JBIG2 Halftones: Analysis and Considerations for T.89

Dave Tompkins and Faouzi Kossentini
University of British Columbia
Contact: davet@ece.ubc.ca

This file and the accompanying bitstreams are available at:
`http://spmg.ece.ubc.ca/jbig2` and `http://spmg.ece.ubc.ca/jbig2/bitstreams`

Introduction

To generate halftoned images in JBIG2, two separate segments are required: a pattern dictionary and a halftone region.

The pattern dictionary contains all of the individual patterns used to generate the halftoned picture. The patterns are concatenated and encoded as one larger region, as illustrated in the following example:

Figure 1: 6x6 Halftone Pattern at 45º.

The pattern dictionary must be decoded and stored in memory before decoding the halftone region segment.

In the halftone region segment, the header section contains numerous parameters which determine how the halftone is to be constructed, such as the details for the halftone grid co-ordinate system.

The data in the halftone region is simply several generic regions. The number of generic regions depends on the number of patterns in the corresponding pattern dictionary. The generic regions can be coded with either the MMR or the MQ coder. After all of the generic regions are decoded, they are combined together as bit planes to construct a multi-level (gray scale) image. Each level (or shade of gray) in the multi-level image corresponds to one of the patterns in the pattern dictionary. For each pixel in the multi-level image, one pattern is placed in the bi-level image.

Angled Halftones

Angled halftones usually look better than non-angled halftones. However, an angled halftone requires more complexity and more memory.

Because the multi-level image is angled over the region, there are elements in the multi-level image that are not used to construct the image. Although these elements can be avoided with the HSKIP parameter, they still require memory (in most practical implementations).

The formula for the multi-level image's width is determined by this awkward formula:

$$HGW = \frac{\cos\left(\tan^{-1}\left(\frac{RW}{RH}\right) - q\right) \cdot \sqrt{RW^2 + RH^2} \cdot \left(\cos(q) + \sin(q)\right)}{HDPW} \qquad (1)$$

where RW and RH are the region width and height, θ is the angle of the halftone, HDPW is the width of each pattern (assuming square patterns for simplicity). The formula for the height is similar, and identical if θ = 45º or RW = RH.

The following figure illustrates how the multi-level image can be larger than the bi-level region.

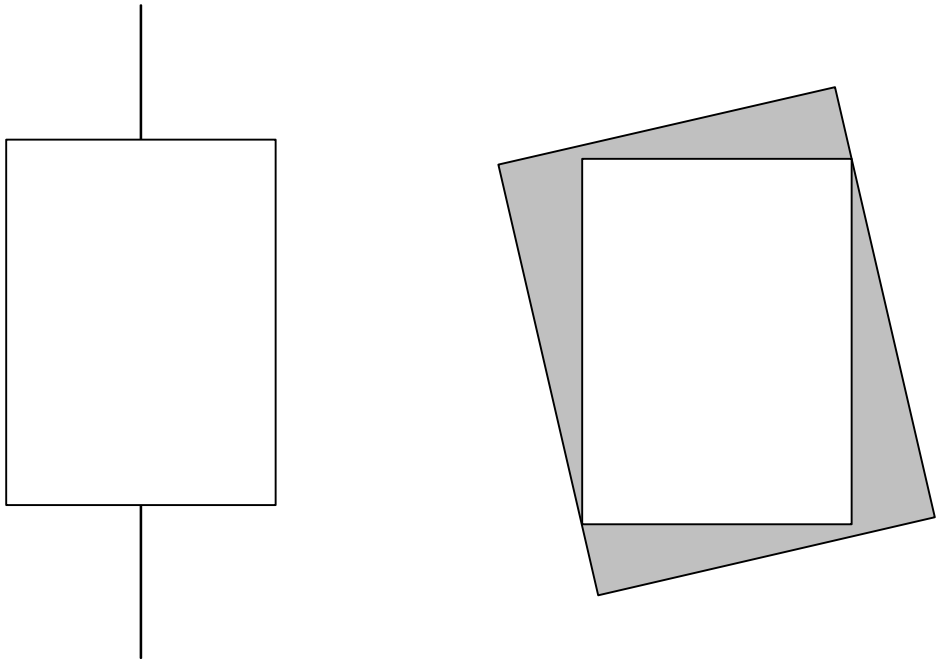

Figure 2: Comparing the sizes of the Gray and Bi-Level Image

From this figure it is also clear and how the halftone angle of 45º requires the most memory.

Assuming RW ~= RH and θ = 45º (worst case scenario for memory), we can approximate the width of the multi-level image with the follow equation:

$$HGW = \frac{\sqrt{2\left(RW^2 + RH^2\right)}}{HDPW} \qquad (2)$$

So for an 8.5" x 11" page at 200dpi and θ = 45º, the grayscale region must be (3932 / HDPW) pixels wide (and high).

Pattern Dictionary Memory

The memory requirements for a pattern dictionary depend on several factors:

- Internal representation of the patterns
- Size of each pattern
- Number of patterns

The patterns could be easily left in their original format, concatenated together. However, it is more likely that they will be separated. If separated, the same structure should be used as the text symbols. In which case, the memory requirements for a pattern dictionary will be:

Pattern Dictionary Memory = (GRAYMAX+1) * (32 + R(HDPW) * HDPH) (3)

Where all the values are defined as in JBIG2, and R() is the rounding function.

For practical implementations, it is reasonable to assume that GRAYMAX <= 255.

Multi-Level Image Memory

In circumstances with small grid sizes, the most important constraint for halftone coding will not be the pattern dictionary memory, but rather the memory required to store the individual bit planes.

Because the bit planes are coded sequentially, all of them must be decoded before the halftone can be constructed. This could require a considerable amount of memory.

Consider the following Table for an 8.5" x 11" page at 200 dpi. Assume square Halftone patterns.

Page Dimensions: 1700 x 2200 (468,600 bytes / 1024 = 458k)

HDPW (HDPH)	Grid Angle (θ)	Gray Image Dimensions	Size of 1 Level	# of Patterns	# of Bit Planes	Image Memory	% of Page Buffer
2	0º	850 x 1100	115k	5	3	345k	75%
3	0º	567 x 734	51k	10	4	204k	45%
3	45º	1311 x 1311	210k	6	3	**630k**	**137%**
4	0º	425 x 550	29k	17	5	145k	32%
4	45º	983 x 983	118k	9	4	**472k**	**103%**
5	0º	340 x 440	18k	26	5	92k	20%
5	45º	787 x 787	76k	13	4	304k	66%
6	0º	284 x 367	13k	37	6	77k	17%
6	45º	656 x 656	53k	19	5	263k	57%
8	0º	213 x 275	7k	65	7	51k	11%
8	45º	492 x 492	30k	33	6	179k	39%
10	0º	170 x 220	5k	101	7	33k	7.2%
10	45º	394 x 394	19k	51	6	115k	25%
20	0º	85 x 110	1k	401*	9*	11k	2.4%
20	45º	197 x 197	5k	201	8	38k	8.5%

Table 1: Gray Image Memory Requirements at 200 dpi

(*) It is unlikely that more than 256 levels of gray (# patterns) or 8 bit planes would be used.

From the table, we can see how in some circumstances, the multi-level image can require ***more*** memory than the actual page buffer itself. For higher resolutions, the % of page buffer would be approximately the same.

Striping Artifacts

Striping is used to break a large image into several, smaller *stripes*. Although there is a compression penalty associated with striping, it is popular because it significantly reduces the memory (buffer) requirements.

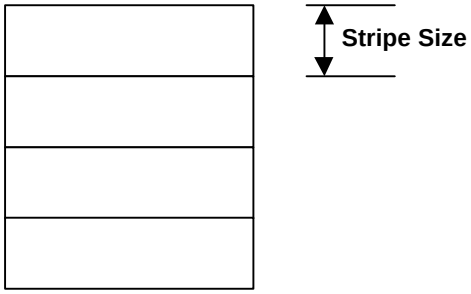

Figure 3: JBIG2 Stripes

With halftone coding, it is possible that the striping of a page will introduce some visible artifacts at the stripe boundaries. With some careful encoding, these artifacts may be eliminated, but it may be very difficult when using angled halftones.

With non-angled halftones, the problem can be avoided by using a stripe size that is divisible by the pattern height (HDPH).

With angled halftones, there are several concerns. The first concern is that the grids may not align properly. Careful calculations for the starting grid position and grid offsets will have to be made to ensure the patterns will align. Another concern is stripe overlap. To avoid any artifacts, the encoder will have to consider regions of the page that are in the adjacent stripes -- which may be awkward for some encoder implementations. And finally, striping may reduce the feasibility of performing filtering or image processing on the grayscale image. These techniques often reduce the image size slightly or produce negligible artifacts at the edges of the image, which will become artifacts at stripe boundaries when striping is used.

Admittedly, not a lot of resources have been invested in developing techniques for analyzing and reducing the artifacts associated with striped halftones.

Summary & T.89 Recommendations

Halftone coding in JBIG2 can achieve a very high compression rates, and could be very popular in applications where lossy compression is acceptable.

There are two different types of memory constraints:

- Pattern Dictionary Memory
- Gray Image Memory

For small grid sizes, the Gray Image memory is of the greater concern. Tables 1 and 2 demonstrate the large Gray Image requirements when small grid sizes are used. To avoid excessive Gray Image memory, striping can be used. However, unless special care is taken, striping can introduce some unwanted artifacts in the image.

For larger grid sizes, the Pattern Dictionary Memory becomes more of a concern. Tables 2 and 3 list the Pattern Dictionary requirements for both small and large grid sizes.

We recommend that memory requirements for both the Gray Image and the Pattern Dictionary are included in the same general buffer of memory as the Symbol Dictionaries. Because of the large memory buffers that could be involved, it would be wasteful to allocate a separate buffer of memory for either the pattern dictionary or the gray level image. We assume that the memory available for symbol dictionaries would be at least as large as the page buffer -- which would be large enough to support most practical halftones. There is no real need to place restrictions on the halftone parameters, as the memory requirements will limit the availability of gray levels and grid size. However, a limit on the number of gray scales to 256 (8 bit planes) would allow for more straightforward implementations.

HDPW (HDPH)	Grid Angle (θ)	# of Patterns	# of Bit Planes	Pattern Memory	Image Memory	Comp. Ratio	Striped Image Memory	Striped Comp. Ratio
2	0º	5	3	0.2k	345k	6 : 1	40k	5 : 1
3	0º	10	4	0.4k	204k	12 : 1	24k	7 : 1
3	45º	6	3	0.3k	630k	7 : 1	242k	4 : 1
4	0º	17	5	0.8k	145k	16 : 1	17k	10 : 1
4	45º	9	4	0.4k	472k	11 : 1	181k	6 : 1
5	0º	26	5	1.3k	92k	20 : 1	11k	12 : 1
5	45º	13	4	0.7k	304k	14 : 1	116k	7 : 1
6	0º	37	6	2.0k	77k	24 : 1	9k	16 : 1
6	45º	19	5	1.0k	263k	17 : 1	101k	9 : 1
8	0º	65	7	4.1k	51k	30 : 1	6k	23 : 1
8	45º	33	6	2.1k	179k	22 : 1	68k	13 : 1
10	0º	101	7	7.1k	33k	37 : 1	4k	29 : 1
10	45º	51	6	3.6k	115k	27 : 1	44k	17 : 1
20	0º	401	9	43.9k	11k	71 : 1	1k	60 : 1
20	45º	201	8	22.0k	38k	49 : 1	15k	39 : 1

Table 2: Memory Requirements for a Full Page 200 dpi Image.
Compression Ratios are listed for figure Miriam200
Strip Size for this test is 256 pixels

HDPW (HDPH)	Grid Angle (θ)	Theoretical # of Patterns	Theoretical Pattern Memory	Practical # of Patterns*	Practical Pattern Memory*
10	0º	101	7k	101	7k
10	45º	51	4k	51	4k
20	0º	401	44k	256	28k
20	45º	201	22k	201	22k
30	0º	901	134k	256	38k
30	45º	451	67k	256	38k
40	0º	1601	550k	256	88k
40	45º	801	275k	256	88k
50	0º	2501	1055k	256	108k
50	45º	1251	528k	256	108k
60	0º	3601	1801k	256	128k
80	0º	6401	6201k	256	248k
100	0º	10001	15939k	256	408k
150	0º	22501	66624k	256	758k

Table 3: Pattern Dictionary Requirements for Large Grid Sizes
(* For Most Practical Implementations, 256 levels of gray will be sufficient)