What is Halftoning

Objective:
- Render an image with multiple levels of grey or color on a device with fewer tones.

Applications:
- Print a greyscale image on a bi-level device (laser printer).
- Display a true color image on a limited color display (LED billboard).

Thresholding

- Throw away the low-order bits of intensity information.

- If intensity > threshold_value:
  - Display full intensity at this channel.
- Otherwise:
  - Display zero intensity at this channel.

Tradeoff between Spatial & Tonal Resolutions

- In an m×n halftone cell, provide m×n+1 tone levels.
- Thresholding:
  - Full resolution, 2 levels.
  - 2×2 cell: 1/2 resolution, 5 levels.
  - 4×4 cell: 1/4 resolution, 17 levels.

Pixel Patterns

- The overall intensity of a given cell is determined by the number of pixels that are turned on.
- The locations of the on-pixels do not matter.
- The layouts of on-pixels under different intensities is referred as pixel pattern.
- The desired pattern is application dependent.
Requirements for Pixel Pattern I

• Minimize visual artifacts:
  • When multiple pixel pattern are displayed together, there should be as fewer noticeable artifacts as possible
  • Avoid spatial patterns

Requirements for Pixel Pattern II

• Growth sequence:
  • Minimize the differences in the patterns for successive intensity levels
  • Any pixel intensified for intensity level i is also intensified for all levels k=i
  • Avoid temporal flicking when intensity changes

Ordered Dither

• Basic idea:
  • Instead of deciding which pixels to turn on within each cell, use a threshold matrix to perform thresholding
  • Values in threshold matrix are:
    • Ordered & scaled by maximum image intensity
    • Not affected by intensities of pixels

Ordered Dither Algorithm

```c
// Input: Image A with value between 0~Imax
//         Dither matrix D of size n×n
// Output: Matrix B with value 0 & 1
for (int j=0 ; j<height ; j++)
  for (int i=0 ; i<width ; i++) {
    int p=i%n;
    int q=j%n;
    if ( A[i][j] > D[p][q] )
      B[i][j] = 1;
    else
      B[i][j] = 0;
  }
```

Dispersed-Dot Dither

• On-pixel are scattered evenly on screen
  • For device that handle individual dots well
  • Optimal pattern:
    • The number of dots are even in submatrix

Generate Dispersed-Dot Matrix

\[
D^{(2)} = \begin{pmatrix} 0 & 2 \\ 3 & 1 \end{pmatrix}
D^{(4)} = \begin{pmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{pmatrix}
U^{(n)} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}
D^{(2n)} = \begin{pmatrix} 4D^{(2)} \\ 4D^{(2)} + 2U^{(2)} \\ 4D^{(2)} + 3U^{(2)} \\ 4D^{(2)} + U^{(2)} \end{pmatrix}
\]
Clustered-Dot Dither

- On-pixels are adjacent to each other
- Increasing intensity -> increasing dot size
- For devices that are poor at accommodating isolated on-pixels
- Orientation of pattern:
  - Human eye is good at discerning horizontal and vertical lines
  - Avoid using 0° or 90° directions

Dither Matrix for Diagonal Direction

- To produce dots along diagonal direction, the dither matrix need to have a diamond shape
- For easy handling, square shaped matrix is used for generating 2 dots at the same time

Generate Clustered-Dot Matrix

\[
\begin{pmatrix}
2n^2 - 1 & D & D \\
D & 2n^2 - 1 & -D \\
-2n^2 + 1 & D & D \\
D & -2n^2 + 1 & 2n^2 - 1
\end{pmatrix}
\]

Error Diffusion

- Proposed by Floyd & Steinberg in 1975.
- Basic idea:
  - Measure the error at current pixel caused by thresholding
  - Distribute the error to the neighbors for compensation
  - The global error is minimized

Error Diffusion Algorithm

```c
for (int j=0 ; j<height ; j++)
  for (int i=0 ; i<width ; i++) {
    if ( A[i][j] < 0.5 )
      B[i][j] = 0;
    else
      B[i][j] = 1;
    error = A[i][j] - B[i][j];
    A[i+1][j] += α * error;
    A[i-1][j+1] += β * error;
    A[i][j+1] += γ * error;
    A[i+1][j+1] += δ * error;
  }
```

Result Comparison

4×4 dispersed
8×8 dispersed
6×6 clustered
Error diffusion