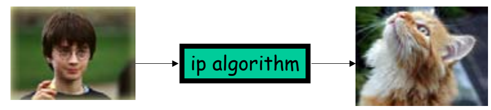
## cs155 - z sweedyk

digital image processing



## digital image processing






## digital image processing

- avoid/correct errors
- restore
- enhance
- analyze
- create



## types of techniques

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



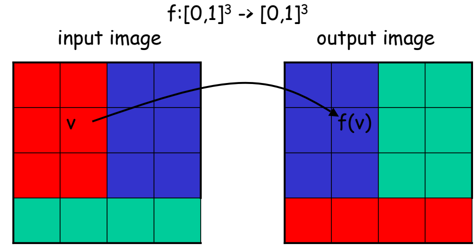
## types of techniques

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



## simple pixel modification

$f:[0,1]^3 \rightarrow [0,1]^3$




apply function **f** to each pixel of input image

## simple pixel modification

- convert to gray
- threshold
- invert
- brighten/darken



## convert to gray

 → 



## convert to gray

$$f(r,g,b) = c_r \bullet r + c_g \bullet g + c_b \bullet b$$

$$c_r = .2126, \; c_g = .7152, \; c_b = .0722$$



## Implementation Details

Our image processor will read uncompressed BMP files that contain

- 8 bit images: i.e. 1 channel, 8 bit per pixel per channel (grayscale)
- 24 bit images: i.e. 3 channel, 8 bit per pixel per channel (RGB)



## Implementation Details

Our image class handles 1 and 3 channel images with 1 to 8 bits per pixel per channel.

Your image processing routines should handle all of these cases.



## convert to gray

 → 

| 3 channel | 1 channel |
|---|---|
| 8 bit per pixel per channel | 8 bit per pixel per channel |

## convert to gray pseudo code

// I is 3 channel, 8 bits per pixel per channel image

Image* convert_to_gray(image I)

- create new one channel image I' that has the same dimensions and bits per pixel per channel as I
- for each pixel (i,j)
  - get (r,g,b) values of pixel (i,j) in I
  - set channel 0 of I' to $c_r \bullet r + c_g \bullet g + c_b \bullet b$
- return ptr to I'



## convert to gray

→

| 3 channel | 1 channel |
|---|---|
| n bits per pixel per channel | n bit per pixel per channel |



## convert to gray pseudo code

// I is 3 channel, 8 bits per pixel per channel image

Image* convert_to_gray(image I)

- create new one channel image I' that has the same dimensions and bits per pixel per channel as I
- for each pixel (i,j)
  - get (r,g,b) values of pixel (i,j) in I
  - set channel 0 of I' to $c_r \bullet r + c_g \bullet g + c_b \bullet b$
- return ptr to I'



## color values

LAST TIME

n bits per pixel per channel

1. $0, 1, 2, \ldots, 2^n - 1$
2. $0, 1/(2^n-1), 2/(2^n-1), \ldots, 1$

we use this convention



## convert to gray pseudo code

// I is 3 channel, 8 bits per pixel per channel image

Image* convert_to_gray(image I)

- create new one channel image I' that has the same dimensions and bits per pixel per channel as I
- for each pixel (i,j)
  - get (r,g,b) values of pixel (i,j) in I
  - set channel 0 of I' to $c_r \bullet r + c_g \bullet g + c_b \bullet b$
- return ptr to I'

Image class chooses closest admissible value!



## convert to gray pseudo code

// I is 3 channel, **n** bits per pixel per channel image

Image* convert_to_gray(image I)

- create new one channel image I' that has the same dimensions and bits per pixel per channel as I
- for each pixel (i,j)
  - get (r,g,b) values of pixel (i,j) in I
  - set channel 0 of I' to $c_r \bullet r + c_g \bullet g + c_b \bullet b$
- return ptr to I'

## convert to gray

→ ?

1 channel — n bits per pixel per channel

1 channel — n bit per pixel per channel



## convert to gray pseudo code

Image* convert_to_gray(image I)

- if I is a one channel image return NULL
- create new one channel image I' that has the same dimensions and bits per pixel per channel as I
- for each pixel (i,j)
  - get (r,g,b) values of pixel (i,j) in I
  - set channel 0 of I' to $c_r \bullet r + c_g \bullet g + c_b \bullet b$
- return ptr to I'



From here on I'll describe the algorithms for an input that is 3 channel, 8 bits per pixel per channel UNLESS the generalizations are not obvious (to me).



## threshold

→

3 channel — 8 bits per pixel per channel

3 channel — 1 bit per pixel per channel



## threshold

for a given threshold T:

$f_T(r,g,b)=(h_T(r),h_T(g),h_T(b))$

where

$h_T(v)=1$ if $v>T$

$h_T(v)=0$ otherwise



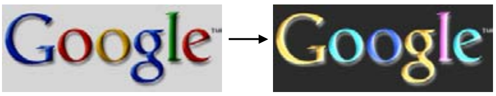
## threshold (1 channel)

→

1 channel — 8 bits per pixel per channel

1 channel — 1 bit per pixel per channel

## invert



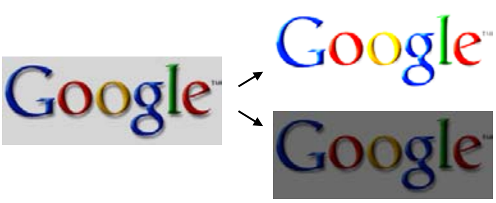


## invert (3 channel)

f(r,g,b) = (1-r,1-g,1-b)



## brighten/darken





## brighten/darken

for brighten/darken factor $\alpha>0$:

$f_{\alpha}(r,g,b) = (\alpha r, \alpha g, \alpha b)$

clamp to [0,1]

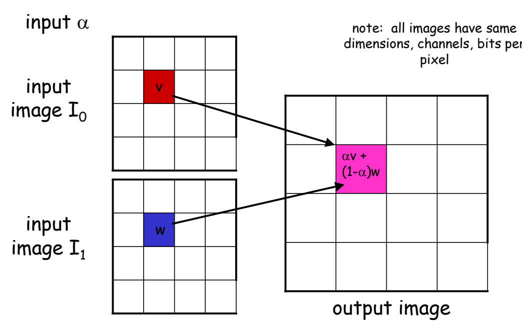


## types of techniques

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



## interpolation/extrapolation



## interpolation ($\alpha$ in [0,1])

input image $I_0$

input image $I_1$

output image= $\alpha I_0 + (1-\alpha)I_1$

interpolation blends $I_0$ and $I_1$



## extrapolation ($\alpha$ > 1)

input image $I_0$

input image $I_1$

output image= $\alpha I_0 + (1-\alpha)I_1$

extrapolation (with $\alpha$>1) creates an image that moves $I_0$ away from $I_1$

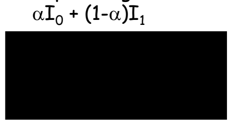


## extrapolation ($\alpha$ <0 )

input image $I_0$

input image $I_1$

output image= $\alpha I_0 + (1-\alpha)I_1$



extrapolation (with $\alpha$<0) creates an image that moves $I_1$ away from $I_0$



## interpolation/extrapolation



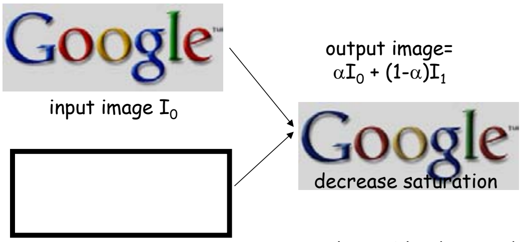
## brighten/darken

interpolate/extrapolate image with black image



## change saturation

## interpolation ($\alpha$ in [0,1])



output image= $\alpha I_0 + (1-\alpha) I_1$

input image $I_0$

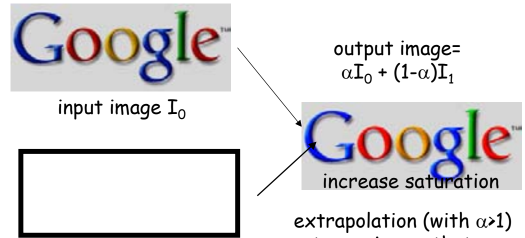
decrease saturation

input image $I_1$

interpolation blends $I_0$ and $I_1$



## extrapolation ($\alpha$ > 1)



output image= $\alpha I_0 + (1-\alpha) I_1$

input image $I_0$

increase saturation

extrapolation (with $\alpha$>1) creates an image that moves $I_0$ away from $I_1$

input image $I_1$



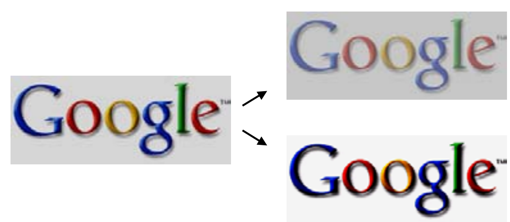
## change saturation

interpolate/extrapolate image with

________________________________

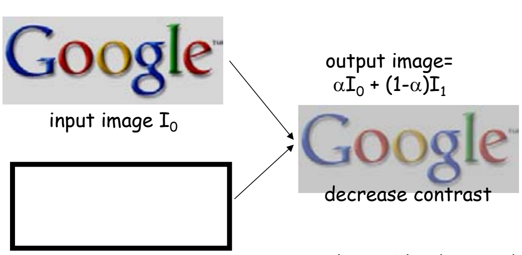


## change contrast





## interpolation ($\alpha$ in [0,1])



output image= $\alpha I_0 + (1-\alpha) I_1$

input image $I_0$

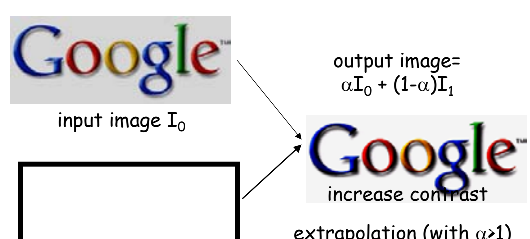
decrease contrast

input image $I_1$

interpolation blends $I_0$ and $I_1$



## extrapolation ($\alpha$ > 1)



output image= $\alpha I_0 + (1-\alpha) I_1$

input image $I_0$

increase contrast

extrapolation (with $\alpha$>1) creates an image that moves $I_0$ away from $I_1$

input image $I_1$

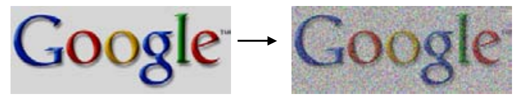
## change contrast

interpolate/extrapolate image with

______________________________



## noisify






## noisify

interpolate/extrapolate image with

______________________________



## invert




## invert

interpolate/extrapolate image with

______________________________



## a little computation

- invert: $f(v)=1-v$

## a little computation

- invert: f(v)=1-v
- interpolate/extrapolate:
  $f(v) = (1-\alpha)\, v + \alpha\, x$

What should $\alpha$ and x be?



## invert

interpolate/extrapolate with

______________________



## type of techniques

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



## compositing



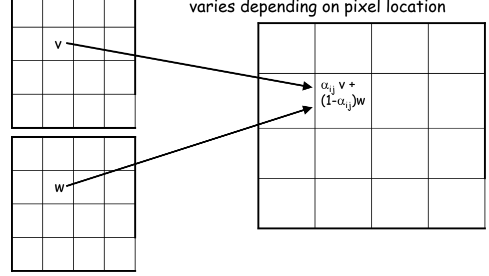



## compositing



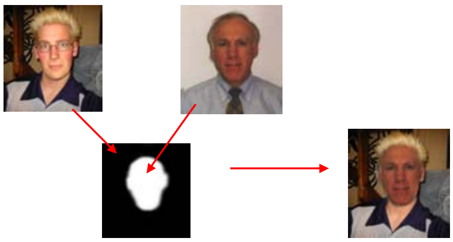

## compositing

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



## compositing

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





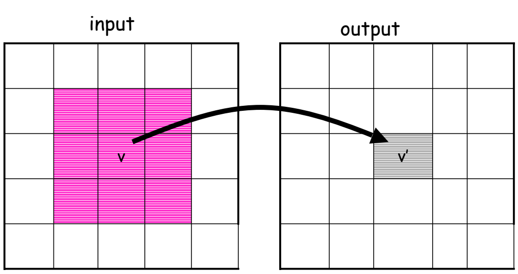
## type of techniques

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



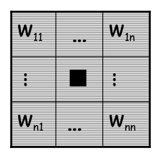
## convolution

v' is a weighted sum of v and its neighbors





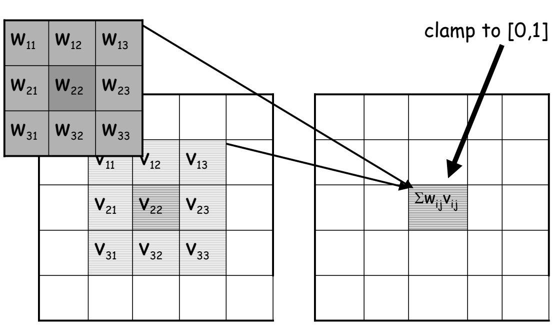

## kernel gives weights




n odd

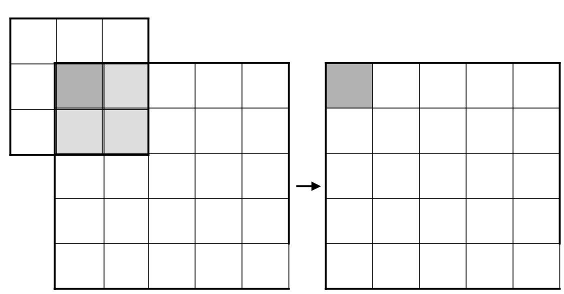


## convolution






## boundaries?



## ____________ kernel

| | | |
|---|---|---|
| -1 | -1 | -1 |
| -1 | 8 | -1 |
| -1 | -1 | -1 |

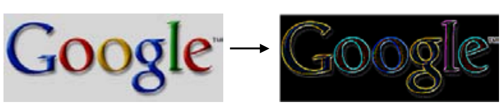


## edge detect kernel

| | | |
|---|---|---|
| -1 | -1 | -1 |
| -1 | 8 | -1 |
| -1 | -1 | -1 |



## edge detect






## _________kernel

| | | |
|---|---|---|
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |



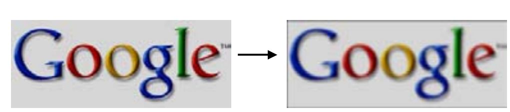
## blur kernel

| | | |
|---|---|---|
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |

3x3 blox blur kernel



## blur

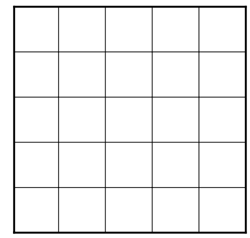



why blur?

## jaggies

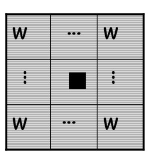


## 5x5 box blur





## nXn box blur

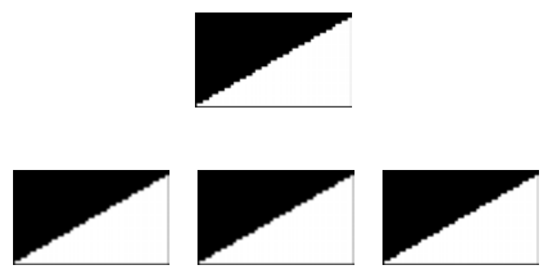



$w=1/n^2$

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



## box, triangle and guassian blurs



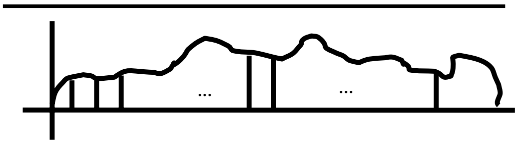


how to construct the kernels for 2D triangle and gaussian blurs?

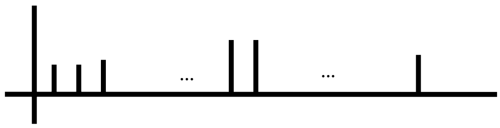
we'll start with 1D blurs.



## 1D convolution

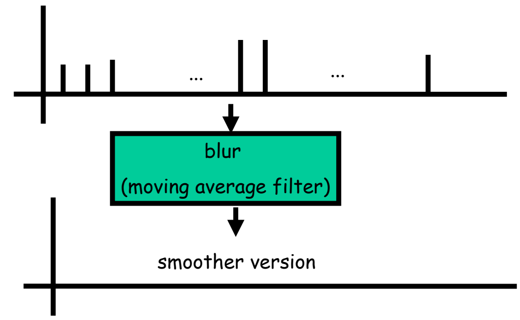


## 1D convolution



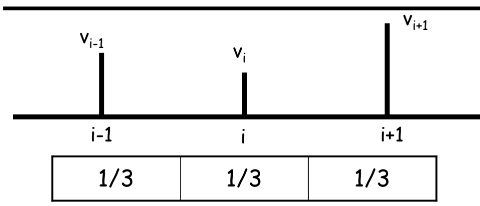


## 1D convolution



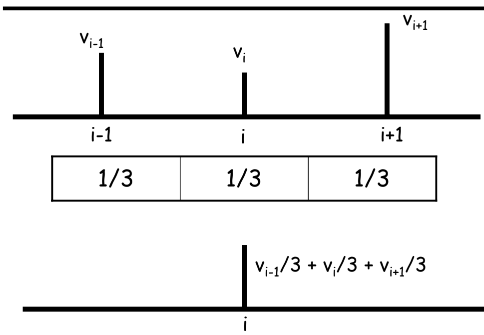



## 1D box blur, n=3



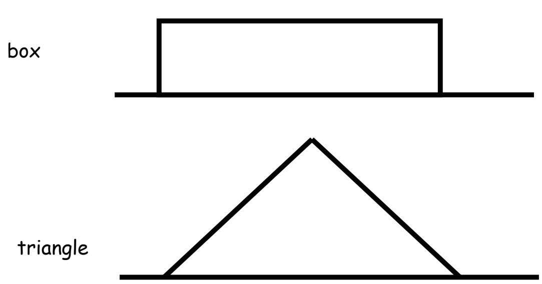



## 1D box blur





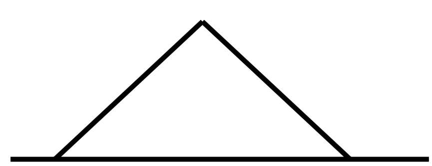

## weights for 1D blurs





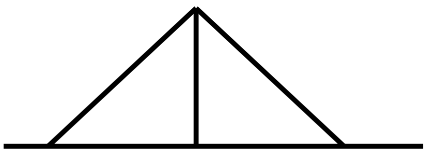

## 1D triangle blur weights for n=3

where should we sample?



## 1D triangle blur weights for n=3

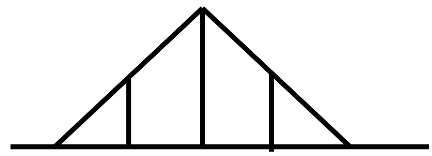
where should we sample?





## 1D triangle blur weights for n=3

where should we sample?

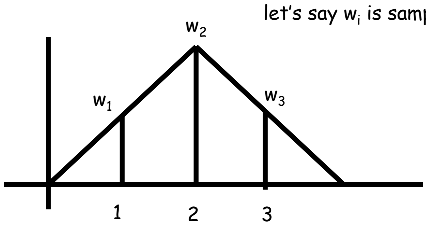




## 1D triangle blur weights for n=3

for convenience

let's say $w_i$ is sampled at i.

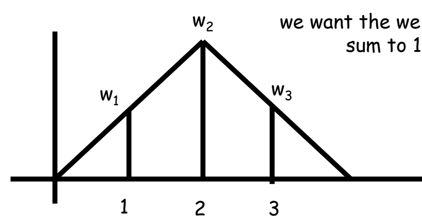





## 1D triangle blur weights for n=3

how high is the peak?

we want the weights to sum to 1!



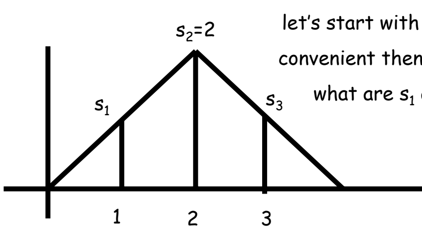



## 1D triangle blur weights for n=3

how high is peak?

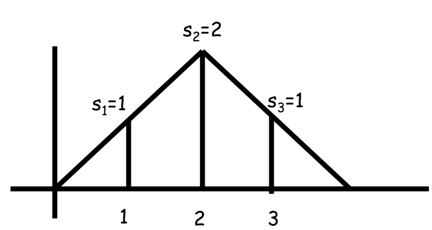
let's start with something convenient then normalize.

what are $s_1$ and $s_3$?

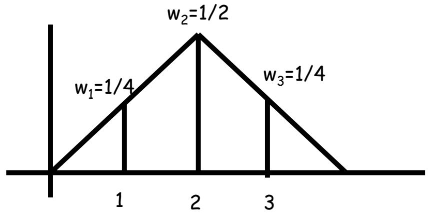





## 1D triangle blur weights for n=3



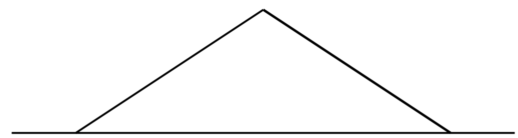
## 1D triangle blur weights for n=3






## General 1D triangle blur

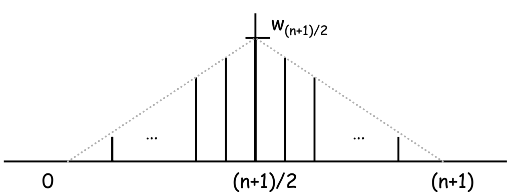
where do take the n samples?





## General 1D triangle blur

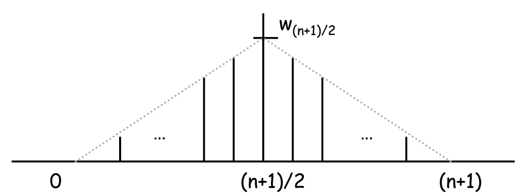
sample $w_i$ at i=1, 2, ... , n






## General 1D triangle blur

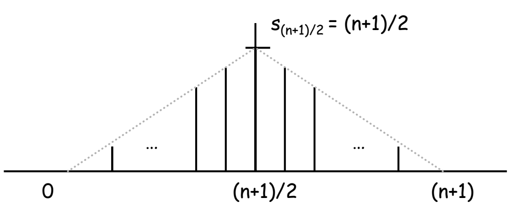
what is height of peak?






## General 1D triangle blur

Let $s_i$ = i for i=1, ..., (n+1)/2
= (n+1)-i for i= (n+3)/2 ... n






## General 1D triangle blur

Let $s_i = i \quad$ for $1 \le i \le (n+1)/2$
$= (n+1)-i \quad$ for $(n+1)/2 < i \le n$

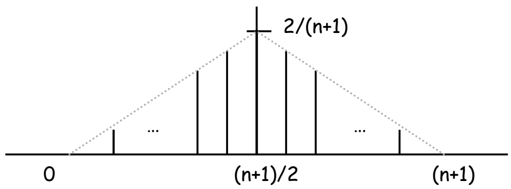
Let $B=\Sigma_{i=1,\ldots,n}\, s_i$

Let $w_i=s_i/B$

## 1D triangle blur

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

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






## 1D → 2D

How about applying the 1D filter to each row, then to each column!!



## box blur - rows

| | | |
|---|---|---|
| $v_{00}$ | $v_{01}$ | $v_{02}$ |
| $v_{10}$ | $v_{11}$ | $v_{12}$ |
| $v_{20}$ | $v_{21}$ | $v_{22}$ |

→

| | | |
|---|---|---|
| ? | $(v_{00}+v_{01}+v_{02})/3$ | ? |
| ? | $(v_{10}+v_{11}+v_{12})/3$ | ? |
| ? | $(v_{20}+v_{21}+v_{22})/3$ | ? |



## box blur - column

| | | |
|---|---|---|
| ? | $(v_{00}+v_{01}+v_{02})/3$ | ? |
| ? | $(v_{10}+v_{11}+v_{12})/3$ | ? |
| ? | $(v_{20}+v_{21}+v_{22})/3$ | ? |

→

| | | |
|---|---|---|
| ? | ? | ? |
| ? | $\Sigma_{i=0,2,j=0..2} V_{ij}/9$ | ? |
| ? | ? | ? |



## 3x3 box blur

| |
|---|
| 1/3 |
| 1/3 |
| 1/3 |

| | | |
|---|---|---|
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |

| | | |
|---|---|---|
| 1/3 | 1/3 | 1/3 |



## separability

| |
|---|
| $w_1$ |
| $w_2$ |
| $w_3$ |

| | | |
|---|---|---|
| $w_1^2$ | $w_1w_2$ | $w_1w_3$ |
| $w_2w_1$ | $w_2^2$ | $w_2w_3$ |
| $w_3w_1$ | $w_3w_2$ | $w_3^2$ |

| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |

## 3x3 triangle blur

VOILA!

| |
|---|
| 1/4 |
| 1/2 |
| 1/4 |

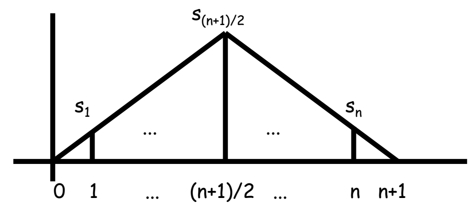
| | | |
|---|---|---|
| 1/16 | 1/8 | 1/16 |
| 1/8 | 1/4 | 1/8 |
| 1/16 | 1/8 | 1/16 |

| | | |
|---|---|---|
| 1/4 | 1/2 | 1/4 |



## Summary: nxn triangle blur

- Compute 1D triangle blur for n (n is odd)
  - Compute $s_i$ for i-1, …, n

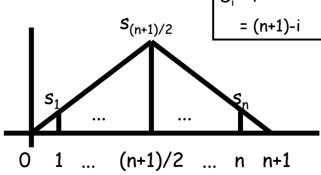





## nxn triangle blur

- Compute 1D triangle blur for n (n is odd)
  - Compute $s_i$ for i-1, …, n

| | |
|---|---|
| $S_i = i$ | for $0 < i \le (n+1)/2$ |
| $= (n+1)-i$ | for $(n+1)/2 < i \le n+1$ |






## nxn triangle blur

- Compute 1D triangle blur for n (n is odd)
  - Compute $s_i$ for i-1, …, n
  - Compute normalizing factor $B=\Sigma s_i$

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



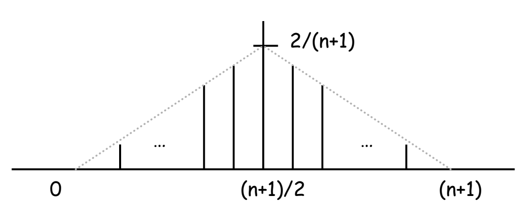
## nxn triangle blur

- Compute 1D triangle blur for n (n is odd)
  - Compute $s_i=i$ for i-1, …, n
  - Compute normalizing factor $B=\Sigma s_i$
  - Compute $w_i=s_i/B$



## 1D triangle blur

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



## separability

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \quad \begin{bmatrix} w_1^2 & w_1w_2 & w_1w_3 \\ w_2w_1 & w_2^2 & w_2w_3 \\ w_3w_1 & w_3w_2 & w_3^2 \end{bmatrix}$$

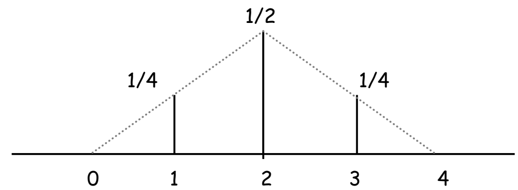
$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}$$



## example: n=3

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

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



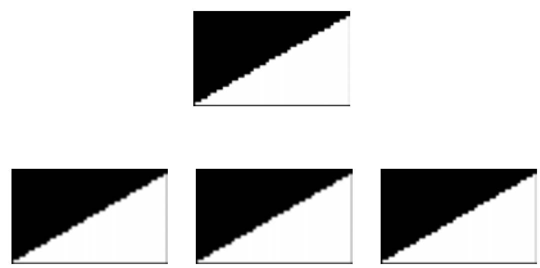



## 3x3 triangle blur filter

| | 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 |



## box, triangle and guassian blurs

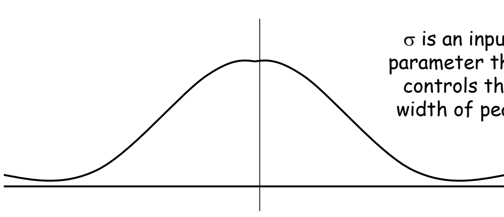




## gaussian function

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

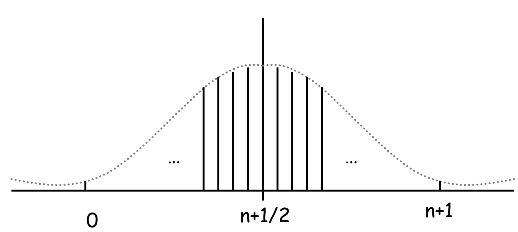
$\sigma$ is an input parameter that controls the width of peak






## sampled

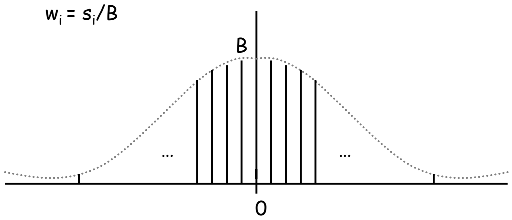
$s_i = e^{-(i-(n+1)/2)^2/\sigma^2}$
for i=1, …, n



## normalized

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

$w_i = s_i/B$






## 2D gaussian – use separability

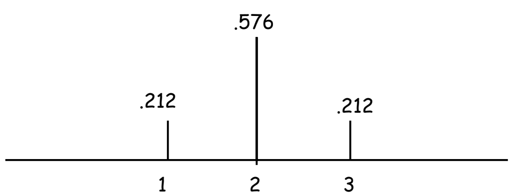
| |
|---|
| $w_1$ |
| $w_2$ |
| $w_3$ |

| | | |
|---|---|---|
| $w_1^2$ | $w_1w_2$ | $w_1w_3$ |
| $w_2w_1$ | $w_2^2$ | $w_2w_3$ |
| $w_3w_1$ | $w_3w_2$ | $w_3^2$ |

| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |



## example: n=3, $\sigma$=1






## 3x3 gaussian blur, $\sigma$ = 1

| | | | |
|---|---|---|---|
| .212 | .045 | .122 | .045 |
| .576 | .122 | .332 | .122 |
| .212 | .045 | .122 | .045 |
| | .212 | .576 | .212 |



## type of techniques

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