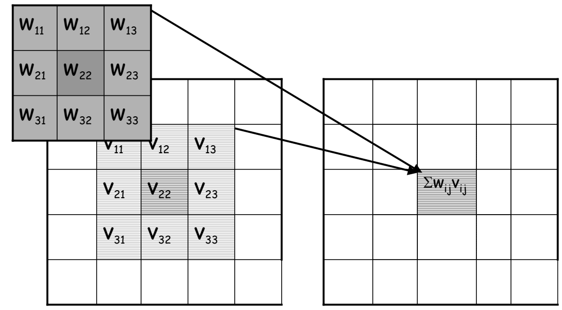
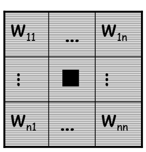
type of techniques

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

convolution

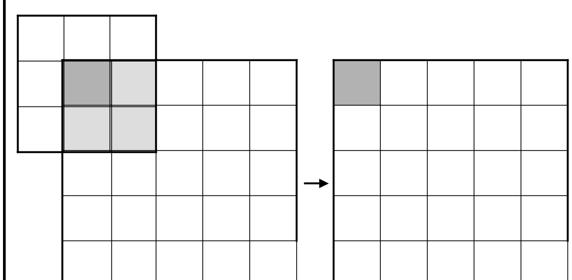


kernel

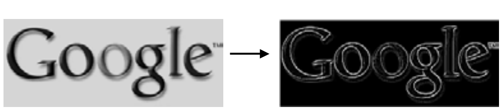


n odd

boundaries?



edge detect

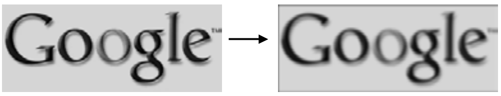


edge detect kernel

-1	-1	-1
-1	8	-1
-1	-1	-1

need to clamp values to [0,1]

blur



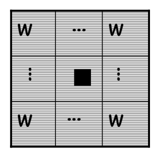
why blur?

anti-aliasing

3x3 box blur

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

nXn box blur



$w=1/n^2$

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

box blur vs. triangle blur

3x3 triangle blur

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

how to compute nXn triangle blur?

- Compute 1D normalized triangle filter
- Use separability

separability

	1/4	1/2	1/4
1/4	1/16	1/8	1/16
1/2	1/8	1/4	1/8
1/4	1/16	1/8	1/16

1D normalized triangle filter

separability

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

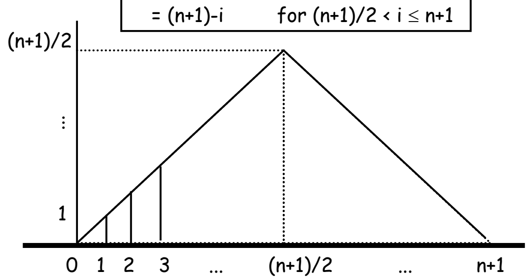
is the box filter separable?

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

$$S_i = i \quad \text{for } 0 < i \leq (n+1)/2$$
$$= (n+1)-i \quad \text{for } (n+1)/2 < i \leq n+1$$



normalized

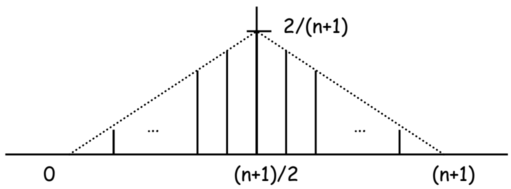
$$B = \Sigma_{i=0..n+1}\, s_i = \Sigma_{i=0\ldots(n+1)/2}\, i + \Sigma_{i=1+(n+1/2)\ldots n+1}\, n+1-i$$
$$= (n+1)^2/4$$

$w_i = s_i/B$

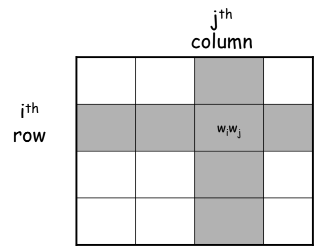
normalized sampled triangle

$w_i = 4i/(n+1)^2$ for $0 < i \le (n+1)/2$

$= 4[(n+1)-i]/(n+1)^2$ for $(n+1)/2 < i \le n+1$



triangle blur filter

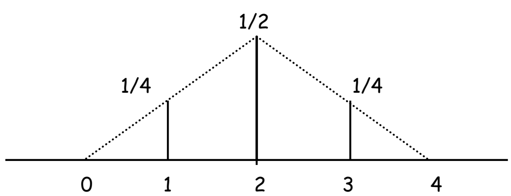


for row i=1, … , n and column j=1, … , n

example: n=3

$w_i = 4i/(n+1)^2$ for $0 < i \le (n+1)/2$

$= 4[(n+1)-i]/(n+1)^2$ for $(n+1)/2 < i \le n+1$



3x3 triangle blur filter

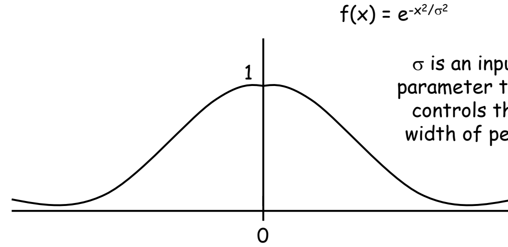
1/4	1/16	1/8	1/16
1/2	1/8	1/4	1/8
1/4	1/16	1/8	1/16
	1/4	1/2	1/4

box, triangle and guassian blurs

gaussian function

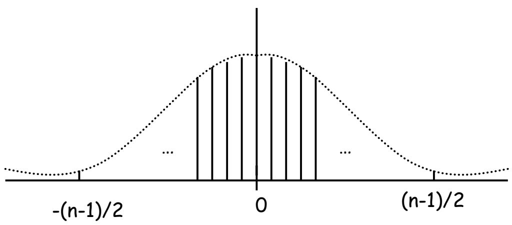
$f(x) = e^{-x^2/\sigma^2}$

σ is an input parameter that controls the width of peak



sampled

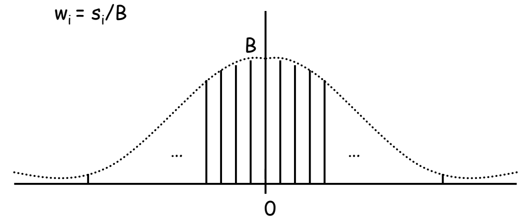
$s_i = e^{-(i-(n+1)/2)^2/\sigma^2}$
for i=1, …, n



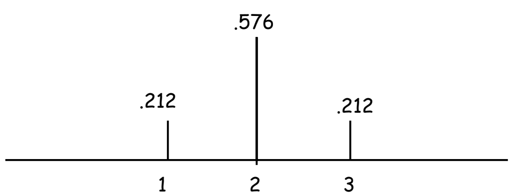
normalized

$B = \Sigma_{i=1,\dots,n}\, e^{-(i-(n+1)/2)^2/\sigma^2})$ is the normalizing constant

$w_i = s_i/B$



example: n=3, σ=1



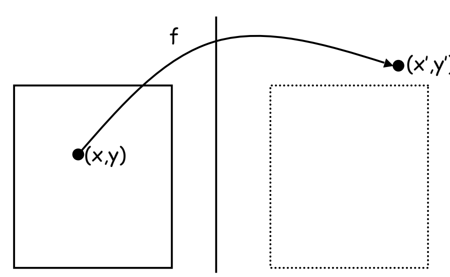
3x3 gaussian blur, σ = 1

.212	.045	.122	.045
.576	.122	.332	.122
.212	.045	.122	.045
	.212	.576	.212

types of techniques

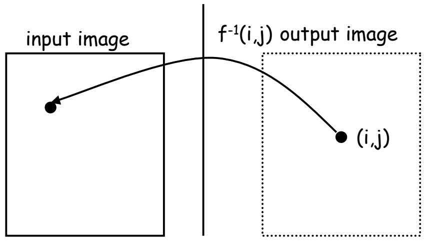
- simple pixel modification
- interpolation/extrapolation
- compositing
- convolution
- dithering
- **warping**
- morphing
- misc. effects

forward warp



f maps points in input image to the plane

forward warp



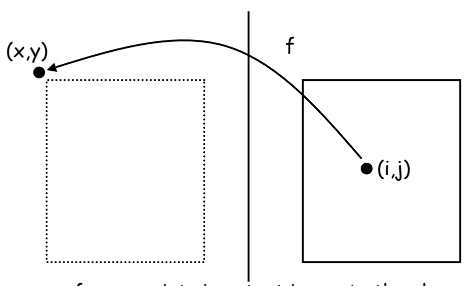
pixel at (i,j) in output image is assigned
the value at location $f^{-1}(i,j)$ in input image

forward warp: problems

if f is not bijective

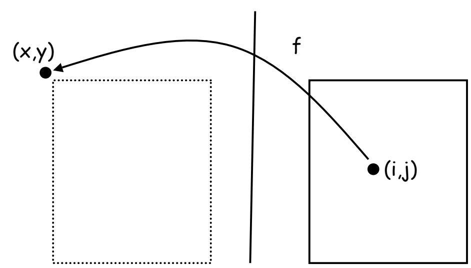
1. $f^{-1}(i,j)$ may not be defined
2. $f^{-1}(i,j)$ may not be unique

backward warp



f maps points in output image to the plane

backward warp

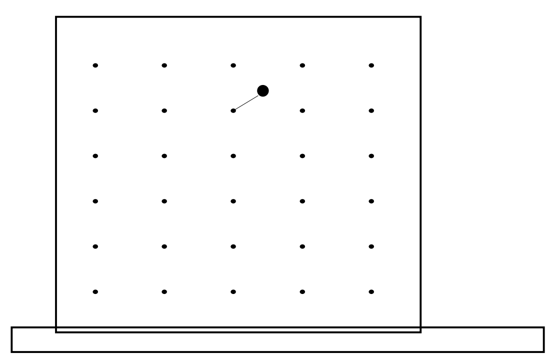


pixel (i,j) in output image is assigned value
of input image at location f(i,j)

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

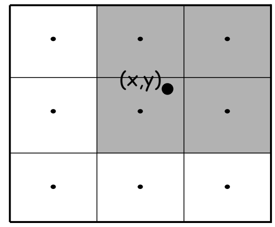
re-sample: estimate input image at arbitrary location



re-sample

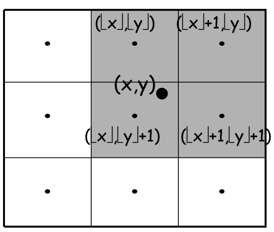
interpolate based on nearby samples
- nearest
- bilinear
- bicubic
- gaussian

which way is up?



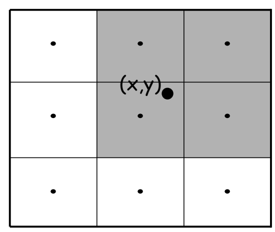
what are the coordinates of the pixels surrounding (x,y)?

which way is up?



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

nearest

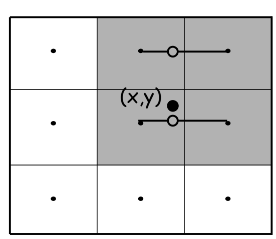


- compute distance between x,y and the locations of the neighboring samples
- set value at x,y to the value of the closest neighbor

re-sample

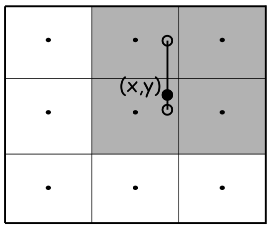
interpolate based on nearby samples
- nearest
- **bilinear**
- bicubic
- gaussian

bilinear interpolation



1. interpolate to find values at $(x,\lfloor y\rfloor)$ and $(x,\lfloor y\rfloor+1)$

bilinear interpolation

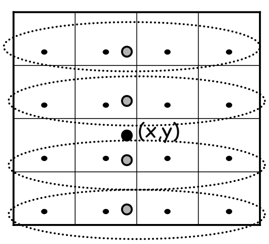


1. interpolate to find values at $(x,\lfloor y \rfloor)$ and $(x,\lfloor y \rfloor+1)$
2. interpolate to find value at x,y

re-sample

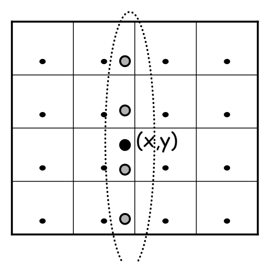
interpolate based on nearby samples
- nearest
- bilinear
- **bicubic**
- gaussian

bicubic



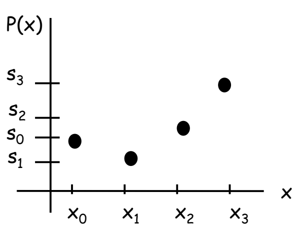
1. interpolate to find values at $(x,\lfloor y \rfloor+i)$

bicubic



1. interpolate to find values at $(x,\lfloor y \rfloor+i)$
2. interpolate to find value at (x,y)

bicubic: lagrangian



there is a unique cubic polynomial through any four distinct sample point

lagrange cubic polynomial

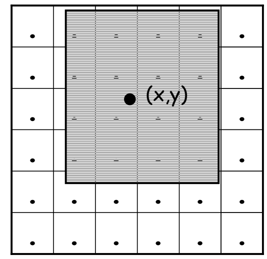
$P(x) = \Sigma_{i=0,1,2,3}\, s_i\, \Pi_{j=0,1,2,3,j\neq i} (x-x_j)/(x_i-x_j)$

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**

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

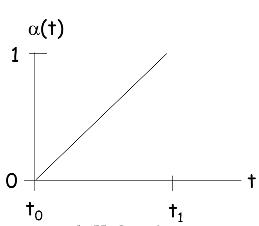
- simple pixel modification
- interpolation/extrapolation
- compositing
- convolution
- dithering
- warping
- **morphing**
- misc. effects

dissolve

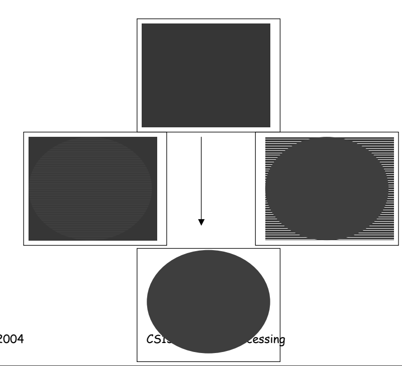
- film/video technique to fade from one shot to another

blending across time

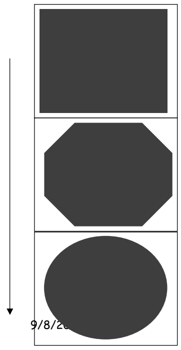
$$\alpha(t)I_0+(1-\alpha(t))I_1$$



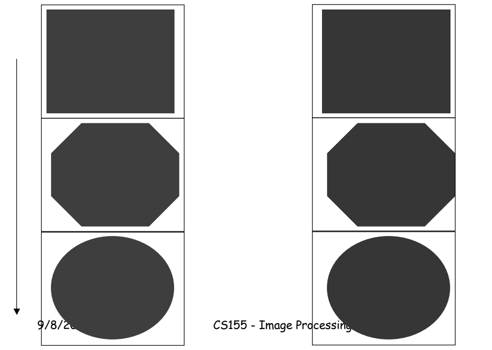
blending example



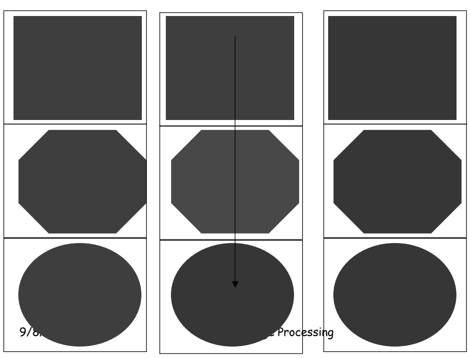
morphing = **warping** + blending



morphing = **warping** + blending



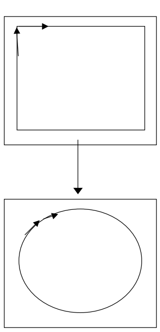
morphing = warping + **blending**



morphing how to

specifying the warp

start

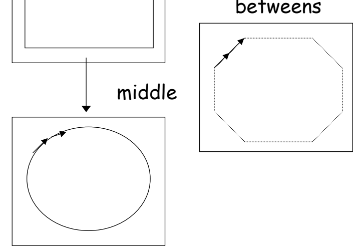


finish

specifying the warp

start

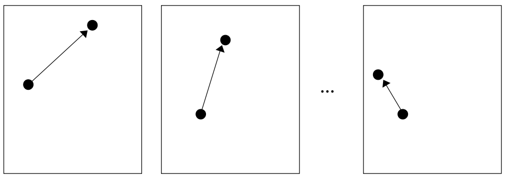
interpolate endpoints for in-betweens



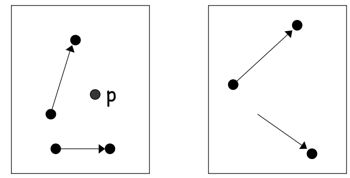
middle

finish

each line moves in time

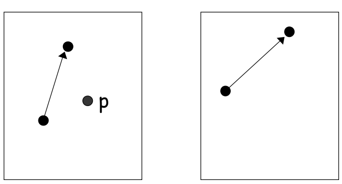


computing the warp between adjacent images



destination source

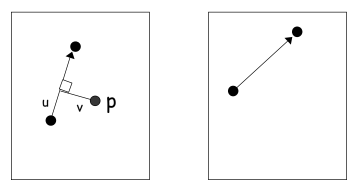
computing the warp between adjacent images-single line



destination source

warp – single line

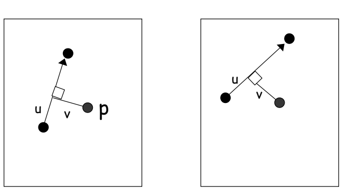
u is fraction along line, v is distance to line



destination source

warp – single line

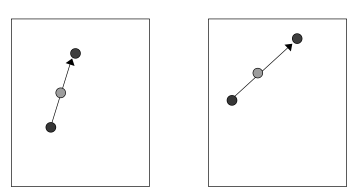
u is fraction along line, v is distance to line



destination source

warp – single line

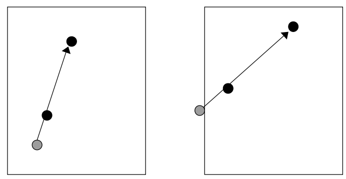
consider some special cases



destination source

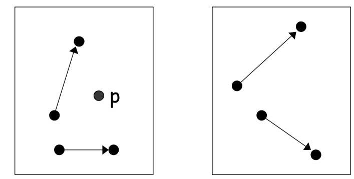
warp – single line

consider some special cases



destination source

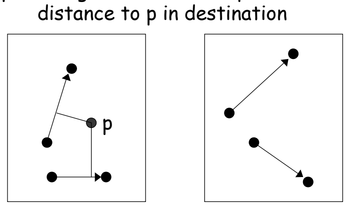
warp – multiple lines



destination source

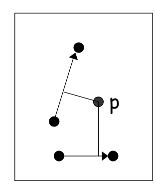
warp – multiple lines

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



destination source

weight



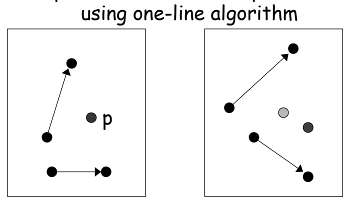
destination

$w = (L^c/(a+d))^b$

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

warp – multiple lines

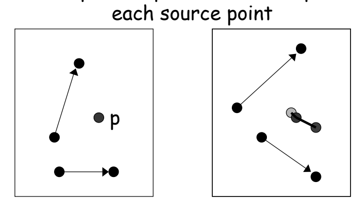
compute source for each pair of lines using one-line algorithm



destination source

warp – multiple lines

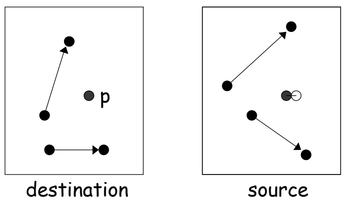
compute displacement from p to each source point



destination source

warp – multiple lines

compute weighted displacement from p in source



demo

do it yourself