Announcements

- **Class Website:**
  
  http://www.teach.cs.toronto.edu/ csc420h/fall/

- Website link is on Piazza

- **Tutorial:**
  
  First Tutorial on Thursday, Sep 14, at 4pm (Matlab for Images) - Jake

- **Waitlist...:**
Syllabus Overview

- **Image Processing**
  - Linear Filters
  - Edge Detection
  - Image Pyramids

- **Features & Matching**
  - KeyPoint 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

Ahmed Ashraf  
CSC420: Intro to Image Understanding  
2 / 49
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

Matrix $I$ can be $m \times n$ (grayscale) or $m \times n \times 3$ (colour)

```
pixel (1, 1): intensity 255
```

Ahmed Ashraf  
CSC420: Intro to Image Understanding  
4 / 49
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)
We can think of a (grayscale) image as a function $f : \mathbb{R}^2 \to \mathbb{R}$ giving the intensity at position $(i,j)$.

Intensity 0 is black and 255 is white.
As with any function, we can apply operators to an image, e.g.:

\[ J(i, j) = I(i, j) + 50 \]

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

[Adapted from: N. Snavely]
As with any function, we can apply operators to an image, e.g.:

\[ I(i, j) \]

\[ J(i, j) = I(i, j) + 50 \]

\[ J(i, j) = I(i, -j) \]

We’ll talk about special kinds of operators, \textit{correlation} and \textit{convolution} (linear filtering)

[Adapted from: N. Snavely]
As with any function, we can apply operators to an image, e.g.:

\[ I(i, j) \]
\[ J(i, j) = I(i, j) + 50 \]
\[ J(i, j) = I(i, -j) \]

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]
• 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

![Local image data](image1)

![Modified image data](image2)

[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

\[ I(i, j) \]

\[ G(i, j) \]

[Source: S. Seitz]
Moving Average in 2D

\[ I(i, j) \]

\[ G(i, j) \]

[Source: S. Seitz]
Moving Average in 2D

\[ I(i, j) \]

\[ G(i, j) \]

[Source: S. Seitz]
Moving Average in 2D

\[ I(i, j) \]

\[ G(i, j) \]

[Source: S. Seitz]
Moving Average in 2D

\[ I(i, j) \quad G(i, j) \]

[Source: S. Seitz]
Moving Average in 2D

\[ I(i, j) \]

\[ G(i, j) \]

[Source: S. Seitz]
Linear Filtering: Correlation

- Involves weighted combinations of pixels in small neighborhoods:

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

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

\[
G(i, j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u, v) \cdot l(i + u, j + v)
\]
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:

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

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

\[ G(i, j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u, v) \cdot l(i + u, j + v) \]

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

- This operator is the correlation operator

\[ G = F \otimes I \]
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!

![Image of an image I with a filter F]
Linear Filtering: Correlation

- It’s really easy!

![Diagram](image)

- filter $F$

- image $I$

- $i$

- $j$
It’s really easy!
It’s really easy!

\[ G(i, j) = k \sum_{u=1}^{k} \sum_{v=1}^{k} F(u, v) \cdot I(i+u, j+v) \]
What happens along the borders of the image?

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

Ahmed Ashraf
CSC420: Intro to Image Understanding
Boundary Effects

- What happens at the border of the image? What’s the size of the output matrix?
- MATLAB: \texttt{filter2(g, F, SHAPE)}
- \texttt{shape = “full”:} output size is bigger than the image
- \texttt{shape = “same”:} output size is same as \textit{f}
- \texttt{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: \texttt{filter2(g, f, \text{shape})}
- \texttt{shape = “full”}: output size is bigger than the image
- \texttt{shape = “same”}: output size is same as \texttt{f}
- \texttt{shape = “valid”}: output size is smaller than the image

[Source: S. Lazebnik]
Filtering with Correlation: Example

- What’s the result?

[Source: D. Lowe]
Filtering with Correlation: Example

What’s the result?

Original

Filtered (no change)

[Source: D. Lowe]
Filtering with Correlation: Example

- What’s the result?

[Source: D. Lowe]
Filtering with Correlation: Example

- What’s the result?

[Source: D. Lowe]
Filtering with Correlation: Example

What’s the result?

Original

\[
\begin{pmatrix}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0
\end{pmatrix} - \frac{1}{9}
\begin{pmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{pmatrix} =
\]

[Source: D. Lowe]
Filtering with Correlation: Example

- What’s the result?

Original

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\]

- \[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

\[= \frac{1}{9}\]

Sharpening filter (accentuates edges)

[Source: D. Lowe]
Sharpening

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

unfiltered

filtered

[Source: N. Snavely]
Smoothing by averaging

What if the filter size was $5 \times 5$ instead of $3 \times 3$?

[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).

![Gaussian filter kernel](image)

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

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

\[
F(i, j)
\]

\[
I(i, j)
\]

[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.

[Source: K. Grauman]
Gaussian filter: Parameters

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

\[
\begin{align*}
\sigma &= 2 \text{ with } 30 \times 30 \text{ kernel} \\
\sigma &= 5 \text{ with } 30 \times 30 \text{ kernel}
\end{align*}
\]

[Source: K. Grauman]
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) $x \in \mathbb{R}^d$

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

- 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

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

image $I$

filter $F$

- Is correlation a good choice?
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:

\[ G(i, j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u, v) \cdot I(i+u, j+v) \]
Remember correlation:

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

\[
G(i, j) = f^T t_{ij}
\]
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 \( f = F(\cdot) \), \( T_{ij} = I(i - k : i + k, j - k : j + k) \), and \( t_{ij} = T_{ij}(\cdot) \)

\[ G(i, j) = f \cdot t_{ij} \]

where \( \cdot \) is a dot product
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 \( f = F(\cdot) \), \( T_{ij} = I(i - k : i + k, j - k : j + k) \), and \( t_{ij} = T_{ij}(\cdot) \)

\[ G(i, j) = f \cdot t_{ij} \]

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 \( f = F(\cdot) \), \( T_{ij} = I(i - k : i + k, j - k : j + k) \), and \( t_{ij} = T_{ij}(\cdot) \)

\[
G(i, j) = f \cdot t_{ij}
\]

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?
- 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 \( f = F(\cdot) \), \( T_{ij} = I(i - k : i + k, j - k : j + k) \), and \( t_{ij} = T_{ij}(\cdot) \)

\[ G(i, j) = f \cdot t_{ij} \]

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:**

\[ G(i, j) = \frac{f^T t_{ij}}{||f|| ||t_{ij}||} \]
image $I$

filter $F$
• Result of normalized cross-correlation
Find the highest peak
Find the highest peak
Find the highest peak
And put a bounding box (rectangle the size of the template) at the point!
Homework: Do it yourself! Code on class webpage. Don’t cheat ;)

We covered till here on Tuesday, Sep 12th 2017
Example of Correlation

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

$I(i, j)$

$F(i, j)$

$G(i, j)$

[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

filter flipped horiz. and vertically
Correlation vs Convolution

For a Gaussian or box filter, how will the outputs $F \ast I$ and $F \otimes I$ differ?
Correlation vs Convolution

- For a Gaussian or box filter, how will the outputs $F \ast I$ 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

![Camera Shake Example]

Figure: Fergus, et al., SIGGRAPH 2006

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

![Blur Example]

Figure: Bokeh: http://lullaby.homepage.dk/diy-camera/bokeh.html
Click for more info

[Source: N. Snavely]
Properties of Convolution

Commutative: \( f \ast g = g \ast f \)

Associative: \( f \ast (g \ast h) = (f \ast g) \ast h \)

Distributive: \( f \ast (g + h) = f \ast g + f \ast h \)

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

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

\[ F(f \ast g) = F(f) \cdot F(g) \]

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?
Properties of Convolution

Commutative: \( f \ast g = g \ast f \)

Associative: \( f \ast (g \ast h) = (f \ast g) \ast h \)

Distributive: \( f \ast (g + h) = f \ast g + f \ast h \)

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

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

\[
\mathcal{F}(f \ast g) = \mathcal{F}(f) \cdot \mathcal{F}(g)
\]

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?
Properties of Convolution

Commutative: \( f \ast g = g \ast f \)

Associative: \( f \ast (g \ast h) = (f \ast g) \ast h \)

Distributive: \( f \ast (g + h) = f \ast g + f \ast h \)

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

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

\[
\mathcal{F}(f \ast g) = \mathcal{F}(f) \cdot \mathcal{F}(g)
\]

- **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.
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?
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.
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.
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:

$$F = v h^T$$

[Source: R. Urtasun]
How it Works

filter

image $I$
How it Works

filter

=
How it Works

filter

output of horizontal convolution
Separable Filters: Gaussian filters

- One famous separable filter we already know:

\[ f(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}} \]
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?

\[
\begin{array}{cccc}
1 & 1 & \ldots & 1 \\
1 & 1 & \ldots & 1 \\
\frac{1}{K^2} & & & \\
& \ddots & 1 & \\
1 & 1 & \ldots & 1 \\
\end{array}
\]

[Source: R. Urtasun]
Let’s play a game...

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

\[
\begin{array}{cccc}
1 & 1 & \ldots & 1 \\
1 & 1 & \ldots & 1 \\
\vdots & \vdots & 1 & \vdots \\
1 & 1 & \ldots & 1 \\
\end{array}
\]

\[\frac{1}{K^2}\]

\[
\begin{array}{cccc}
1 & 1 & \ldots & 1 \\
\frac{1}{K} & 1 & \ldots & 1 \\
\end{array}
\]

What does this filter do?

[Source: R. Urtasun]
Let’s play a game...

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

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

[Source: R. Urtasun]
Let’s play a game...

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

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

What does this filter do?

[Source: R. Urtasun]
Let’s play a game...

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

\[
\begin{array}{ccc}
\frac{1}{8} & -1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1 \\
\end{array}
\]

[Source: R. Urtasun]
Let’s play a game...

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

\[
\begin{array}{ccc}
\frac{1}{8} & -1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1 \\
\end{array}
\]

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.

\[ F = U \Sigma V^T = \sum_i \sigma_i u_i v_i^T \]

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

Matlab:

```
[U,S,V] = svd(F);
```

$\sqrt{\sigma_1} u_1$ and $\sqrt{\sigma_1} v_1^T$ are the vertical and horizontal filter.

[Source: R. Urtasun]
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.

$$F = U \Sigma V^T = \sum_i \sigma_i u_i v_i^T$$ with $\Sigma = \text{diag}(\sigma_i)$.

Matlab:

```matlab
[U,S,V] = svd(F);
```

$\sqrt{\sigma_1} u_1$ and $\sqrt{\sigma_1} v_1^T$ are the vertical and horizontal filter.

[Source: R. Urtasun]
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

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

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

Matlab:

```
[U,S,V] = svd(F);
```

$\sqrt{\sigma_1} u_1$ and $\sqrt{\sigma_1} v_1^T$ are the vertical and horizontal filter.
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

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

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

\[ F = U\Sigma V^T = \sum_{i} \sigma_i u_i v_i^T \]

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

- Matlab: $[U,S,V] = \text{svd}(F)$;
- $\sqrt{\sigma_1} u_1$ and $\sqrt{\sigma_1} v_1^T$ 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 $\sigma$ 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