Digital Halftoning

Prof. George Wolberg
Dept. of Computer Science
City College of New York
Objectives

In this lecture we review digital halftoning techniques to convert grayscale images to bitmaps:
- Unordered (random) dithering
- Ordered dithering
- Patterning
- Error diffusion
Background

• An 8-bit grayscale image allows 256 distinct gray levels.
• Such images can be displayed on a computer monitor if the hardware supports the required number of intensity levels.
• However, some output devices print or display images with much fewer gray levels.
• In these cases, the grayscale images must be converted to binary images, where pixels are only black (0) or white (255).
• Thresholding is a poor choice due to objectionable artifacts.
• Strategy: sprinkle black-and-white dots to simulate gray.
• Exploit spatial integration (averaging) performed by eye.
Thresholding

- The simplest way to convert from grayscale to binary.

Loss of information is unacceptable.
Unordered Dither (1)

• Reduce quantization error by adding uniformly distributed white noise (dither signal) to the input image prior to quantization.
• Dither hides objectional artifacts.
• To each pixel of the image, add a random number in the range \([-m, m]\), where \(m\) is MXGRAY/quantization-levels.
Unordered Dither (2)

Quantization

Dither/Quantization

1 bpp 2 bpp 3 bpp 4 bpp
Ordered Dithering

- Objective: expand the range of available intensities.
- Simulates n bpp images with m bpp, where n>m (usually m = 1).
- Exploit eye’s spatial integration.
  - Gray is due to average of black/white dot patterns.
  - Each dot is a circle of black ink whose area is proportional to \((1 – \text{intensity})\).
  - Graphics output devices approximate the variable circles of halftone reproductions.

- 2 x 2 pixel area of a bilevel display produces 5 intensity levels.
- \(n \times n\) group of bilevel pixels produces \(n^2+1\) intensity levels.
- Tradeoff: spatial vs. intensity resolution.
Dither Matrix (1)

• Consider the following 2x2 and 3x3 dither matrices:

\[
D^{(2)} = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix} \quad D^{(3)} = \begin{bmatrix} 6 & 8 & 4 \\ 1 & 0 & 3 \\ 5 & 2 & 7 \end{bmatrix}
\]

• To display a pixel of intensity \( I \), we turn on all pixels whose associated dither matrix values are less than \( I \).

• The recurrence relation given below generates larger dither matrices of dimension \( n \times n \), where \( n \) is a power of 2.

\[
D^{(n)} = \begin{bmatrix} 4D^{(n/2)} + D^{(2)}_{00}U^{(n/2)} & 4D^{(n/2)} + D^{(2)}_{01}U^{(n/2)} \\ 4D^{(n/2)} + D^{(2)}_{10}U^{(n/2)} & 4D^{(n/2)} + D^{(2)}_{11}U^{(n/2)} \end{bmatrix}
\]

where \( U^{(n)} \) is an \( n \times n \) matrix of 1’s.
Dither Matrix (2)

- Example: a 4x4 dither matrix can be derived from the 2x2 matrix.

\[
D^{(4)} = \begin{bmatrix}
0 & 8 & 2 & 10 \\
12 & 4 & 14 & 6 \\
3 & 11 & 1 & 9 \\
15 & 7 & 13 & 5
\end{bmatrix}
\]
Patterning

• Let the output image be larger than the input image.
• Quantize the input image to \([0 \ldots n^2]\) gray levels.
• Threshold each pixel against all entries in the dither matrix.
  - Each pixel forms a 4x4 block of black-and-white dots for a \(D^{(4)}\) matrix.
  - An \(n \times n\) input image becomes a \(4n \times 4n\) output image.
• Multiple display pixels per input pixel.
• The dither matrix \(D_{ij}^{(n)}\) is used as a spatially-varying threshold.
• Large input areas of constant value are displayed exactly as before.
Implementation

• Let the input and output images share the same size.
• First quantize the input image to $[0…n^2]$ gray levels.
• Compare the dither matrix with the input image.

```c
for(y=0; y<h; y++) // visit all input rows
    for(x=0; x<w; x++){ // visit all input cols
        i = x % n;  // dither matrix index
        j = y % n;  // dither matrix index

        // threshold pixel using dither value $D_{ij}^{(n)}$
        out[y*w+x] = (in[y*w+x] > $D_{ij}^{(n)}$)? 255 : 0;
    }
```
Examples

8 bpp (256 levels)  
1 bpp (D³)  
1 bpp (D⁴)  
1 bpp (D⁸)
Error Diffusion

- An error is made every time a gray value is assigned to be black or white at the output.
- Spread that error to its neighbors to compensate for over/undershoots in the output assignments
  - If input pixel 130 is mapped to white (255) then its excessive brightness (255-130) must be subtracted from neighbors to enforce a bias towards darker values to compensate for the excessive brightness.
- Like ordered dithering, error diffusion permits the output image to share the same dimension as the input image.
Floyd-Steinberg Algorithm

\[ f^*(x, y) = f(x, y) + \sum_i \sum_j w_{ij} e(x - i, y - j) \]

\[ g(x, y) = \begin{cases} 255 & \text{if } f^*(x, y) > MXGRAY / 2 \\ 0 & \text{otherwise} \end{cases} \]

\[ e(x, y) = f^*(x, y) - g(x, y) \]

\[ \sum_i \sum_j w_{ij} = 1 \]
Error Diffusion Weights

- Note that visual improvements are possible if left-to-right scanning among rows is replaced by serpentine scanning (zig-zag). That is, scan odd rows from left-to right, and scan even rows from right-to-left.
- Further improvements can be made by using larger neighborhoods.
- The sum of the weights should equal 1 to avoid emphasizing or suppressing the spread of errors.

Floyd-Steinberg: $\begin{array}{cccc}
\rightarrow & x & 7/16 \\
3/16 & 5/16 & 1/16 \\
\end{array}$

Jarvis-Judice-Ninke: $\begin{array}{cccc}
\rightarrow & x & 7/48 & 5/48 \\
\end{array}$

Stucki: $\begin{array}{cccc}
\rightarrow & x & 8/42 & 4/42 \\
2/42 & 4/42 & 8/42 & 4/42 \\
1/42 & 2/42 & 4/42 & 2/42 \\
\end{array}$
Examples (1)

Floyd-Steinberg  Jarvis-Judice-Ninke
Examples (2)

Floyd-Steinberg

Jarvis-Judice-Ninke
Examples (3)

Floyd-Steinberg

Jarvis-Judice-Ninke
Implementation

thr = MXGRAY /2;  // init threshold value
for(y=0; y<h; y++){
    for(x=0; x<w; x++) {  // visit all input cols
        *out = (*in < thr) ?  // threshold
            BLACK : WHITE; // note: use LUT!
        e = *in - *out;  // eval error
        in[1] += (e*7/16.);  // add error to E nbr
        in[w-1] += (e*3/16.);  // add error to SW nbr
        in[w] += (e*5/16.); // add error to S nbr
        in[w+1] += (e*1/16.); // add error to SE nbr
        in++;  // advance input ptr
        out++; // advance output ptr
    }
}

Wolberg: Image Processing Course Notes
Comments

- Two potential problems complicate implementation:
  - errors can be deposited beyond image border
  - errors may force pixel grayvalues outside the [0,255] range
Solutions to Border Problem (1)

• Perform *if* statement prior to every error deposit
  - Drawback: inefficient / slow

• Limit excursions of sliding weights to lie no closer than 1 pixel from image boundary (2 pixels for J-J-N weights).
  - Drawback: output will be smaller than input

• Pad image with extra rows and columns so that limited excursions will yield smaller image that conforms with original input dimensions. Padding serves as placeholder.
  - Drawback: excessive memory needs for intermediate image
Solutions to Border Problem (2)

• Use of padding is further undermined by fact that 16-bit precision (\texttt{short}) is needed to accommodate pixel values outside $[0, 255]$ range.

• A better solution is suggested by fact that only two rows are active while processing a single scanline in the Floyd-Steinberg algorithm (3 for JJN).

• Therefore, use a 2-row (or 3-row) circular buffer to handle the two (or three) current rows.

• The circular buffer will have the necessary padding and 16-bit precision.

• This significantly reduces memory requirements.
Circular Buffer
New Implementation

thr = MXGRAY / 2;    // init threshold value

for(y=0; y<h; y++){    // visit all input rows
    copyRowToCircBuffer(y+1);    // copy next row to circ buffer
    in1 = buf[(y+1) % 2] + 1;     // circ buffer ptr; skip over pad
    in2 = buf[(y+1) % 2] + 1;     // circ buffer ptr; skip over pad
    e = *in1 - *out;             // eval error
    in1[ 1] += (e*7/16.);        // add error to E nbr
    in2[ -1] += (e*3/16.);       // add error to SW nbr
    in2[ 0] += (e*5/16.);        // add error to S nbr
    in2[ 1] += (e*1/16.);        // add error to SE nbr
    *out = (*in1 < thr)? BLACK : WHITE;    // threshold
    in1++; in2++                 // advance circ buffer ptrs
    out++;                       // advance output ptr
}
**Floyd-Steinberg**

\[
\begin{array}{ccc}
200 & 90 & 100 \\
50 & 200 & 80 \\
\end{array}
\]

\[
\begin{array}{ccc}
? & ? & ? \\
? & ? & ? \\
\end{array}
\]

\[
\begin{array}{ccc}
x & 90 & 100 \\
x & 200 & 80 \\
\end{array}
\]

\[
\begin{array}{ccc}
255 & ? & ? \\
? & ? & ? \\
\end{array}
\]

\[
e = 200 - 255 = -55
\]

\[
\begin{array}{ccc}
x & 66 & 100 \\
33 & 197 & 80 \\
\end{array}
\]

\[
\begin{array}{ccc}
x & x & 128 \\
45 & 217 & 94 \\
\end{array}
\]

\[
e = 65 - 0 = 65
\]

\[
\begin{array}{ccc}
x & x & x \\
45 & 197 & 94 \\
\end{array}
\]

\[
\begin{array}{ccc}
255 & 0 & 255 \\
? & ? & ? \\
\end{array}
\]

\[
e = 128 - 255 = -127
\]

\[
\begin{array}{ccc}
x & x & x \\
45 & 173 & 54 \\
\end{array}
\]

\[
\begin{array}{ccc}
x & x & x \\
x & 173 & 54 \\
\end{array}
\]

\[
\begin{array}{ccc}
255 & 0 & 255 \\
0 & ? & ? \\
\end{array}
\]

\[
e = 45 - 0 = 45
\]

\[
\begin{array}{ccc}
x & x & x \\
x & 193 & 54 \\
\end{array}
\]

\[
\begin{array}{ccc}
x & x & x \\
x & x & 54 \\
\end{array}
\]

\[
\begin{array}{ccc}
255 & 0 & 255 \\
0 & 255 & ? \\
\end{array}
\]

\[
e = 193 - 255 = -62
\]

\[
\begin{array}{ccc}
x & x & x \\
x & x & 26 \\
\end{array}
\]

\[
\begin{array}{ccc}
255 & 0 & 255 \\
0 & 255 & 0 \\
\end{array}
\]

Wolberg: Image Processing Course Notes
Pintillism Art

• A new artform that leverages error diffusion to recreate images with tens of thousands of pins
• Image tonalities are reproduced by varying the pin density
  - Dark regions use a higher density of pins
  - Lighter regions use a lower pin density
• Pintillism is painting with pins
Pointillism

- Developed by Impressionist artists Seurat and Signac in the 1880’s
- Art consists of intricate placement of spots of color
- Exploits viewer’s ability to visually blend together color spots
- Pintillism is related to pointillism since spots are replaced with pins
Georges Seurat
Georges Seurat
Stippling

- Uses small dots for creating imagery
- Stippling is completed in black and white, while pointillism uses color
Stippling

Layers

1 2 3 4
Stippling
Allure of Pintillism

• Distills images into primitive dot patterns
• Challenges our brain to fuse them to perceive continuous tones
• The economy of dots is a refreshing counterpoint to images marked by hyper-resolution and color vibrancy
• Pintillism sits at the opposite end of the spectrum
  - Allows us to relish in its abstraction
  - Engages us to interact with the piece to explore meaning from multiple viewpoints and levels of resolution
• Extruding flat dots into their 3D counterparts is a 21st century twist that allows us to add another dimension to the classic art form of pointillism
Example

Input

Remove background
Example