Fast Inverse Halftone for Dispersed-dot Ordered Dithering

Pedro Garcia Freitas
Mylène C.Q. Farias
Aletéia P. F. de Araújo

Department of Computer Science
University of Brasília, (UnB)
Brasília, Brazil
Dither & Halftone

• Halftone is the reprographic technique.

• Simulates continuous tone imagery through the use of dots.

• Varying either in size, in shape or in spacing.
Dither & Halftone

- Use of noise to randomize quantization error

- Prevents large-scale patterns, such as 'banding’ in images.

- Dither: Used to generate halftone images.
Dither & Halftone

Dithering with error diffusion

Ordered Dithering

Fast Inverse Halftoning Algorithm for Ordered Dithered Images
GENERATING DITHERED HALFTONE
Halftone: Ordered Dithering

Used Method: Dispersed-dot Ordered Dithering-related:
Quantize each pixel from 8-bit/pixel to 1-bit/pixel
Halftone: How To

1. Quantize each pixel with 10 levels:

<table>
<thead>
<tr>
<th>Interval</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,25.5]</td>
<td>1</td>
</tr>
<tr>
<td>[25.5,51]</td>
<td>2</td>
</tr>
<tr>
<td>[51,76.5]</td>
<td>3</td>
</tr>
<tr>
<td>[76.5,102]</td>
<td>4</td>
</tr>
<tr>
<td>[102,127.5]</td>
<td>5</td>
</tr>
<tr>
<td>[127.5,153]</td>
<td>6</td>
</tr>
<tr>
<td>[153,178.5]</td>
<td>7</td>
</tr>
<tr>
<td>[178.5,204]</td>
<td>8</td>
</tr>
<tr>
<td>[204,229.5]</td>
<td>9</td>
</tr>
<tr>
<td>[229.5,255]</td>
<td>10</td>
</tr>
</tbody>
</table>
Halftone: How To

2. Replace each level by a mask

Bayesian Matrix:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>7</td>
</tr>
</tbody>
</table>
INVERSE HALFTONING
Inverse Halftoning

• From a 1-bit binary to a 8-bit image
## Inverse Halftone: How To

1. Choose randomly

<table>
<thead>
<tr>
<th>Level</th>
<th>Choose a random value in</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0, 25.5[</td>
</tr>
<tr>
<td>2</td>
<td>[25.5, 51[</td>
</tr>
<tr>
<td>3</td>
<td>[51, 76.5[</td>
</tr>
<tr>
<td>4</td>
<td>[76.5, 102[</td>
</tr>
<tr>
<td>5</td>
<td>[102, 127.5[</td>
</tr>
<tr>
<td>6</td>
<td>[127.5, 153[</td>
</tr>
<tr>
<td>7</td>
<td>[153, 178.5[</td>
</tr>
<tr>
<td>8</td>
<td>[178.5, 204[</td>
</tr>
<tr>
<td>9</td>
<td>[204, 229.5[</td>
</tr>
<tr>
<td>10</td>
<td>[229.5, 255]</td>
</tr>
</tbody>
</table>
Inverse Halftone: How To

This approach will give us:

![Image of a woman with a hat]

Fast Inverse Halftoning Algorithm for Ordered Dithered Images
Inverse Halftone: How To

• This approach has two problems
  – detail loss.
  – false contours.
Inverse Halftone: How To

• Avoiding restoration problems
  – Detail loss
    • pre-filter with unsharp filter.
    • Decode the binary image with a Gaussian low-pass.

  – False contours
    • Use Gaussian Random Walk Process to chose the gray levels.
    • Nearest neighborhood are used as weight values for random process.
Inverse Halftone: How To

Coding

Unsharp Mask → Quantize (10 levels) → Dot Pattern
Inverse Halftone: How To

Decoding

Dot Pattern → Dot-pattern to Level → Select a random value

Gaussian Low-pass → Select a new random value → Redefine Interval

Get the nearest neighborhood

All neighborhood are Equals?

No → Yes
Inverse Halftone: How To

* Avoiding restoration errors*
EXPERIMENTS
Experiments

- We compared three models:
  - Our model: Fast Inverse Halftone from Dispersed-dot Ordered Dithering (Stochastic)
  - The fastest model: Fast Blind Inverse Halftoning (FBIH)
  - The model with highest visual quality: Wavelet-based Inverse Halftoning via Deconvolution (WinHD)
SUBJECTIVE RESULTS
OBJECTIVE RESULTS
Objective Results

Elapsed time for reconstruction (in seconds).

<table>
<thead>
<tr>
<th>Image</th>
<th>Our Method</th>
<th>FBIH</th>
<th>WinHD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cameraman</td>
<td>1.79</td>
<td>2.46</td>
<td>35.14</td>
</tr>
<tr>
<td>Dollar</td>
<td>1.01</td>
<td>3.00</td>
<td>52.27</td>
</tr>
<tr>
<td>Einstein</td>
<td>1.53</td>
<td>3.00</td>
<td>53.36</td>
</tr>
<tr>
<td>Galaxy</td>
<td>1.44</td>
<td>3.01</td>
<td>65.78</td>
</tr>
<tr>
<td>Hurricane</td>
<td>1.24</td>
<td>2.99</td>
<td>75.48</td>
</tr>
<tr>
<td>Rose</td>
<td>1.24</td>
<td>2.02</td>
<td>73.13</td>
</tr>
<tr>
<td>Lena</td>
<td>1.26</td>
<td>2.40</td>
<td>36.29</td>
</tr>
</tbody>
</table>
Objective Results

Comparison results using Peak signal-to-noise ratio (PSNR)

<table>
<thead>
<tr>
<th>Image</th>
<th>FBIH</th>
<th>WinHD</th>
<th>Our</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cameraman</td>
<td>24.7800</td>
<td>31.1112</td>
<td>24.2410</td>
</tr>
<tr>
<td>Galaxy</td>
<td>33.6986</td>
<td>35.1189</td>
<td>25.6677</td>
</tr>
<tr>
<td>Peppers</td>
<td>27.4723</td>
<td>31.0780</td>
<td>24.8490</td>
</tr>
<tr>
<td>Chester cathedral</td>
<td>20.9075</td>
<td>23.5957</td>
<td>22.1560</td>
</tr>
<tr>
<td>Hurricane</td>
<td>26.8154</td>
<td>26.0245</td>
<td>25.4556</td>
</tr>
<tr>
<td>Pills</td>
<td>28.0535</td>
<td>31.3024</td>
<td>25.7641</td>
</tr>
<tr>
<td>Dollar</td>
<td>22.1636</td>
<td>21.4012</td>
<td>24.2334</td>
</tr>
<tr>
<td>Lena</td>
<td>29.5776</td>
<td>32.4471</td>
<td>25.3655</td>
</tr>
<tr>
<td>Bear</td>
<td>27.0785</td>
<td>30.5968</td>
<td>25.9224</td>
</tr>
</tbody>
</table>
## Objective Results

**Comparison results using Universal Image Quality Index (UIQI)**

<table>
<thead>
<tr>
<th>Image</th>
<th>FBIH</th>
<th>WinHD</th>
<th>Our</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cameraman</td>
<td>0.9413</td>
<td>0.9675</td>
<td>0.9978</td>
</tr>
<tr>
<td>Galaxy</td>
<td>0.9993</td>
<td>0.9992</td>
<td>0.9290</td>
</tr>
<tr>
<td>Peppers</td>
<td>1.0003</td>
<td>0.9945</td>
<td>0.9688</td>
</tr>
<tr>
<td>Chester cathedral</td>
<td>0.9987</td>
<td>1.0006</td>
<td>0.9915</td>
</tr>
<tr>
<td>Hurricane</td>
<td>0.9840</td>
<td>0.9822</td>
<td>0.8949</td>
</tr>
<tr>
<td>Pills</td>
<td>0.9976</td>
<td>1.0054</td>
<td>0.9742</td>
</tr>
<tr>
<td>Dollar</td>
<td>0.9973</td>
<td>0.9981</td>
<td>0.9914</td>
</tr>
<tr>
<td>Lena</td>
<td>0.9989</td>
<td>0.9999</td>
<td>0.9804</td>
</tr>
<tr>
<td>Bear</td>
<td>0.9158</td>
<td>0.9427</td>
<td>0.9092</td>
</tr>
</tbody>
</table>
## Objective Results

Comparison results using structural similarity (SSIM)

<table>
<thead>
<tr>
<th>Image</th>
<th>FBIH</th>
<th>WinHD</th>
<th>Our</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cameraman</td>
<td>0.7375</td>
<td>0.9139</td>
<td>0.8599</td>
</tr>
<tr>
<td>Galaxy</td>
<td>0.8640</td>
<td>0.9191</td>
<td>0.9076</td>
</tr>
<tr>
<td>Peppers</td>
<td>0.7894</td>
<td>0.8469</td>
<td>0.8590</td>
</tr>
<tr>
<td>Chester cathedral</td>
<td>0.6648</td>
<td>0.8337</td>
<td>0.8352</td>
</tr>
<tr>
<td>Hurricane</td>
<td>0.7451</td>
<td>0.6842</td>
<td>0.8529</td>
</tr>
<tr>
<td>Pills</td>
<td>0.8597</td>
<td>0.9157</td>
<td>0.9279</td>
</tr>
<tr>
<td>Dollar</td>
<td><strong>0.7352</strong></td>
<td><strong>0.6783</strong></td>
<td><strong>0.8921</strong></td>
</tr>
<tr>
<td>Lena</td>
<td>0.8318</td>
<td>0.8896</td>
<td>0.8698</td>
</tr>
<tr>
<td>Bear</td>
<td>0.8092</td>
<td>0.8807</td>
<td>0.8418</td>
</tr>
</tbody>
</table>
PROS & CONS
Pros & Cons

• Pros
  – Fastest
  – Low computational complexity
  – Good subjective results
  – Preserves details

• Cons
  – Low-frequency loss
  – Require two filtering stages
FUTURE WORK
Future Work

Self Embedding Watermarking Using Halftoning for Error Concealment

1. Watermarking Halftone
2. Lost Packets
   2. Send Frame
3. Extract self-embedding Halftone image
4. Use fast inverse halftone
5. Replace the lost blocks

Fast Inverse Halftoning Algorithm for Ordered Dithered Images
Thank You