**type of techniques**

- simple pixel modification
- interpolation/extrapolation
- compositing
- convolution
- **dithering**
- warping
- morphing
- misc. effects

**random dither**

- add noise to camouflage quantization artifacts

**quantization algorithm**

- 8 bits per pixel per channel
- 1 bit per pixel per channel

**uniform quantization**

- 1 channel, 1 bit per pixel = 2 levels
- 1 channel, 2 bits per pixel = 4 levels
- n levels
  - output levels: evenly spaced
  - thresholds: midpoints
n level uniform quantization

- output levels:
  \[ L_n^i = \frac{i}{n-1}, \quad i = 0, \ldots, n-1 \]

- thresholds:
  \[ T_n^i = \frac{(L_n^i + L_n^{i+1})}{2} = \frac{(2i-1)}{2(n-1)}, \quad i = 1, \ldots, n-1 \]

- quantization function:
  \[ Q_n : [0,1] \rightarrow \{0, \frac{1}{n-1}, \frac{2}{n-1}, \ldots, 1\} \]
  \[ Q_n(v) = \left\lfloor \frac{v(n-1) + 0.5}{n-1} \right\rfloor \]

uniform quantization: n level

quantization error

- 8 bits per channel per pixel
- 1 bit per channel per pixel

dithering

- technique for camouflaging error
- algorithms
  - random
  - ordered
  - error diffusion

comparison

- original 8 bits/pixel/channel
- random 1 bit/pixel/channel
- ordered 1 bit/pixel/channel
- error-diffusion 1 bit/pixel/channel

random dither

- add noise to camouflage quantization artifacts

- 8 bits/pixel/channel
- 1 bit/pixel/channel
- 1 bit/pixel/channel dithered
**random dither**

- For each pixel in the input image, add random noise to pixel value.
- Uniformly quantize new value.

**random dither**

- Add noise to camouflage quantization artifacts.

---

**ordered dither**

- Add pseudo-random noise to camouflage quantization artifacts.

**ordered dither: intuition**

- Suppose all 2x2 neighborhoods were uniform.

<table>
<thead>
<tr>
<th>.2</th>
<th>.2</th>
<th>.6</th>
<th>.6</th>
<th>.3</th>
<th>.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>.2</td>
<td>.2</td>
<td>.6</td>
<td>.6</td>
<td>.3</td>
<td>.3</td>
</tr>
<tr>
<td>.4</td>
<td>.4</td>
<td>.9</td>
<td>.9</td>
<td>.2</td>
<td>.2</td>
</tr>
<tr>
<td>.4</td>
<td>.4</td>
<td>.9</td>
<td>.9</td>
<td>.2</td>
<td>.2</td>
</tr>
</tbody>
</table>

---

**quantized blocks**

- Every neighborhood is quantized in one of two ways.

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**ordered dither: intuition**

- Suppose all 2x2 neighborhoods were uniform.
- We could quantize neighborhoods together.

<table>
<thead>
<tr>
<th>.2</th>
<th>.2</th>
<th>.6</th>
<th>.6</th>
<th>.3</th>
<th>.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>.2</td>
<td>.2</td>
<td>.6</td>
<td>.6</td>
<td>.3</td>
<td>.3</td>
</tr>
<tr>
<td>.4</td>
<td>.4</td>
<td>.9</td>
<td>.9</td>
<td>.2</td>
<td>.2</td>
</tr>
<tr>
<td>.4</td>
<td>.4</td>
<td>.9</td>
<td>.9</td>
<td>.2</td>
<td>.2</td>
</tr>
</tbody>
</table>
neighborhood blocks

intuition

try to preserve average intensity over neighborhoods

more intuition

average intensity simulates 5 output levels

threshold-midpoints

quantization

.2 .2 .6 .6 .3 .3
.2 .2 .6 .6 .3 .3
.4 .4 .9 .9 .2 .2
.4 .4 .9 .9 .2 .2

0 0 1 0 0
1 0 1 1 1
0 1 1 1 0
1 0 1 1 1

0 1/8 1/4 3/8 1/2
5/8 .75 7/8 1
but we don’t have uniform neighborhoods...

or do we?

image coherence

ordered dither intuition
choose thresholds for each neighborhood position that preserve, as well as possible, average intensity over uniform neighborhoods

ordered dither: step 1
determine pixel location in neighborhood

ordered dither: step 2
quantize based on threshold for pixel location

what are thresholds?
quantize based on threshold for pixel location

intuitively
• output levels: evenly spaced
• thresholds: midpoints
simulated output levels

threshold-midpoints

thresholds

ordered dither recap: step 1

determine pixel location in neighborhood

generalizations

ordered dither: step 2

quantize based on threshold for pixel location

- simulate more output levels
- use more bits/pixel/channel

Location 1: 1/8 Location 2: 3/8 Location 3: 5/8 Location 4: 7/8
ordered dither (1 bit/pixel/channel)

- to simulate $M = m^2 + 1$ levels
  - assign pixel locations in $m \times m$ neighborhoods to $m^2$ classes
  - use $T_m$, as threshold to quantize pixel in $i^{th}$ class

we just did $m=2$, $M=5$

$m=4$, $M=17$

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>8</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>3</td>
<td>11</td>
</tr>
</tbody>
</table>

Bayer's ordered 4x4

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>8</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>3</td>
<td>11</td>
</tr>
</tbody>
</table>

Bayer's ordered 4x4 (recursive)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>8</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>3</td>
<td>11</td>
</tr>
</tbody>
</table>

Bayer's 2x2 ordered dither

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

have to be careful to avoid creating artifacts
generalizations
• simulate more output levels
• use more bits/pixel/channel

Uniform Quantization: $n$ bits=$2^n$ levels

Ordered Dither: $n$ bits=$2^n$ levels
- 0 bits per channel per pixel
- 8 bits per channel per pixel dithered

error-diffusion dither
- 8 bits per channel per pixel
- 1 bits per channel per pixel dithered

comparison
original ordered error-diffusion

error-diffusion dither intuition
quantize $I_{00}$ using uniform quantization
error-diffusion dither intuition

this introduces some error ... suppose $Q_{00}$ is too dark

$I_{00}$  $I_{01}$  $I_{02}$  $Q_{00}$
$I_{10}$  $I_{11}$  $I_{12}$

we can compensate by lightening the neighbors of $I_{00}$.

$I_{00}$  $I_{01}$  $I_{02}$  $Q_{00}$
$I_{10}$  $I_{11}$  $I_{12}$

now continue quantization on modified image

$I_{00}$  $I_{01}$  $I_{02}$  $Q_{00}$
$I_{10}$  $I_{11}$  $I_{12}$

quantize $I_{00}$ using uniform quantization

$I_{00}$  $I_{01}$  $I_{02}$  $Q_{00}$
$I_{10}$  $I_{11}$  $I_{12}$

now for the details

quantize $I_{00} + \alpha e_{00}$

$\alpha + \beta + \chi + \delta = 1$
error diffusion dither

\[ e_{01} = I_{01} + \alpha e_{00} - Q_{01} \]

error contributions by upper & left neighbors

floyd-steinberg

\[ \alpha = \frac{7}{16} \]
\[ \beta = \frac{3}{16} \]
\[ \chi = \frac{5}{16} \]
\[ \delta = \frac{1}{16} \]

floyd-steinberg: example

\[ \begin{array}{cc}
\frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & \frac{1}{2} \\
\end{array} \]
\[ \begin{array}{c}
0 \\
1 \\
\end{array} \]

types of techniques

- simple pixel modification
- interpolation/extrapolation
- compositing
- convolution
- dithering
- warping
- morphing
- misc. effects
## Forward Warp

**Diagram:**

- **Function:** $f$ maps points in the input image to the plane.

- **Equation:** $f(x, y) = (x', y')$

- **Input:** $(x, y)$

- **Output:** $(x', y')$

**Problem:**

- If $f$ is not bijective,
  1. $f^{-1}(i, j)$ may not be defined.
  2. $f^{-1}(i, j)$ may not be unique.

## Backward Warp

**Diagram:**

- **Function:** $f$ maps points in the output image to the plane.

- **Equation:** $f(x', y') = (i, j)$

- **Input:** $(i, j)$

- **Output:** $(x', y')$

**Problem:**

1. $f(i, j)$ may lie outside the input image area.
   - Solution: give image an infinite, black (or other default) border.

2. $f(i, j)$ may not lie on a sample of the input image.
   - Solution: resample input.
**re-sample:** estimate input image at arbitrary location

![Diagram of re-sample process]

**re-sample**
interpolate based on nearby samples
- nearest
- bilinear
- bicubic
- gaussian

**which way is up?**
what are the coordinates of the pixels surrounding \((x,y)\)?

![Diagram showing pixel coordinates]

**nearest**
- compute distance between \(x,y\) and the locations of the neighboring samples
- set value at \(x,y\) to the value of the closest neighbor

![Diagram of nearest neighbor interpolation]

**re-sample**
interpolate based on nearby samples
- nearest
- **bilinear**
- bicubic
- gaussian

![Diagram showing bilinear interpolation]

**which way is up?**
what are the coordinates of the pixels surrounding \((x,y)\)?

![Diagram showing pixel coordinates and coordinates of neighboring samples]
### Bilinear Interpolation

1. Interpolate to find values at \((x, y)\) and \((x, y+1)\)

### Re-sample

Interpolate based on nearby samples
- Nearest
- Bilinear
- Bicubic
- Gaussian

### Bicubic

1. Interpolate to find values at \((x, y+i)\)
2. Interpolate to find value at \((x, y)\)

### Bicubic: Lagrangian

- There is a unique cubic polynomial through any four distinct sample points.
lagrange cubic polynomial

\[ P(x) = \sum_{i=0,1,2,3} \prod_{j=0,1,2,3, j \neq i} \frac{(x-x_j)}{(x_i-x_j)} \]

exercise: what is the value of \( P(x_i) \) i=0,1,2,3

re-sample

interpolate based on nearby samples
- nearest
- bilinear
- bicubic
- gaussian

interpolate nearby samples using normalized gaussian weights

unnormalized weight at (i,j) in window is \( \exp\left[-\frac{(x-i)^2+(y-j)^2}{\sigma^2}\right] \)