digital image processing

types of techniques
- simple pixel modification
- interpolation/extrapolation
- compositing
- convolution
- dithering
- warping
- morphing
- misc. effects

simple pixel modification

convert to gray

convert to gray

apply function f to each pixel of input image

f(r,g,b) = .3r + .59g + .11b
simple pixel modification

apply function $f$ to each channel of each pixel of input image

threshold

if $v > t$ then $f(v) = 1$
else $f(v) = 0$

invert

$f(v) = 1 - v$

brighten/darken
**brighten/darken**

\[ f(v) = \alpha v \text{ for } \alpha \geq 0 \]
clamp to [0,1]

**types of techniques**
- simple pixel modification
- interpolation/extrapolation
- compositing
- convolution
- dithering
- warping
- morphing
- non-photo-realistic effects

**interpolation/extrapolation**

\[ v' = \alpha v + (1-\alpha)w \]

**interpolate/extrapolate image with**

**invert**

**interpolate/extrapolate image with**
**type of techniques**

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

**compositing**

![Image 1](image1.png)

generalization of interpolation/extrapolation in which $\alpha$ varies depending on pixel location

$$v_{i,j} = \alpha_{i,j} v + (1-\alpha_{i,j}) w$$

**compositing**

typically $\alpha \in [0,1]$ so the array of $\alpha$ values can be represented by a single channel image called a mask

![Image 2](image2.png)

**type of techniques**

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

$$\sum_{ij} w_{ij} v_{ij}$$

Kernel:

$$\begin{bmatrix} w_{11} & \ldots & w_{1n} \\ \\
\vdots & \ddots & \vdots \\ \\
w_{n1} & \ldots & w_{nn} \end{bmatrix}$$

$n$ odd

Boundaries?

Edge detect:

![Google before and after edge detection]

Edge detect kernel:

$$\begin{bmatrix} -1/8 & -1/8 & -1/8 \\ \\
-1/8 & 1 & -1/8 \\ \\
-1/8 & -1/8 & -1/8 \end{bmatrix}$$

Blur:

![Google before and after blurring]
anti-aliasing

nXn box blur

w = 1/n^2

why is it important that the sum of the weights is 1?

3x3 box blur

a kernel is separable if W_{ij} = w_i * w_j

is the box filter separable?

separability

box blur vs. triangle blur

3x3 triangle blur
separability

\[
\begin{array}{ccc}
1/4 & 1/16 & 1/8 & 1/16 \\
1/2 & 1/8 & 1/4 & 1/8 \\
1/4 & 1/16 & 1/8 & 1/16 \\
1/4 & 1/2 & 1/4 & \\
\end{array}
\]

triangle function

\[
\begin{array}{c}
\text{B} \\
\text{O} \\
\text{T} \\
\end{array}
\]

discrete triangle

\[
\begin{array}{c}
\text{B} \\
\text{O} \\
\text{T} \\
\end{array}
\]

normalized, discrete triangle

1. \( T = (n+1)/2 \) gives \( n \) non-zero samples
2. \( \sum_{j=-T}^{T} f(j) = 1 \) provided \( B = 2/(n+1) \)

triangle blur filter

\[
\begin{array}{c}
\text{ith row} \\
\text{jth column} \\
\end{array}
\]

triangle samples: \( w_1, w_2, \ldots, w_n \)

example: \( n=3 \)

\[
\begin{array}{c}
\text{B} \\
\text{O} \\
\text{T} \\
\end{array}
\]

\[
\begin{array}{c}
1/2 \\
\end{array}
\]

\[
\begin{array}{c}
1/4 \\
\end{array}
\]

\[
\begin{array}{c}
-2 \\
-1 \\
0 \\
1 \\
2 \\
\end{array}
\]
3x3 triangle blur filter

<table>
<thead>
<tr>
<th>1/4</th>
<th>1/16</th>
<th>1/8</th>
<th>1/16</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2</td>
<td>1/8</td>
<td>1/4</td>
<td>1/8</td>
</tr>
<tr>
<td>1/4</td>
<td>1/16</td>
<td>1/8</td>
<td>1/16</td>
</tr>
<tr>
<td></td>
<td>1/4</td>
<td>1/2</td>
<td>1/4</td>
</tr>
</tbody>
</table>

box, triangle and gaussian blurs

example: n=3, \( \sigma = 1 \)

\( f(x) = B e^{-x^2/\sigma^2} \)

\( \sigma \) is an input parameter that controls the width of peak

\[ B = \frac{1}{\sum f(j)} \]

boxed, triangle and gaussian blurs

3x3 gaussian blur, \( \sigma = 1 \)

\[
\begin{matrix}
0.212 & 0.576 & 0.212 \\
0.045 & 0.122 & 0.045 \\
0.122 & 0.332 & 0.122 \\
0.045 & 0.122 & 0.045 \\
0.212 & 0.576 & 0.212 \\
\end{matrix}
\]
type of techniques

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

N level uniform quantization

- evenly spaced centers

quantization

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

random dither

- add noise to camouflage quantization artifacts
ordered dither

8 bits per channel per pixel

1 bits per channel per pixel

1 bits per channel per pixel dithered

ordered dither: 2 level output

output intensity

input intensity

t_i,j

quantization threshold depends on pixel location

ordered dither

to simulate $M=m^2+1$ ($m \geq 1$) intensity levels:

- assign pixel locations to $m^2$ classes

- use $T_{in}$ as threshold to quantize pixel in $i^{th}$ class

ordered dither: $m=2$

apply $2 \times 2$ mask to classify pixel locations

ordered dither: example

Class 1: $T_{1}=1/8$

Class 2: $T_{2}=3/8$

Class 3: $T_{3}=5/8$

Class 4: $T_{4}=7/8$
ordered dither: another view

add $\frac{1}{2}$ - $T_i$, "noise" to pixel in class $i$ then quantize

ordered dither

To simulate $M=m^2+1$ intensity levels:

- assign pixel locations to $m^2$ classes
- use $T_i$ as threshold to quantize pixel in $i^{th}$ class

how? answer: carefully

bayer's 2x2 ordered dither

bayer's 4x4 ordered dither matrix

<table>
<thead>
<tr>
<th></th>
<th>15</th>
<th>8</th>
<th>14</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>12</th>
<th>2</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

error-diffusion dither

8 bits per channel per pixel
1 bits per channel per pixel
1 bits per channel per pixel dithered

error-diffusion dither

intuition
error diffusion dither

quantize $I_{00}$ using uniform quantization

\[
\begin{array}{ccc}
I_{00} & I_{01} & I_{02} \\
I_{10} & I_{11} & I_{12} \\
\end{array}
\]

error diffusion dither

distribute error $e_{00} = I_{00} - Q_{00}$ to neighbors not yet quantized

\[
\begin{array}{ccc}
I_{00} & I_{01} & I_{02} \\
I_{10} & I_{11} & I_{12} \\
\end{array}
\]

\[\alpha + \beta + \chi + \delta = 1\]

error diffusion dither

quantize $I_{01} + \alpha e_{00}$

\[
\begin{array}{ccc}
I_{01} & I_{02} \\
I_{10} & I_{11} & I_{12} \\
\end{array}
\]

error diffusion dither

distribute error: $e_{01} = I_{01} + \alpha e_{00} - Q_{01}$

\[
\begin{array}{ccc}
I_{01} & I_{02} \\
I_{10} & I_{11} & I_{12} \\
\end{array}
\]

error diffusion dither

error contributions by upper & left neighbors

error diffusion dither

order of quantization is important
floyd-steinberg

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

floyd-steinberg: example

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

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

forward warp

\( f \) maps points in input image to the plane

pixel at \((i,j)\) in output image is assigned the value at location \(f^{-1}(i,j)\) in input image

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

\[ (x,y) \rightarrow f \rightarrow (i,j) \]

- f maps points in output image to the plane

backward warp: problems

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

- nearest
- bilinear
- bicubic
- gaussian

which way is up?

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

what are the coordinates of the pixels surrounding (x,y)?

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

re-sample

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

bilinear interpolation

1. interpolate to find values at (x_0, y_0) and (x_0, y_0 + 1)
2. interpolate to find value at x_0, y_0

bilinear interpolation

1. interpolate to find values at (x_0, y_0) and (x_0, y_0 + 1)

re-sample

interpolate based on nearby samples
- nearest
- bilinear
- bicubic
- gaussian
1. interpolate to find values at \((x, y)\)

2. interpolate to find value at \((x, y)\)

Bicubic: Lagrangian

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

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

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

Resample

Interpolate based on nearby samples:
- nearest
- bilinear
- bicubic
- Gaussian

Gaussian

Interpolate nearby samples using normalized Gaussian weights.

Unnormalized weight at \((i,j)\) in window is

\[
\exp\left[-\frac{(x-i)^2 + (y-j)^2}{\sigma^2}\right]
\]
types of techniques

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

morphing how to

non-photo-realistic effects

• emboss
• cubism
• mosaic
• etc.

See photoshop, gimp, or our own ip for more examples.