type of techniques

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

convolution

\[ \sum w_{ij} v_{ij} \]

boundaries?

kernel

\[
\begin{array}{ccc}
  w_{11} & \ldots & w_{1n} \\
  \vdots & \ddots & \vdots \\
  w_{n1} & \ldots & w_{nn}
\end{array}
\]

n odd

edge detect

edge detect kernel

\[
\begin{pmatrix}
  -1 & -1 & -1 \\
  -1 & 8 & -1 \\
  -1 & -1 & -1
\end{pmatrix}
\]

need to clamp values to \([0, 1]\)
why blur?

3x3 box blur

\[
\begin{array}{ccc}
1/9 & 1/9 & 1/9 \\
1/9 & 1/9 & 1/9 \\
1/9 & 1/9 & 1/9 \\
\end{array}
\]

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

nXn box blur

\[
\begin{array}{ccc}
w & \cdots & w \\
\cdot & \cdot & \cdot \\
w & \cdots & w \\
\end{array}
\]

w = \frac{1}{n^2}

box blur vs. triangle blur

3x3 triangle blur

\[
\begin{array}{ccc}
1/16 & 1/8 & 1/16 \\
1/8 & 1/4 & 1/8 \\
1/16 & 1/8 & 1/16 \\
\end{array}
\]
how to compute nXn triangle blur?

- Compute 1D normalized triangle filter
- Use separability

separability

A kernel is separable if $W_{ij} = w_i w_j$

is the box filter separable?

n x n triangle blur kernel

- Compute sample values for 1D triangle function: $w_1, \ldots, w_{(n+1)/2}, \ldots, w_n$
- Normalize 1D values
- Compute kernel $W_{ij} = w_i w_j$

sampled triangle function

\[
S_i = \begin{cases} 
   i & \text{for } 0 \leq i \leq \frac{(n+1)}{2} \\
   (n+1) - i & \text{for } \frac{(n+1)}{2} < i \leq n+1
\end{cases}
\]

Normalized

\[
B = \sum_{i=0}^{\frac{(n+1)}{2}} S_i = \sum_{i=0}^{\frac{(n+1)}{2}} i \ast \sum_{i=1}^{\frac{(n+1)}{2}} \ldots \frac{n+1-i}{2} = \frac{(n+1)^2}{4}
\]

\[
w_i = \frac{S_i}{B}
\]
normalized sampled triangle

\[ w_i = \frac{4i}{(n+1)^2} \quad \text{for} \quad 0 < i \leq \frac{n+1}{2} \]
\[ w_i = \frac{4((n+1)-i)}{(n+1)^2} \quad \text{for} \quad \frac{n+1}{2} < i \leq n+1 \]

triangle blur filter

\[ w_i w_j \]

ith row

jth column

for row \( i = 1, \ldots, n \) and column \( j = 1, \ldots, n \)

example: \( n = 3 \)

\[ w_i = \frac{4i}{(n+1)^2} \quad \text{for} \quad 0 < i \leq \frac{n+1}{2} \]
\[ w_i = \frac{4((n+1)-i)}{(n+1)^2} \quad \text{for} \quad \frac{n+1}{2} < i \leq n+1 \]

3x3 triangle blur filter

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

box, triangle and gaussian blurs

gaussian function

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

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

\[ s_i = e^{-(i-(n+1)/2)^2/\sigma^2} \]
for \( i = 1, \ldots, n \)

normalized

\[ B = \sum_{i=1}^{n} e^{-(i-(n+1)/2)^2/\sigma^2} \]
is the normalizing constant

\[ w_i = \frac{s_i}{B} \]

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

\[
\begin{array}{ccc}
1 & 2 & 3 \\
.212 & .576 & .212 \\
.212 & .212 & .212 \\
\end{array}
\]

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

\[
\begin{array}{ccc}
.122 & .332 & .122 \\
.045 & .122 & .045 \\
.212 & .576 & .212 \\
\end{array}
\]

types of techniques

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

forward warp

\( f \)

\( (x, y) \)

\( (x', y') \)

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

<table>
<thead>
<tr>
<th>input image</th>
<th>$f^{-1}(i,j)$ output image</th>
</tr>
</thead>
<tbody>
<tr>
<td>pixel at $(i,j)$ in output image is assigned the value at location $f^{-1}(i,j)$ in input image</td>
<td></td>
</tr>
</tbody>
</table>

**forward warp: problems**

- 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**

<table>
<thead>
<tr>
<th>$(x,y)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f$ maps points in output image to the plane</td>
<td></td>
</tr>
</tbody>
</table>

**backward warp**

<table>
<thead>
<tr>
<th>$(x,y)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>pixel $(i,j)$ in output image is assigned value of input image at location $f(i,j)$</td>
<td></td>
</tr>
</tbody>
</table>

**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:** estimate input image at arbitrary location
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)?

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

bilinear interpolation
1. interpolate to find values at (x,y) and (x+1,y)

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

1. interpolate to find values at \( (x, y) \) and \( (x, y+1) \)
2. interpolate to find value at \( x, y \)

re-sample

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

bicubic

1. interpolate to find values at \( (x, y+j) \)

bicubic

1. interpolate to find values at \( (x, y+j) \)
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

\[
P(x) = \sum_{i=0,1,2,3} s_i \prod_{j=1,2,3,4} (x-x_j)/(x_i-x_j)
\]

exercise: what is the value of \( P(x) \) for \( i=0,1,2,3 \)
**re-sample**

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 
\[
\exp[-((x-i)^2+(y-j)^2)/\sigma^2]
\]

**types of techniques**

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

**dissolve**

- film/video technique to fade from one shot to another

**blending across time**

\[
\alpha(t)I_0 + (1-\alpha(t))I_1
\]

\[
\alpha(t)
\]

\[
0 \quad \quad 1
\]

\[
t_0 \quad \quad t_1
\]

**blending example**
morphing = warping + blending

morphing = warping + blending

morphing = warping + blending

morphing how to

specifying the warp

specifying the warp
each line moves in time

computing the warp between adjacent images

computing the warp between adjacent images - single line

warp - single line

warp - single line

warp - single line

consider some special cases
warp – single line

consider some special cases

warp – multiple lines

compute weight for each line pair based on distance to $p$ in destination

$w = \left( \frac{Lc}{a+d} \right)^b$

where $L$ is the length of the line segment, $d$ is the distance from $p$ to the line segment, $a$, $b$, and $c$ are parameters to control the effect

compute source for each pair of lines using one-line algorithm

compute displacement from $p$ to each source point
warp - multiple lines

compute weighted displacement from p in source

demo

do it yourself