

Announcements

- **Class Website:**

Quercus: <https://q.utoronto.ca>

Last Year: <http://www.teach.cs.toronto.edu/~csc420h/fall/>

- Everything is / will be on Quercus! (assignments on MarkUs)

- **Tutorial:**

First Tutorial on Thursday, Sep 13, at 4pm
(Matlab, Colab, Matlab for Images)

- **Office Hours (confirmed):**

Mondays 4-5 and Fridays 11-12, room BA 2283

- **Waitlist ...**

Syllabus Overview

Image Processing

- Linear Filters
- Edge Detection
- Image Pyramids

Features & Matching

- Key Point Detection
- Scale Invariance
- Local Descriptors

Geometry

- Camera Models
- Stereo Vision

Recognition and Detection

- Object Recognition
- Neural Nets
- Deep Learning

- Shape Models
- HoG detectors
- Deformable Parts

- Segmentation

Images

Digital Image

- Image is a matrix with integer values
 - We will typically denote it with I



Digital Image

- Image is a matrix with integer values
 - We will typically denote it with I
 - $I(i,j)$ is called **intensity**



2

pixel (1, 1): intensity 255

Digital Image

- Image is a matrix with integer values
 - We will typically denote it with I
 - $I(i,j)$ is called **intensity**
 - Matrix I can be $m \times n$ (grayscale)



Digital Image

- Image is a matrix with integer values
 - We will typically denote it with I
 - $I(i,j)$ is called **intensity**
 - Matrix I can be $m \times n$ (grayscale)
 - or $m \times n \times 3$ (colour)



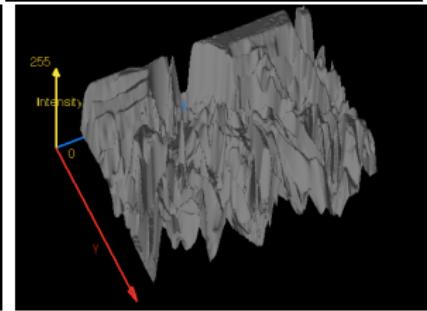
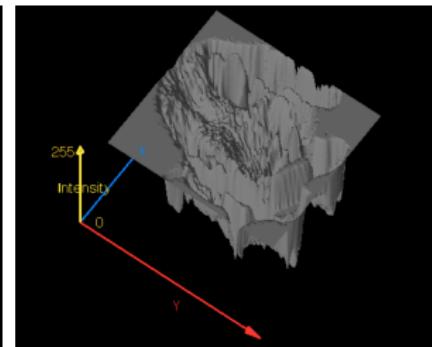
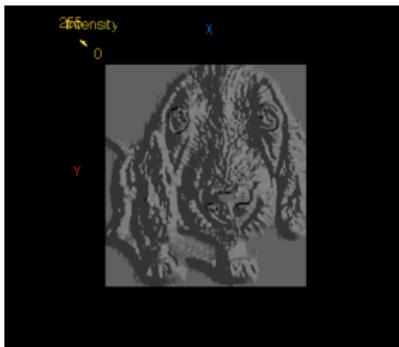
Digital Image

- Image is a matrix with integer values
 - We will typically denote it with I
 - $I(i,j)$ is called **intensity**
 - Matrix I can be $m \times n$ (grayscale)
 - or $m \times n \times 3$ (colour)



Babak Taati

Intensity



- We can think of a (grayscale) image as a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ giving the intensity at position (i, j)
- Intensity 0 is black and 255 is white

Image Transformations

- As with any function, we can apply operators to an image, e.g.:



$$I(i, j)$$

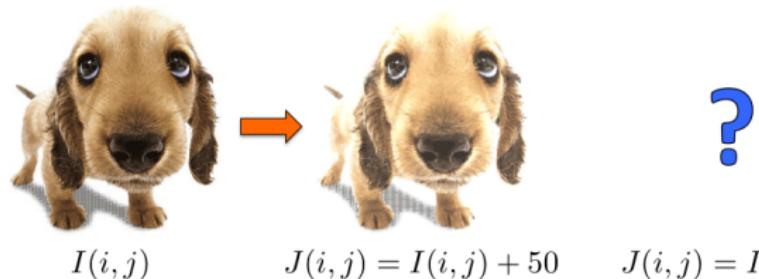
$$J(i, j) = I(i, j) + 50$$

- We'll talk about special kinds of operators, **correlation** and **convolution** (linear filtering)

[Adapted from: N. Snavely]

Image Transformations

- As with any function, we can apply operators to an image, e.g.:

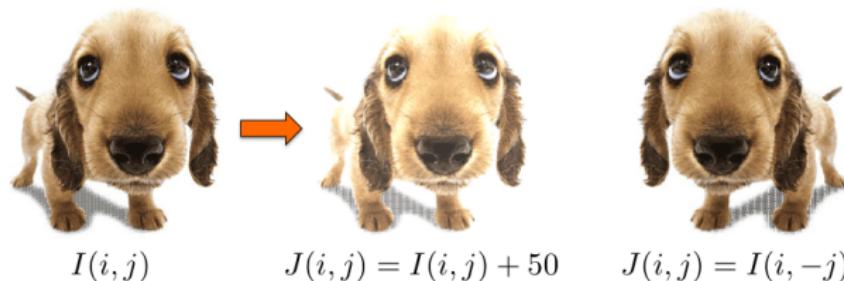


- We'll talk about special kinds of operators, **correlation** and **convolution** (linear filtering)

[Adapted from: N. Snavely]

Image Transformations

- As with any function, we can apply operators to an image, e.g.:



- We'll talk about special kinds of operators, **correlation** and **convolution** (linear filtering)

[Adapted from: N. Snavely]

Linear Filters

Reading: Szeliski book, Chapter 3.2

Motivation: Finding Waldo

- How can we find Waldo?



[Source: R. Urtasun]

Answer

- Slide and compare!
- In formal language: **filtering**

Motivation: Noise reduction

- Given a camera and a still scene, how can you reduce noise?



[Source: S. Seitz]

Image Filtering

- Modify the pixels in an image based on some function of a local neighborhood of each pixel
- In other words... Filtering

10	5	3
4	5	1
1	1	7

Local image data

Some function



	7	

Modified image data

[Source: L. Zhang]

Applications of Filtering

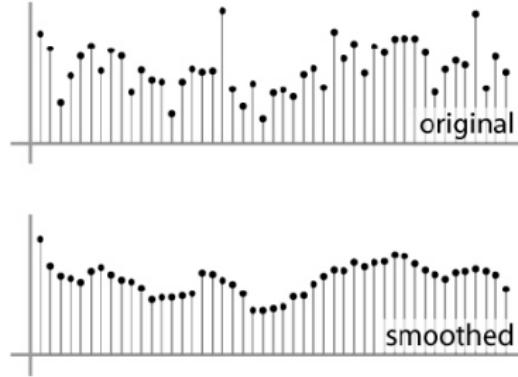
- Enhance an image, e.g., **denoise**.
- Detect patterns, e.g., **template matching**.
- Extract information, e.g., **texture, edges**.

Applications of Filtering

- Enhance an image, e.g., **denoise**. Let's talk about this first
- Detect patterns, e.g., **template matching**.
- Extract information, e.g., **texture, edges**.

Noise reduction

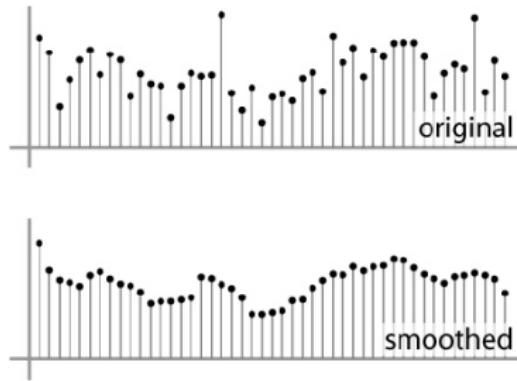
- Simplest thing: replace each pixel by the average of its neighbors.
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.



[Source: S. Marschner]

Noise reduction

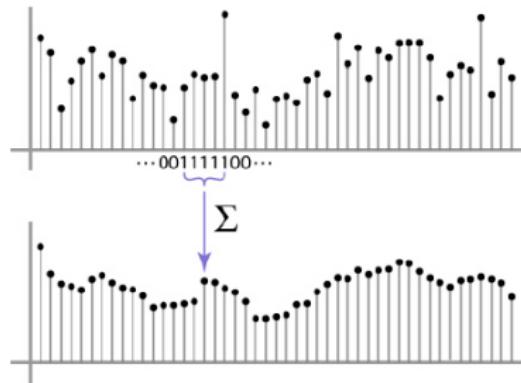
- Simplest thing: replace each pixel by the average of its neighbors.
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.



[Source: S. Marschner]

Noise reduction

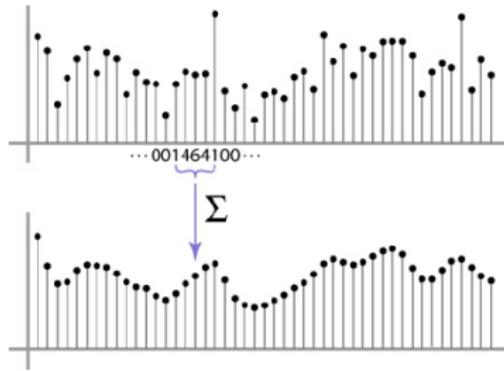
- Simplest thing: replace each pixel by the average of its neighbors
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.
- **Moving average** in 1D: $[1, 1, 1, 1, 1]/5$



[Source: S. Marschner]

Noise reduction

- Simplest thing: replace each pixel by the average of its neighbors
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.
- Non-uniform weights $[1, 4, 6, 4, 1] / 16$

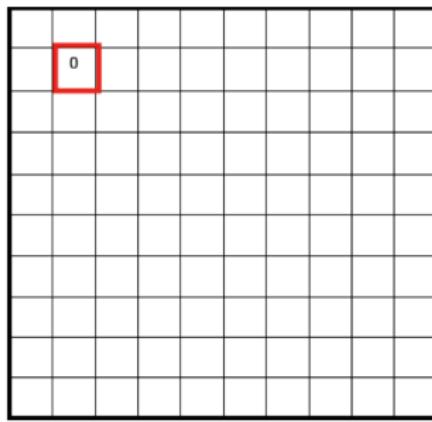


[Source: S. Marschner]

Moving Average in 2D

 $I(i, j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

 $G(i, j)$ 

[Source: S. Seitz]

Moving Average in 2D

$$I(i, j)$$

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0	0
0	0	0	90	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	90	0	0	0
0	0	0	90	0	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	90	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

$$G(i, j)$$

0	10										

[Source: S. Seitz]

Moving Average in 2D

$$I(i, j)$$

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

$$G(i, j)$$

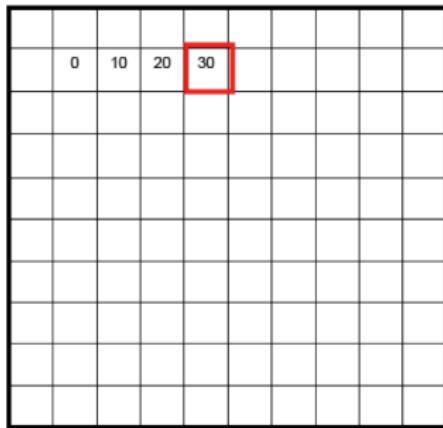
0	10	20								

[Source: S. Seitz]

Moving Average in 2D

$$I(i,j)$$

$$G(i,j)$$



[Source: S. Seitz]

Moving Average in 2D

 $I(i, j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

 $G(i, j)$

0	10	20	30	30
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

[Source: S. Seitz]

Moving Average in 2D

$$I(i, j)$$

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	0	90	90	90	0	0	0
0	0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

$$G(i, j)$$

	0	10	20	30	30	30	20	10		
	0	20	40	60	60	60	40	20		
	0	30	60	90	90	90	60	30		
	0	30	50	80	80	90	60	30		
	0	30	50	80	80	90	60	30		
	0	20	30	50	50	60	40	20		
	10	20	30	30	30	30	20	10		
	10	10	10	0	0	0	0	0		

[Source: S. Seitz]

Linear Filtering: Correlation

- Involves weighted combinations of pixels in small neighborhoods:

$$G(i, j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^k \sum_{v=-k}^k I(i+u, j+v)$$

- The output pixels value is determined as a weighted sum of input pixel values

$$G(i, j) = \sum_{u=-k}^k \sum_{v=-k}^k F(u, v) \cdot I(i+u, j+v)$$

Linear Filtering: Correlation

- Involves weighted combinations of pixels in small neighborhoods:

$$G(i, j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^k \sum_{v=-k}^k I(i+u, j+v)$$

- The output pixels value is determined as a weighted sum of input pixel values

$$G(i, j) = \sum_{u=-k}^k \sum_{v=-k}^k F(u, v) \cdot I(i+u, j+v)$$

- The entries of the weight **kernel** or **mask** $F(u, v)$ are often called the **filter coefficients**.

Linear Filtering: Correlation

- Involves weighted combinations of pixels in small neighborhoods:

$$G(i, j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^k \sum_{v=-k}^k I(i+u, j+v)$$

- The output pixels value is determined as a weighted sum of input pixel values

$$G(i, j) = \sum_{u=-k}^k \sum_{v=-k}^k F(u, v) \cdot I(i+u, j+v)$$

- The entries of the weight **kernel** or **mask** $F(u, v)$ are often called the **filter coefficients**.
- This operator is the **correlation** operator

$$G = F \otimes I$$

Linear Filtering: Correlation

- Involves weighted combinations of pixels in small neighborhoods:

$$G(i, j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^k \sum_{v=-k}^k I(i+u, j+v)$$

- The output pixels value is determined as a weighted sum of input pixel values

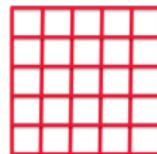
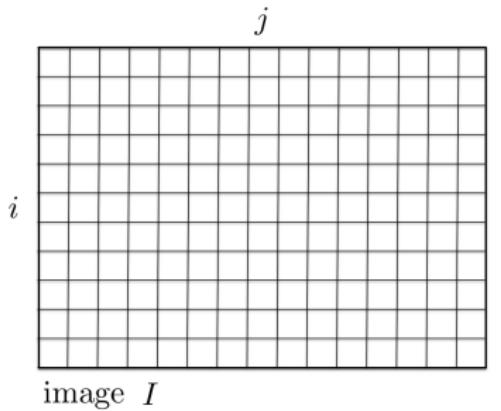
$$G(i, j) = \sum_{u=-k}^k \sum_{v=-k}^k F(u, v) \cdot I(i+u, j+v)$$

- The entries of the weight **kernel** or **mask** $F(u, v)$ are often called the **filter coefficients**.
- This operator is the **correlation** operator

$$G = F \otimes I$$

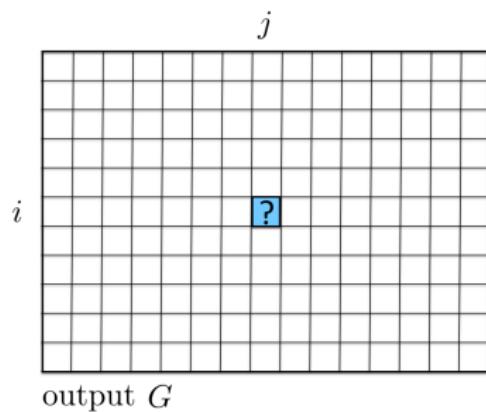
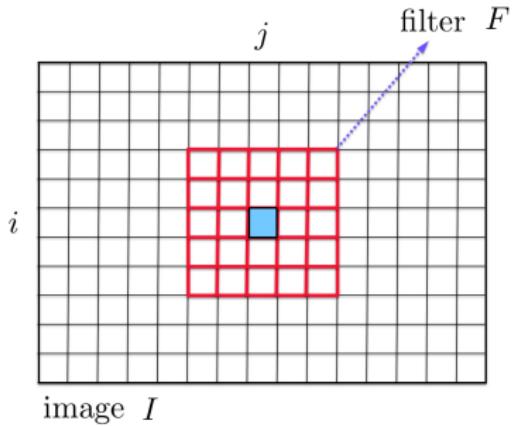
Linear Filtering: Correlation

- It's really easy!



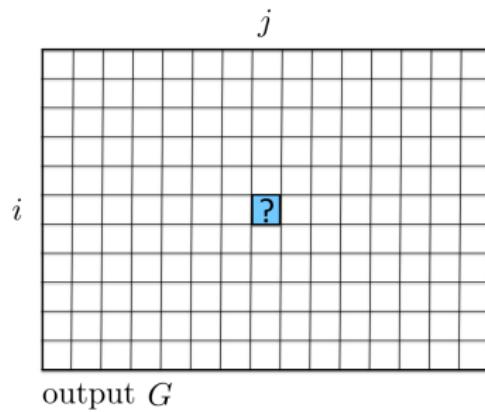
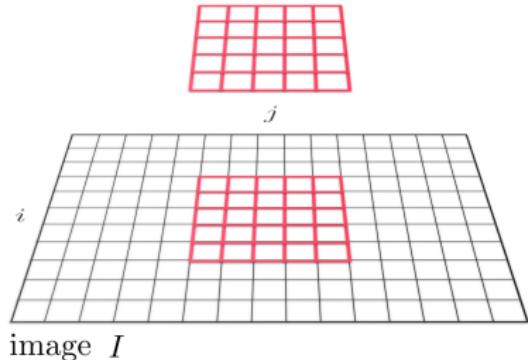
Linear Filtering: Correlation

- It's really easy!



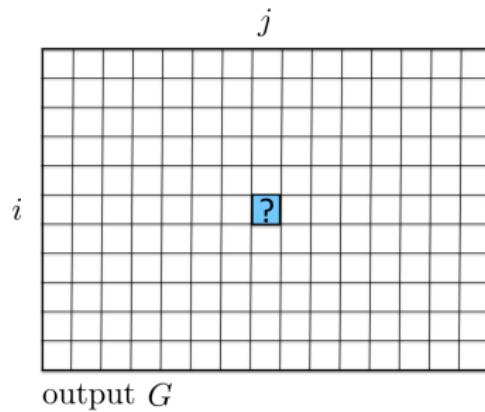
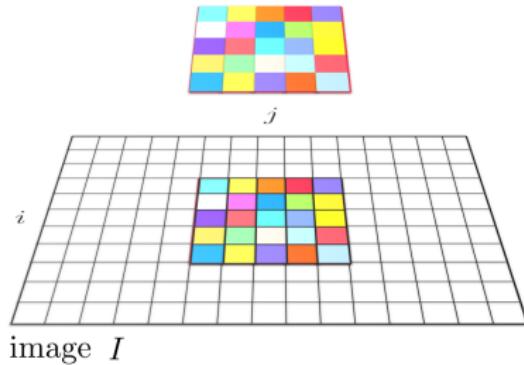
Linear Filtering: Correlation

- It's really easy!



Linear Filtering: Correlation

- It's really easy!

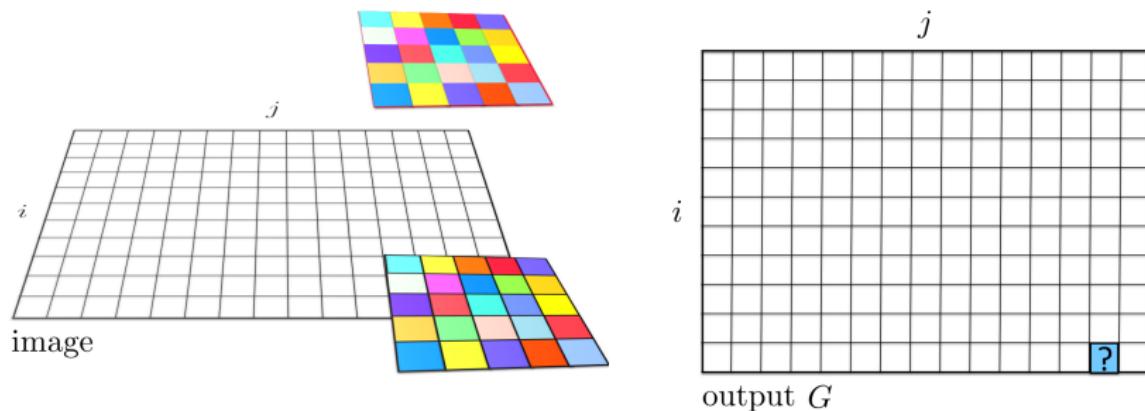


$$G(i, j) = \sum_{u=-k}^k \sum_{v=-k}^k F(u, v) \cdot I(i+u, j+v)$$

$$G(i, j) = F(\square) \cdot I(\square) + F(\square) \cdot I(\square) + F(\square) \cdot I(\square) + \dots + F(\square) \cdot I(\square)$$

Linear Filtering: Correlation

- What happens along the borders of the image?



$$G(i, j) = \sum_{u=-k}^k \sum_{v=-k}^k F(u, v) \cdot I(i + u, j + v)$$

$$G(i, j) = F(\square) \cdot I(\square) + F(\square) \cdot I(\square) + F(\square) \cdot I(\square) + \dots + F(\square) \cdot I(\square)$$

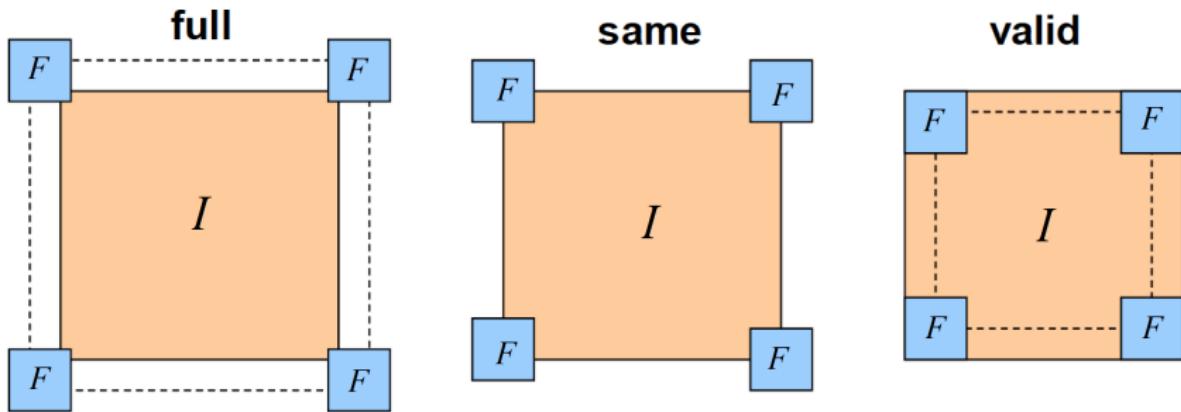
Boundary Effects

- What happens at the border of the image? What's the size of the output matrix?
- MATLAB: `FILTER2(I, F, SHAPE)`
- `shape = 'full'` output size is bigger than the image
- `shape = 'same'`: output size is same as `I`
- `shape = 'valid'`: output size is smaller than the image

[Source: S. Lazebnik]

Boundary Effects

- What happens at the border of the image? What's the size of the output matrix?
- MATLAB: FILTER2(I, F, SHAPE)
- shape = 'full' output size is bigger than the image
- shape = 'same': output size is same as I
- shape = 'valid': output size is smaller than the image



[Source: S. Lazebnik]

Filtering with Correlation: Example

- What's the result?



Original

0	0	0
0	1	0
0	0	0

?

[Source: D. Lowe]

Filtering with Correlation: Example

- What's the result?



Original

0	0	0
0	1	0
0	0	0



**Filtered
(no change)**

[Source: D. Lowe]

Filtering with Correlation: Example

- What's the result?



0	0	0
0	0	1
0	0	0

?

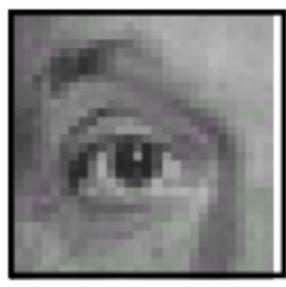
[Source: D. Lowe]

Filtering with Correlation: Example

- What's the result?



0	0	0
0	0	1
0	0	0



[Source: D. Lowe]

Filtering with Correlation: Example

- What's the result?


$$\text{Original} \quad * \left(\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{array} \right) - \frac{1}{9} \left(\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array} \right) =$$

[Source: D. Lowe]

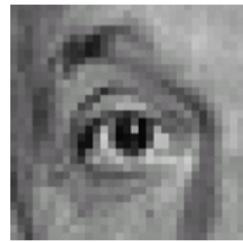
Filtering with Correlation: Example

- What's the result?



Original

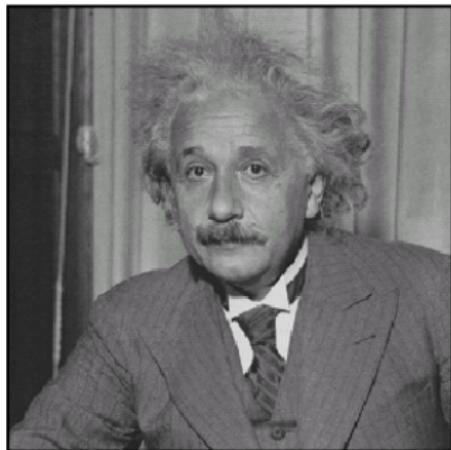
$$\text{Original} * \left(\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{array} - \frac{1}{9} \begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array} \right) =$$



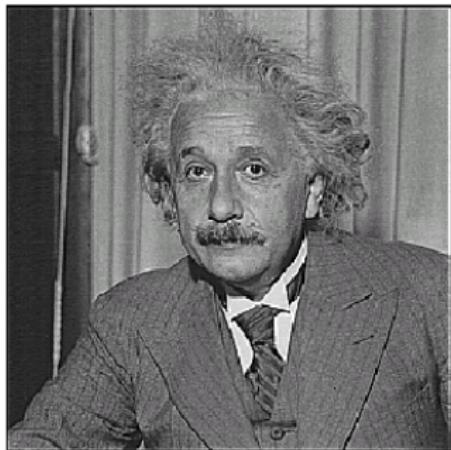
Sharpening filter
(accentuates edges)

[Source: D. Lowe]

Sharpening



before



after

This is a prelude to edge detection (next time)! [Source: D. Lowe]

Sharpening



[Source: N. Snavely]

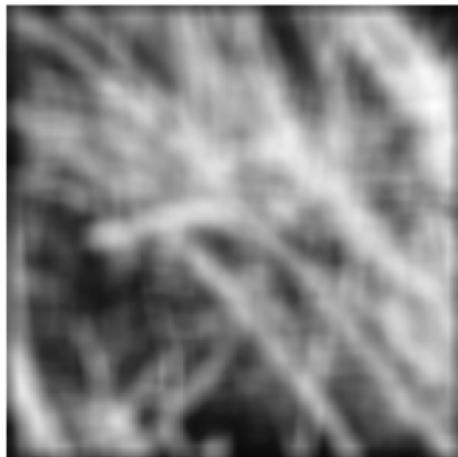
Smoothing by averaging



depicts box filter:
white = high value, black = low value



original



filtered

- What if the filter size was 5×5 instead of 3×3 ?

[Source: K. Graumann]

Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?
- Removes high-frequency components from the image (low-pass filter).

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	0	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

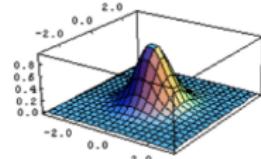
$$I(i, j)$$

$$\frac{1}{16} \begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix}$$

$$F(i, j)$$

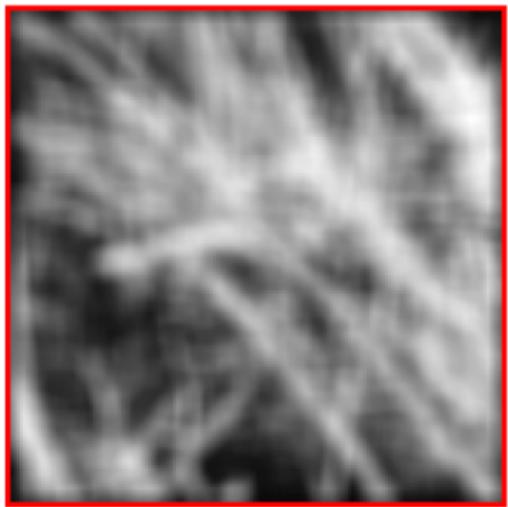
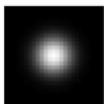
This kernel is an approximation of a 2d Gaussian function:

$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



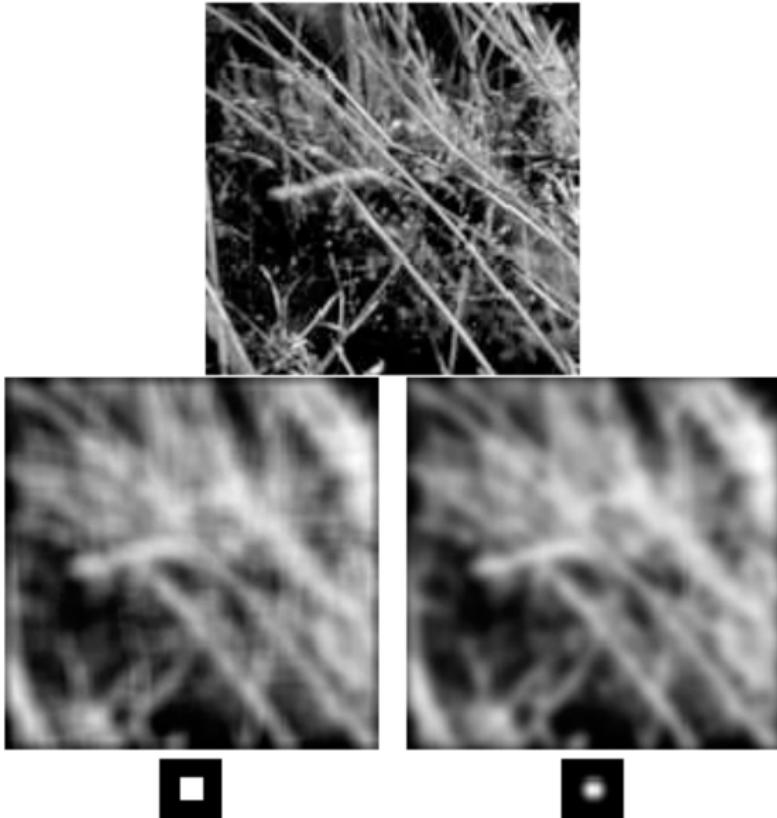
[Source: S. Seitz]

Smoothing with a Gaussian



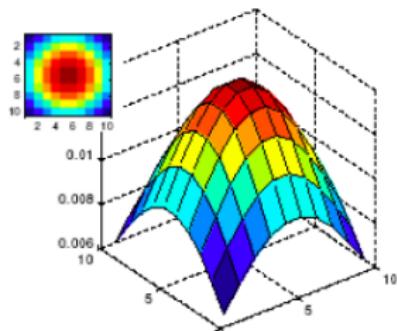
[Source: K. Grauman]

Mean vs Gaussian

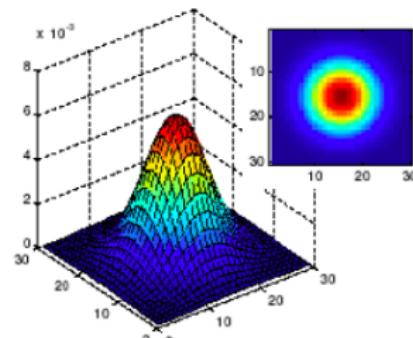


Gaussian filter: Parameters

- **Size of filter or mask:** Gaussian function has infinite support, but discrete filters use finite kernels.



$\sigma = 5$ with
10 x 10
kernel

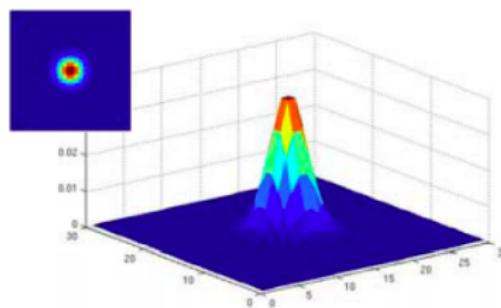


$\sigma = 5$ with
30 x 30
kernel

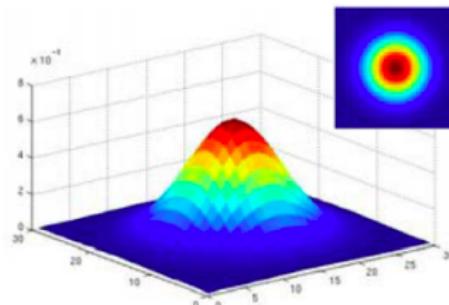
[Source: K. Grauman]

Gaussian filter: Parameters

- **Variance of the Gaussian:** determines extent of smoothing.



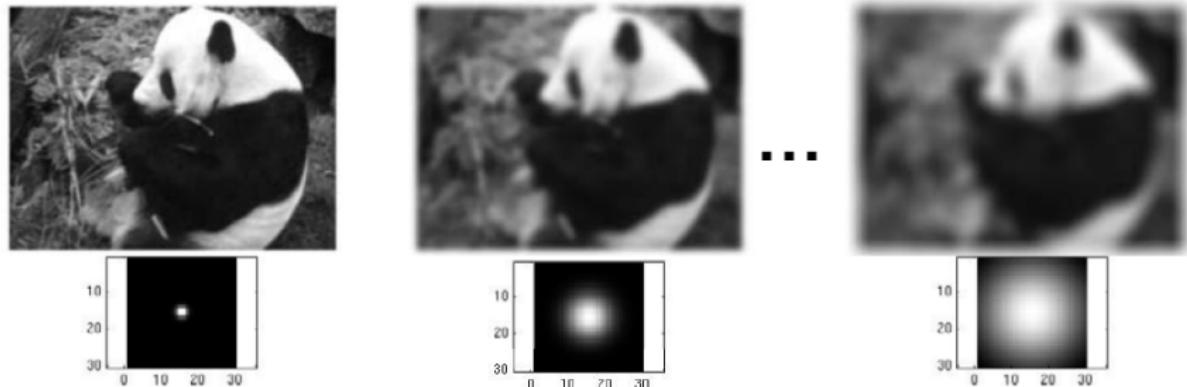
$\sigma = 2$ with
30 x 30
kernel



$\sigma = 5$ with
30 x 30
kernel

[Source: K. Grauman]

Gaussian filter: Parameters



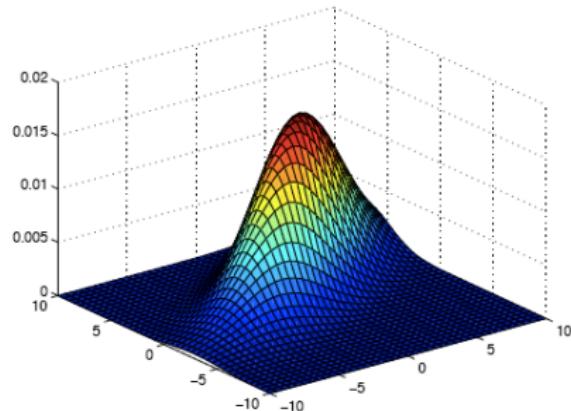
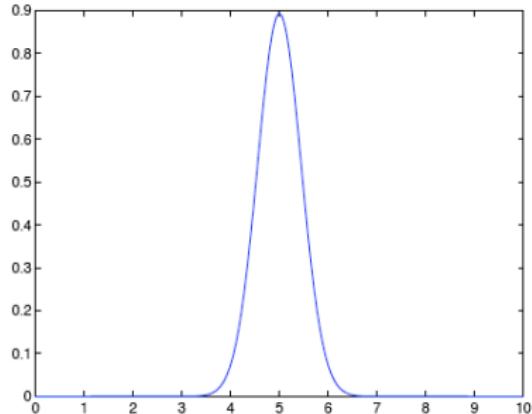
```
for sigma=1:3:10
    h = fspecial('gaussian', fsize, sigma);
    out = imfilter(im, h);
    imshow(out);
    pause;
end
```

[Source: K. Grauman]

Is this the most general Gaussian?

- No, the most general form is anisotropic (i.e not symmetric) $\mathbf{x} \in \Re^d$

$$\mathcal{N}(\mathbf{x}; \mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\right)$$



- But the simplified version is typically used for filtering.

Properties of the Smoothing

- All values are positive.
- They all sum to 1 to prevent re-scaling of the image.

Properties of the Smoothing

- All values are positive.
- They all sum to 1 to prevent re-scaling of the image.
- Remove high-frequency components; low-pass filter.

Properties of the Smoothing

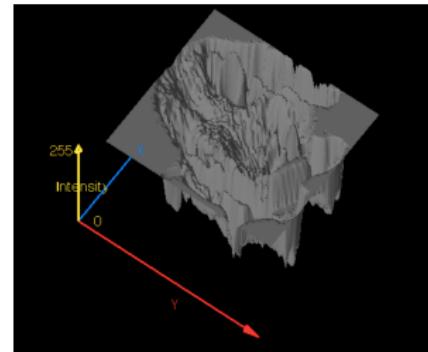
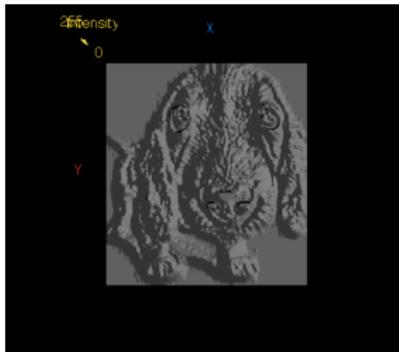
- All values are positive.
- They all sum to 1 to prevent re-scaling of the image.
- Remove high-frequency components; low-pass filter.
- What is frequency in this context?

Properties of the Smoothing

- All values are positive.
- They all sum to 1 to prevent re-scaling of the image.
- Remove high-frequency components; low-pass filter.
- What is frequency in this context?
- Edges!

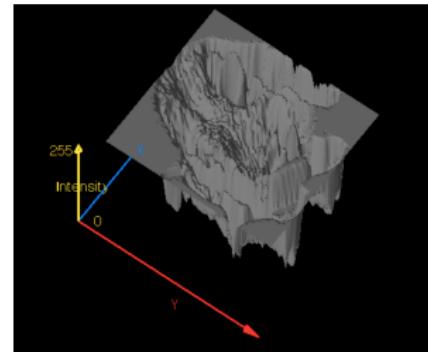
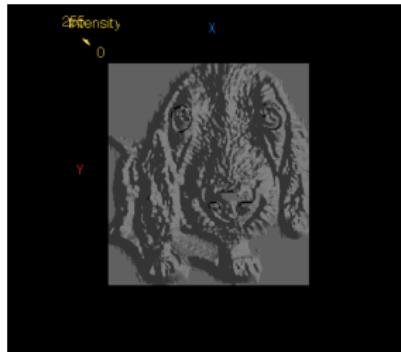
Properties of the Smoothing

- All values are positive.
- They all sum to 1 to prevent re-scaling of the image.
- Remove high-frequency components; low-pass filter.
- What is frequency in this context?
- Edges!



Properties of the Smoothing

- All values are positive.
- They all sum to 1 to prevent re-scaling of the image.
- Remove high-frequency components; low-pass filter.
- What is frequency in this context?
- Edges!



Finding Waldo



image /

- How can we use what we just learned to find Waldo?

Finding Waldo



image I

filter F

- Is correlation a good choice?

A Slight Detour: Correlation in Matrix Form

- Remember correlation:

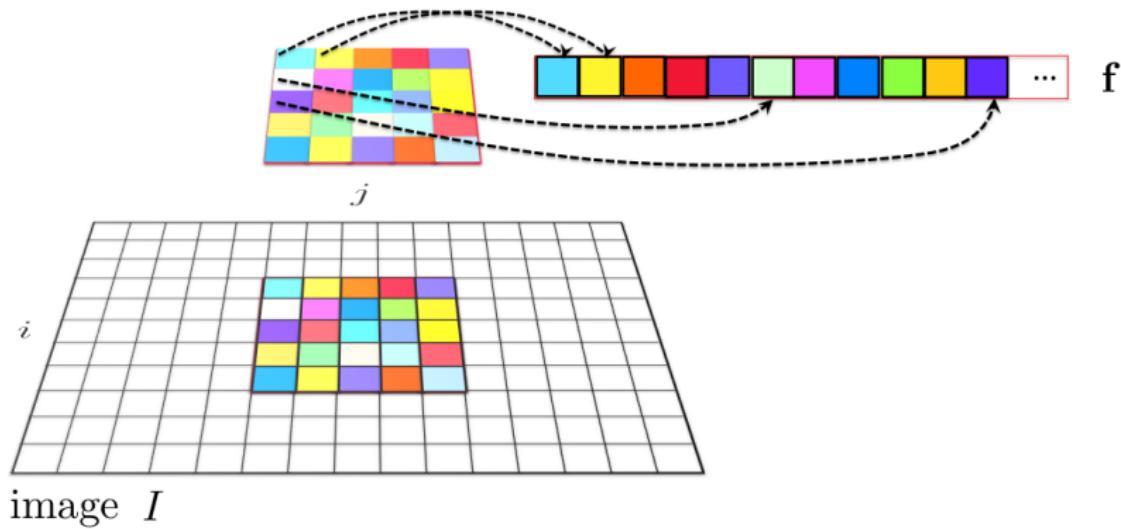
$$G(i, j) = \sum_{u=-k}^k \sum_{v=-k}^k F(u, v) \cdot I(i + u, j + v)$$

- Can we write that in a more compact form (with vectors)?

A Slight Detour: Correlation in Matrix Form

- Remember correlation:

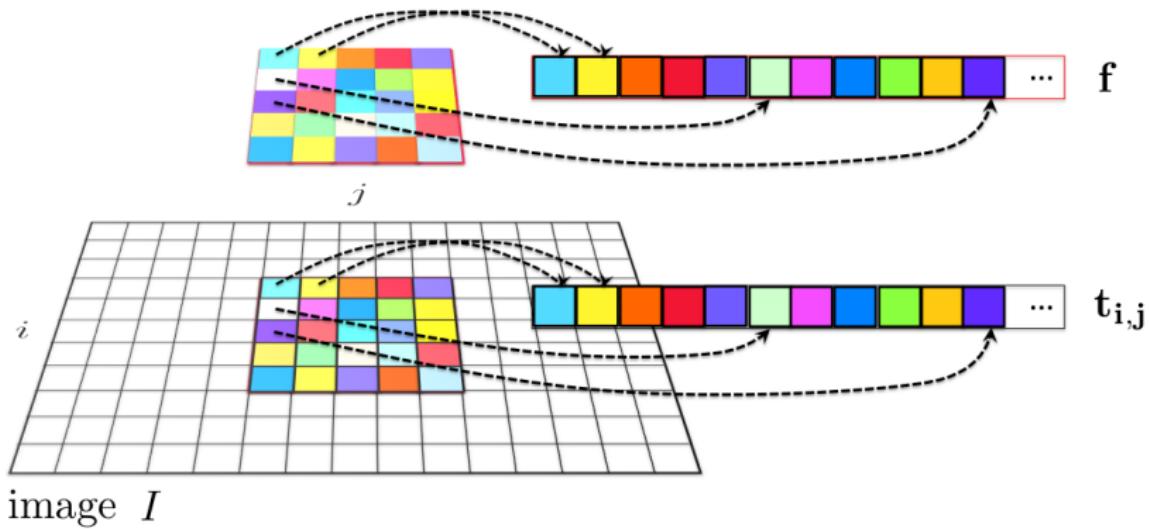
$$G(i, j) = \sum_{u=-k}^k \sum_{v=-k}^k F(u, v) \cdot I(i + u, j + v)$$



A Slight Detour: Correlation in Matrix Form

- Remember correlation:

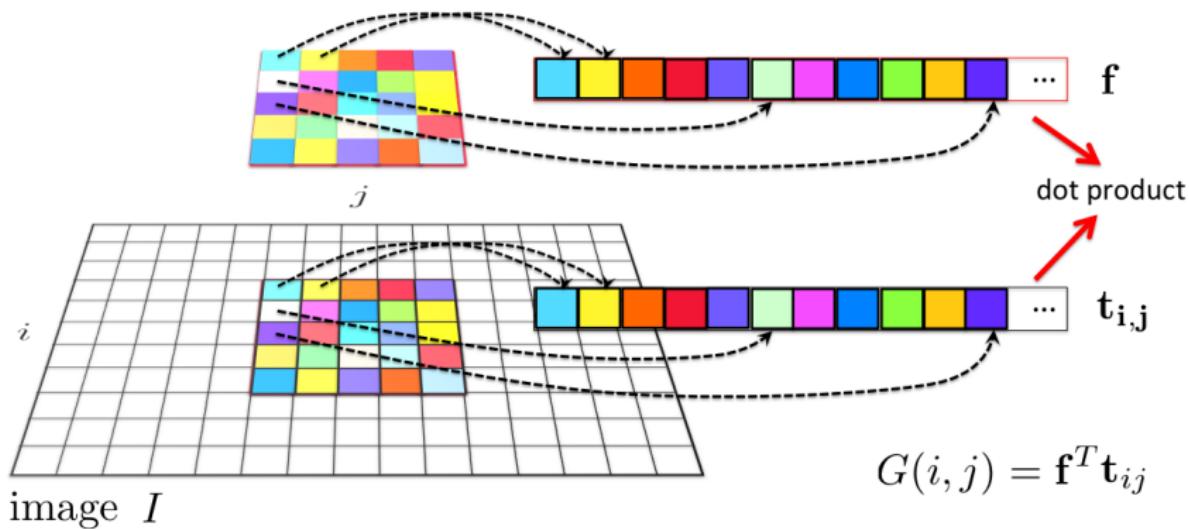
$$G(i, j) = \sum_{u=-k}^k \sum_{v=-k}^k F(u, v) \cdot I(i + u, j + v)$$



A Slight Detour: Correlation in Matrix Form

- Remember correlation:

$$G(i, j) = \sum_{u=-k}^k \sum_{v=-k}^k F(u, v) \cdot I(i + u, j + v)$$



A Slight Detour: Correlation in Matrix Form

- Remember correlation:

$$G(i, j) = \sum_{u=-k}^k \sum_{v=-k}^k F(u, v) \cdot I(i + u, j + v)$$

- Can we write that in a more compact form (with vectors)?
- Define $\mathbf{f} = F(:)$, $T_{ij} = I(i - k : i + k, j - k : j + k)$, and $\mathbf{t}_{ij} = T_{ij}(:)$

$$G(i, j) = \mathbf{f} \cdot \mathbf{t}_{ij}$$

where \cdot is a dot product

A Slight Detour: Correlation in Matrix Form

- Remember correlation:

$$G(i, j) = \sum_{u=-k}^k \sum_{v=-k}^k F(u, v) \cdot I(i + u, j + v)$$

- Can we write that in a more compact form (with vectors)?
- Define $\mathbf{f} = F(:)$, $T_{ij} = I(i - k : i + k, j - k : j + k)$, and $\mathbf{t}_{ij} = T_{ij}(::)$

$$G(i, j) = \mathbf{f} \cdot \mathbf{t}_{ij}$$

where \cdot is a dot product

- Homework:** Can we write full correlation $G = F \otimes I$ in matrix form?

A Slight Detour: Correlation in Matrix Form

- Remember correlation:

$$G(i, j) = \sum_{u=-k}^k \sum_{v=-k}^k F(u, v) \cdot I(i + u, j + v)$$

- Can we write that in a more compact form (with vectors)?
- Define $\mathbf{f} = F(:)$, $T_{ij} = I(i - k : i + k, j - k : j + k)$, and $\mathbf{t}_{ij} = T_{ij}(:)$

$$G(i, j) = \mathbf{f} \cdot \mathbf{t}_{ij}$$

where \cdot is a dot product

- Finding Waldo: How could we ensure to get the best “score” (e.g. 1) for an image crop that looks exactly like our filter?

A Slight Detour: Correlation in Matrix Form

- Remember correlation:

$$G(i, j) = \sum_{u=-k}^k \sum_{v=-k}^k F(u, v) \cdot I(i + u, j + v)$$

- Can we write that in a more compact form (with vectors)?
- Define $\mathbf{f} = F(:)$, $T_{ij} = I(i - k : i + k, j - k : j + k)$, and $\mathbf{t}_{ij} = T_{ij}(:)$

$$G(i, j) = \mathbf{f} \cdot \mathbf{t}_{ij}$$

where \cdot is a dot product

- Finding Waldo: How could we ensure to get the best “score” (e.g. 1) for an image crop that looks exactly like our filter?
- Normalized cross-correlation:**

$$G(i, j) = \frac{\mathbf{f}^T \mathbf{t}_{ij}}{\|\mathbf{f}\| \|\mathbf{t}_{ij}\|}$$

Back to Waldo

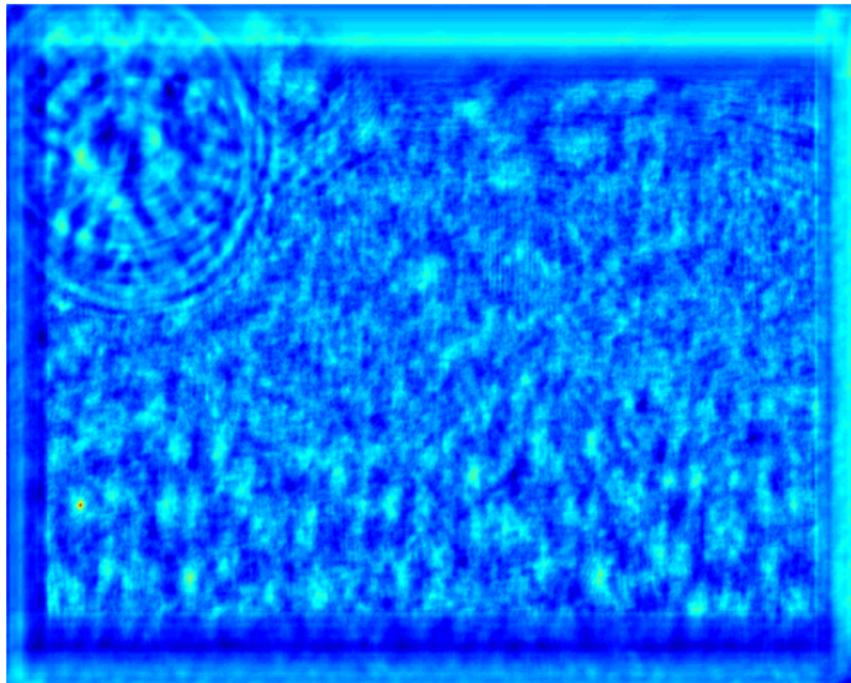


image I



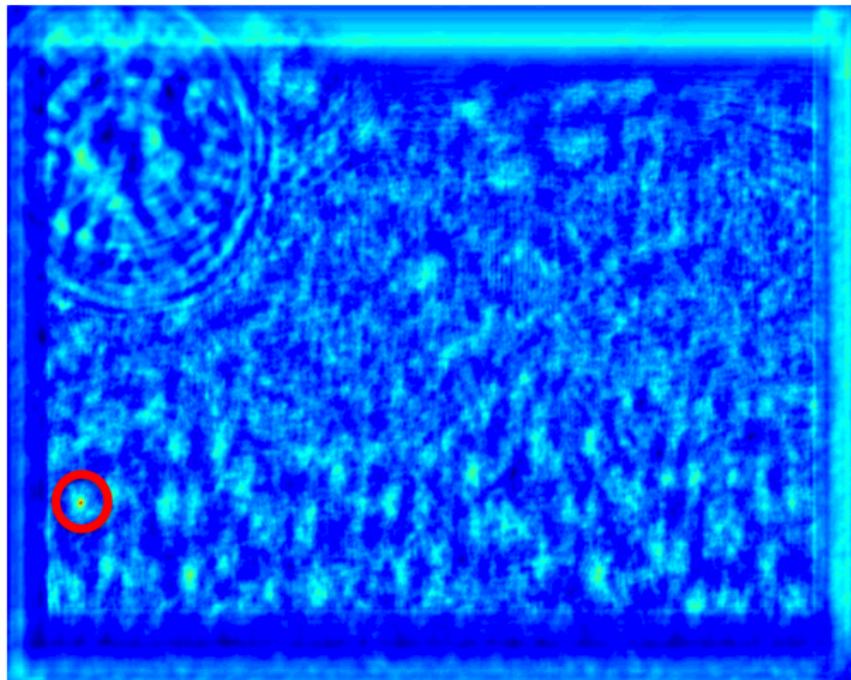
filter F

Back to Waldo



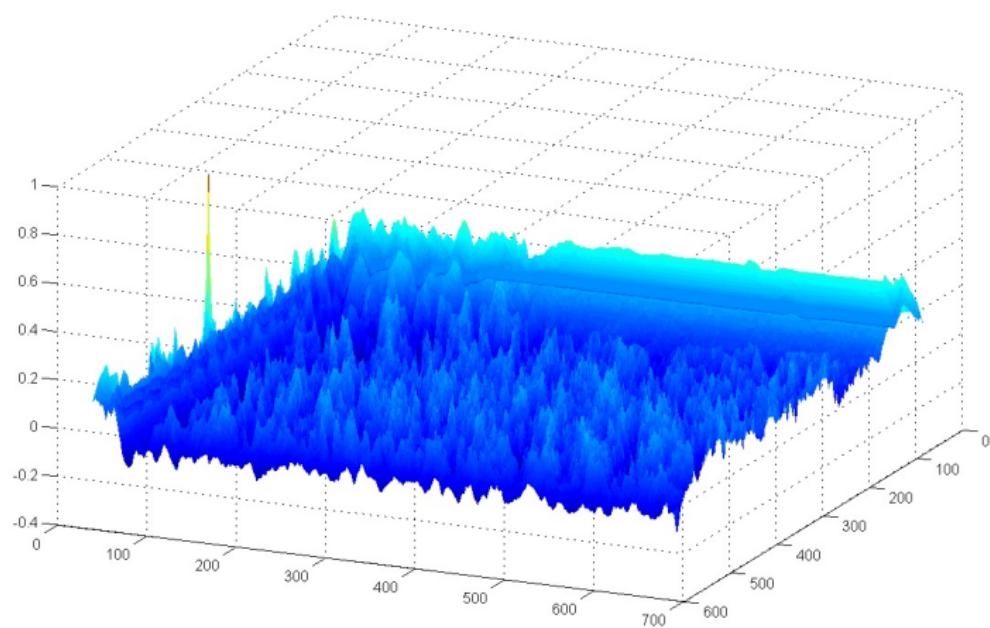
- Result of normalized cross-correlation

Back to Waldo



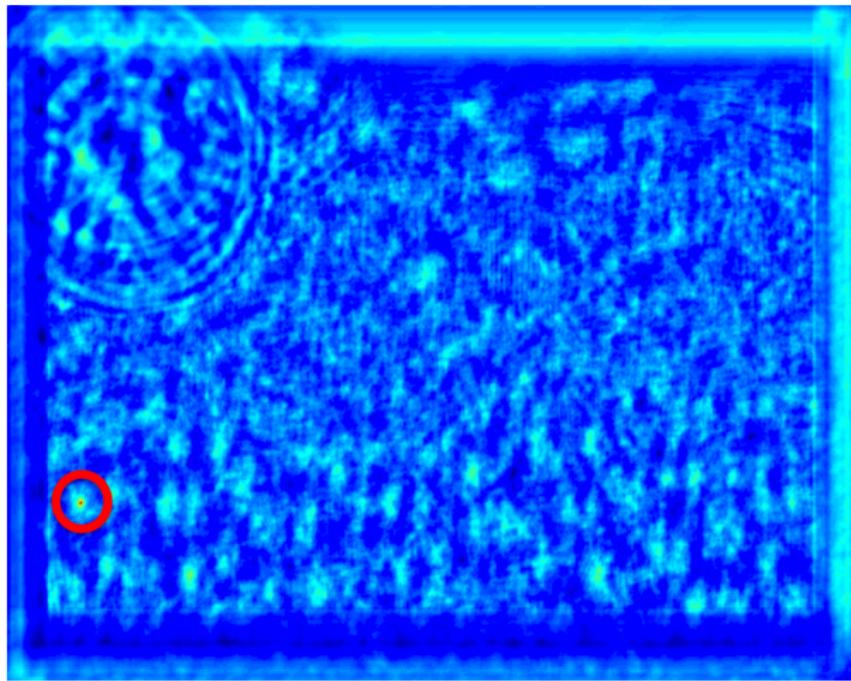
- Find the highest peak

Back to Waldo



- Find the highest peak

Back to Waldo



- Find the highest peak

Back to Waldo



And put a bounding box (rectangle the size of the template) at the point!

Example of Correlation

- What is the result of filtering the impulse signal (image) I with the arbitrary filter F ?

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



a	b	c
d	e	f
g	h	i

$$F(i, j)$$

$$I(i, j)$$

$$G(i, j)$$

[Source: K. Grauman]

Convolution

- **Convolution** operator

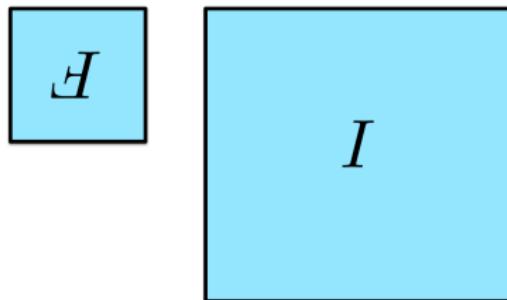
$$G(i, j) = \sum_{u=-k}^k \sum_{v=-k}^k F(u, v) \cdot I(i - u, j - v)$$

Convolution

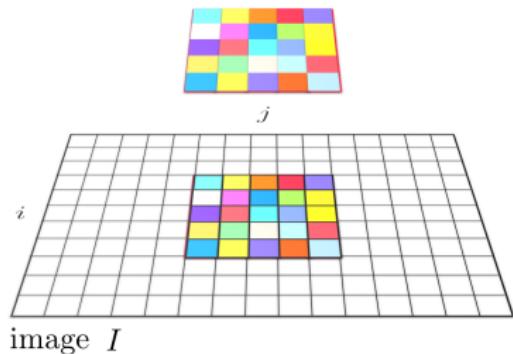
- **Convolution** operator

$$G(i, j) = \sum_{u=-k}^k \sum_{v=-k}^k F(u, v) \cdot I(i - u, j - v)$$

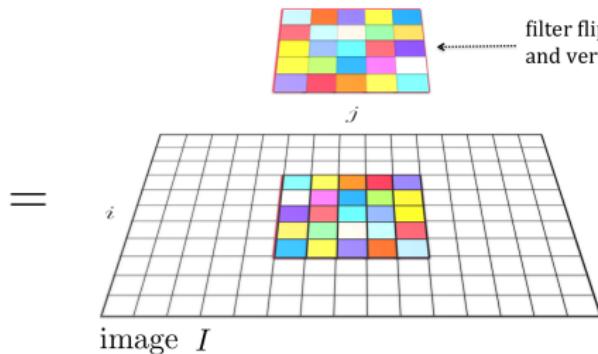
- **Equivalent** to flipping the filter in both dimensions (bottom to top, right to left) and apply correlation.



Correlation vs Convolution



Correlation



Convolution

Correlation vs Convolution

- For a Gaussian or box filter, how will the outputs $F * I$ and $F \otimes I$ differ?

Correlation vs Convolution

- For a Gaussian or box filter, how will the outputs $F * I$ and $F \otimes I$ differ?
- How will the outputs differ for:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

“Optical” Convolution

- Camera Shake



Figure: Fergus, et al., SIGGRAPH 2006

- Blur in out-of-focus regions of an image.



Figure: Bokeh: <http://lullaby.homepage.dk/diy-camera/bokeh.html>

Click for more info

[Source: N. Snavely]

Properties of Convolution

Commutative : $f * g = g * f$

Associative : $f * (g * h) = (f * g) * h$

Distributive : $f * (g + h) = f * g + f * h$

Assoc. with scalar multiplier : $\lambda \cdot (f * g) = (\lambda \cdot f) * g$

Properties of Convolution

Commutative : $f * g = g * f$

Associative : $f * (g * h) = (f * g) * h$

Distributive : $f * (g + h) = f * g + f * h$

Assoc. with scalar multiplier : $\lambda \cdot (f * g) = (\lambda \cdot f) * g$

- The Fourier transform of two convolved images is the product of their individual Fourier transforms:

$$\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g)$$

Properties of Convolution

Commutative : $f * g = g * f$

Associative : $f * (g * h) = (f * g) * h$

Distributive : $f * (g + h) = f * g + f * h$

Assoc. with scalar multiplier : $\lambda \cdot (f * g) = (\lambda \cdot f) * g$

- The Fourier transform of two convolved images is the product of their individual Fourier transforms:

$$\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g)$$

- **Homework:** Why is this good news?
- **Hint:** Think of complexity of convolution and Fourier Transform
- What if we wanted to undo the result of convolution?

Separable Filters: Speed-up Trick!

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter.

Separable Filters: Speed-up Trick!

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter.
- Can we do faster?

Separable Filters: Speed-up Trick!

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter.
- Can we do faster?
- In many cases (**not all!**), this operation can be speed up by first performing a 1D horizontal convolution followed by a 1D vertical convolution, **requiring only $2K$ operations**.

Separable Filters: Speed-up Trick!

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter.
- Can we do faster?
- In many cases (**not all!**), this operation can be speed up by first performing a 1D horizontal convolution followed by a 1D vertical convolution, **requiring only $2K$ operations**.
- If this is possible, then the convolution filter is called **separable**.

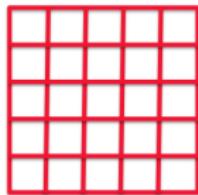
Separable Filters: Speed-up Trick!

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter.
- Can we do faster?
- In many cases (**not all!**), this operation can be speed up by first performing a 1D horizontal convolution followed by a 1D vertical convolution, **requiring only $2K$ operations**.
- If this is possible, then the convolution filter is called **separable**.
- And it is the outer product of two filters:

$$\mathbf{F} = \mathbf{v} \mathbf{h}^T$$

[Source: R. Urtasun]

How it Works



filter

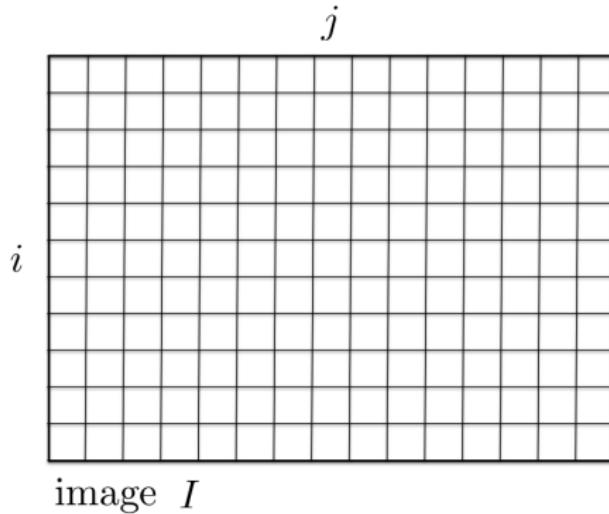
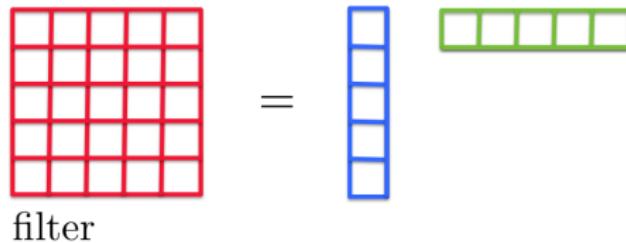
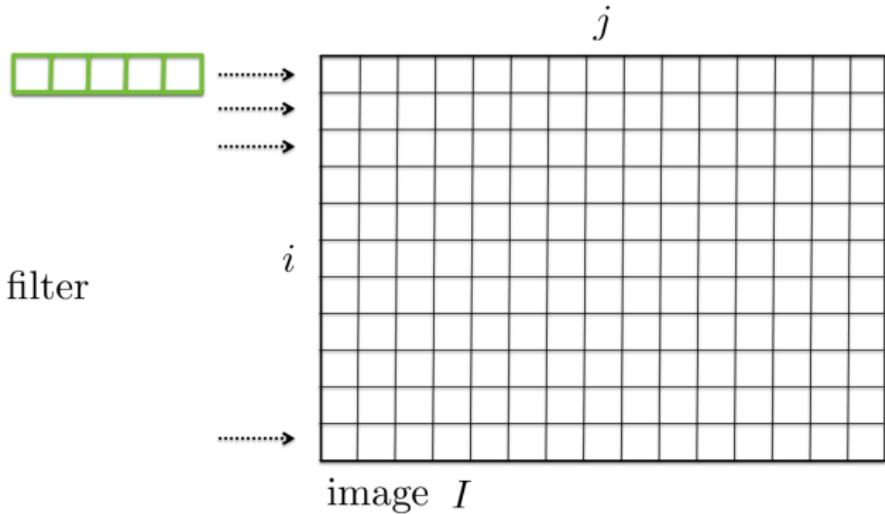


image I

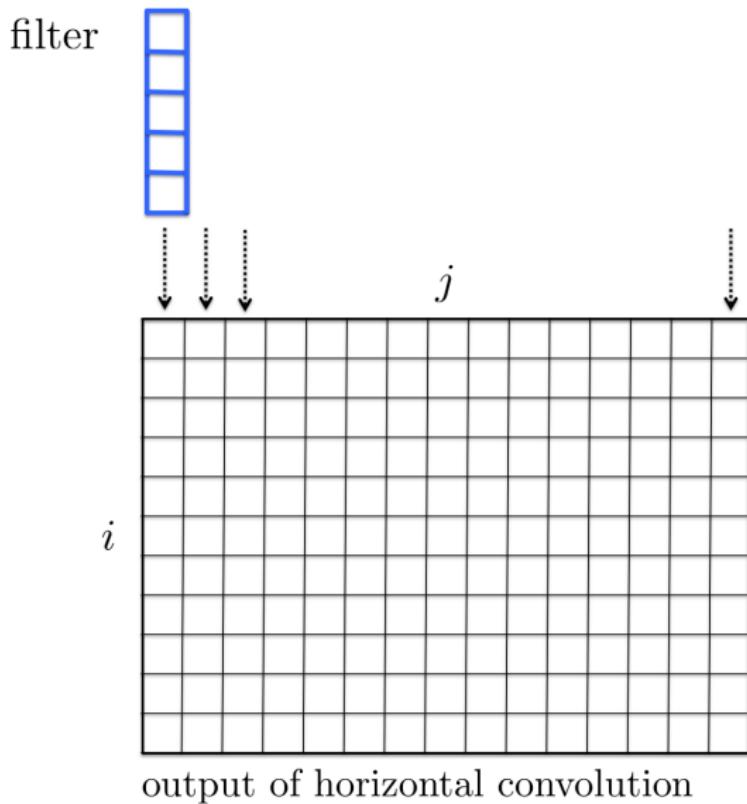
How it Works



How it Works



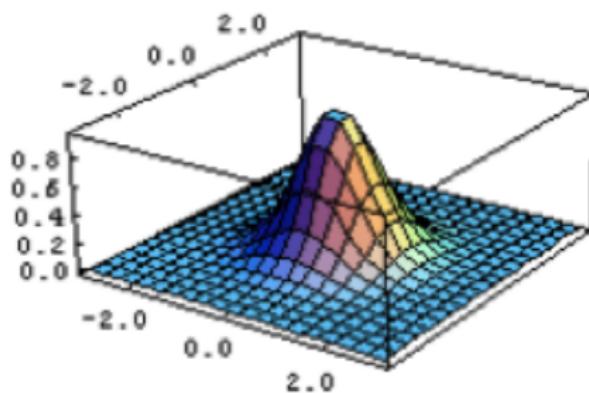
How it Works



Separable Filters: Gaussian filters

- One famous separable filter we already know:

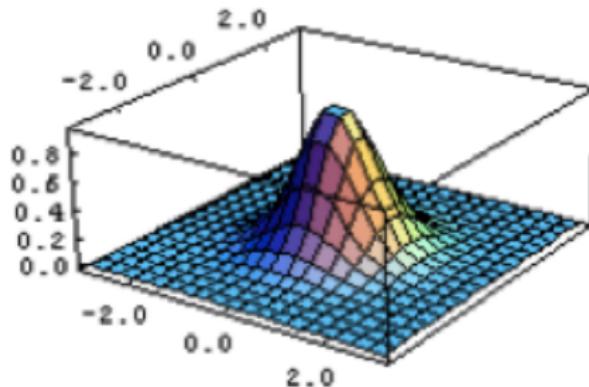
$$\text{Gaussian} : f(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}}$$



Separable Filters: Gaussian filters

- One famous separable filter we already know:

$$\text{Gaussian} : f(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}}$$
$$= \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{\sigma^2}} \right) \cdot \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{\sigma^2}} \right)$$



Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{K^2} \begin{array}{|c|c|c|c|} \hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array}$$

[Source: R. Urtasun]

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{K^2} \begin{matrix} \begin{array}{|c|c|c|c|} \hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array} & \frac{1}{K} \begin{array}{|c|c|c|c|} \hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \end{matrix}$$

What does this filter do?

[Source: R. Urtasun]

Let's play a game...

Is this separable? If yes, what's the separable version?

$\frac{1}{16}$	1	2	1
	2	4	2
	1	2	1

[Source: R. Urtasun]

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

What does this filter do?

[Source: R. Urtasun]

Let's play a game...

Is this separable? If yes, what's the separable version?

$\frac{1}{8}$	-1	0	1
	-2	0	2
	-1	0	1

[Source: R. Urtasun]

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$
$$\frac{1}{2} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

What does this filter do?

[Source: R. Urtasun]

How can we tell if a given filter F is indeed separable?

- Inspection... this is what we were doing.

How can we tell if a given filter F is indeed separable?

- Inspection... this is what we were doing.
- Looking at the analytic form of it.

How can we tell if a given filter F is indeed separable?

- Inspection... this is what we were doing.
- Looking at the analytic form of it.
- Look at the **singular value decomposition (SVD)**, and if only one singular value is non-zero, then it is separable

$$F = \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \sigma_i u_i v_i^T$$

with $\Sigma = \text{diag}(\sigma_i)$.

How can we tell if a given filter F is indeed separable?

- Inspection... this is what we were doing.
- Looking at the analytic form of it.
- Look at the **singular value decomposition (SVD)**, and if only one singular value is non-zero, then it is separable

$$F = \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \sigma_i u_i v_i^T$$

with $\Sigma = \text{diag}(\sigma_i)$.

- Matlab: $[U,S,V] = \text{svd}(F);$

How can we tell if a given filter F is indeed separable?

- Inspection... this is what we were doing.
- Looking at the analytic form of it.
- Look at the **singular value decomposition (SVD)**, and if only one singular value is non-zero, then it is separable

$$F = \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \sigma_i u_i v_i^T$$

with $\Sigma = \text{diag}(\sigma_i)$.

- Matlab: $[\mathbf{U}, \mathbf{S}, \mathbf{V}] = \text{svd}(F)$;
- $\sqrt{\sigma_1} \mathbf{u}_1$ and $\sqrt{\sigma_1} \mathbf{v}_1^T$ are the vertical and horizontal filter.

[Source: R. Urtasun]

Summary – Stuff You Should Know

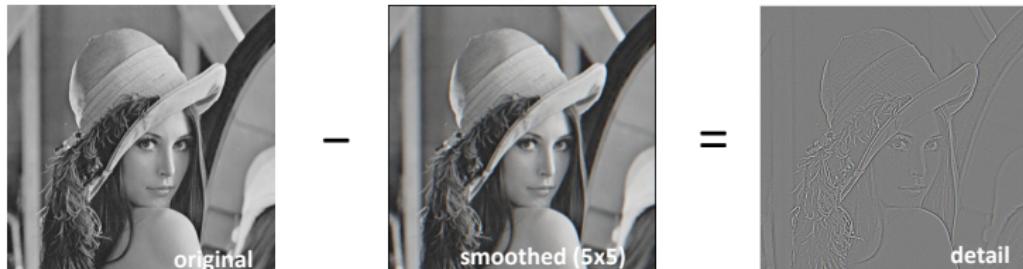
- **Correlation:** Slide a filter across image and compare (via dot product)
- **Convolution:** Flip the filter to the right and down and do correlation
- **Smooth** image with a Gaussian kernel: bigger σ means more blurring
- **Some** filters (like Gaussian) are **separable**: you can filter faster. First apply 1D convolution to each row, followed by another 1D conv. to each column

Matlab functions:

- IMFILTER: can do both correlation and convolution
- CORR2, FILTER2: correlation, NORMXCORR2 normalized correlation
- CONV2: does convolution
- FSPECIAL: creates special filters including a Gaussian

Edges

- What does blurring take away?



[Source: S. Lazebnik]

Next time:
Edge Detection