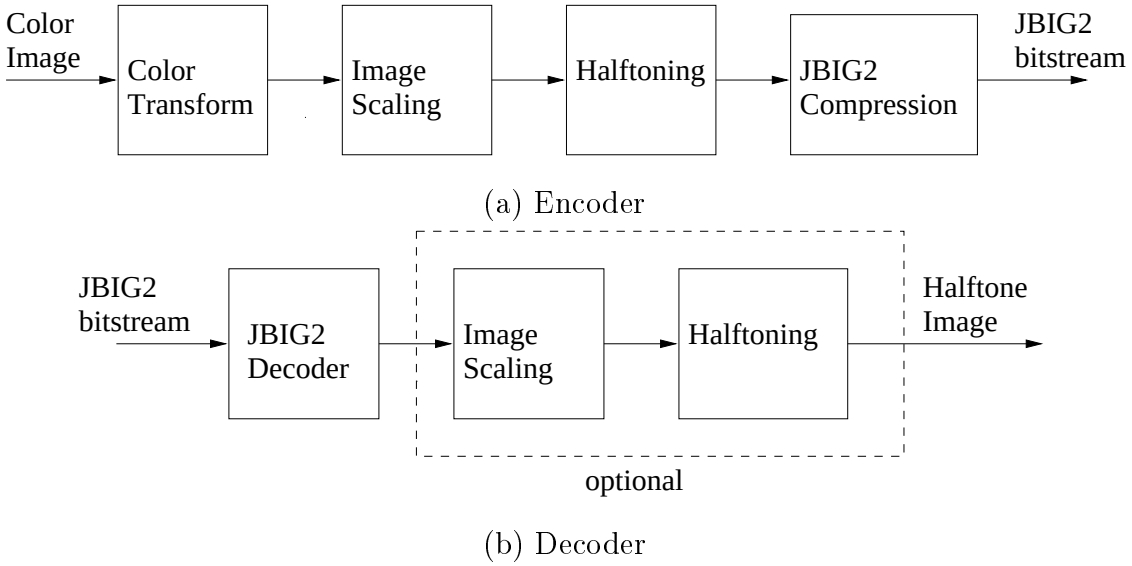

Fig. 1. System block diagram for a color image printing pipeline using JBIG2 compliant halftone coding.

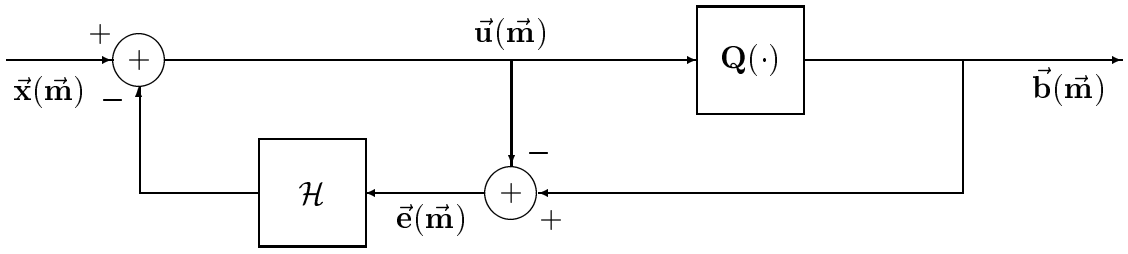

Fig. 2. System block diagrams for vector color error diffusion halftoning where $\mathcal{H}$ represents a fixed 2-D nonseparable FIR error filter with matrix valued coefficients.

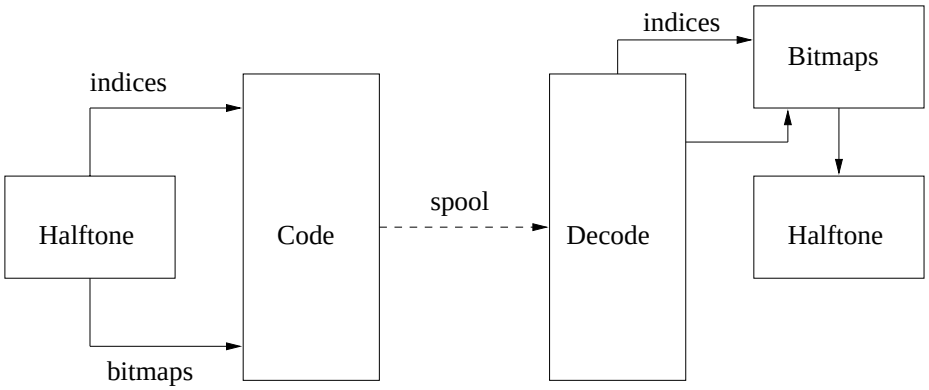

Fig. 3. Block diagram illustrating halftone coding with JBIG2.

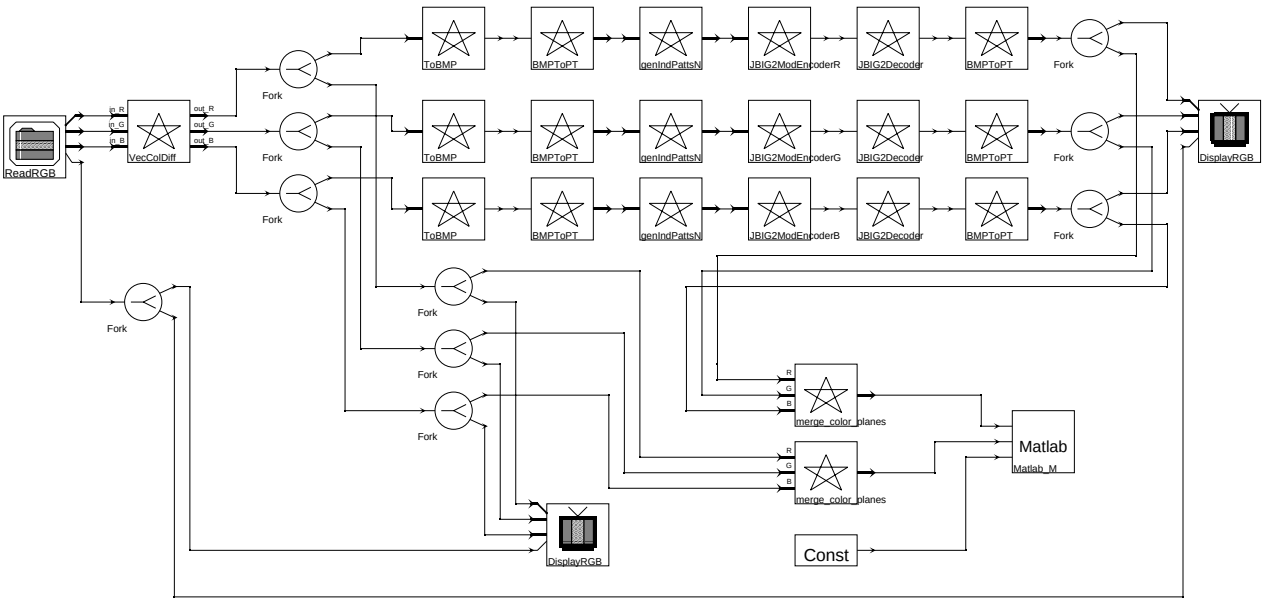

Fig. 4. Ptolemy schematic of the proposed printer pipeline.