Computer Vision - Lecture 3

Linear Filters

31.10.2016

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de

leibe@vision.rwth-aachen.de
Reminder from Last Lecture

- Convert the image into binary form
  - Thresholding

- Clean up the thresholded image
  - Morphological operators

- Extract individual objects
  - Connected Components Labeling

- Describe the objects
  - Region properties
Region Properties

• From the previous steps, we can obtain separated objects.

• Some useful features can be extracted once we have connected components, including
  - Area
  - Centroid
  - Extremal points, bounding box
  - Circularity
  - Spatial moments
Area and Centroid

- We denote the set of pixels in a region by $R$
- Assuming square pixels, we obtain

- **Area:**
  
  $$A = \sum_{(x, y) \in R} 1$$

- **Centroid:**
  
  $$\bar{x} = \frac{1}{A} \sum_{(x, y) \in R} x$$

  $$\bar{y} = \frac{1}{A} \sum_{(x, y) \in R} y$$
Circularity

- Measure the deviation from a perfect circle

  \[ C = \frac{\mu_R}{\sigma_R} \]

  where \( \mu_R \) and \( \sigma_R^2 \) are the mean and variance of the distance from the centroid of the shape to the boundary pixels \((x_k, y_k)\).

- Mean radial distance:

  \[ \mu_R = \frac{1}{K} \sum_{k=0}^{K-1} \| (x_k, y_k) - (\bar{x}, \bar{y}) \| \]

- Variance of radial distance:

  \[ \sigma_R^2 = \frac{1}{K} \sum_{k=0}^{K-1} \left[ \| (x_k, y_k) - (\bar{x}, \bar{y}) \| - \mu_R \right]^2 \]

Source: Shapiro & Stockman
Invariant Descriptors

- Often, we want features independent of location, orientation, scale.

![Feature space](image)

\[ [a_1, a_2, a_3, \ldots] \quad [b_1, b_2, b_3, \ldots] \]

Feature space distance

Slide credit: Kristen Grauman
Central Moments

• $S$ is a subset of pixels (region).

• Central $(j,k)^{th}$ moment defined as:

$$\mu_{jk} = \sum_{(x,y) \in S} (x - \bar{x})^j (y - \bar{y})^k$$

• Invariant to translation of $S$.

• Interpretation:
  - $0^{th}$ central moment: area
  - $2^{nd}$ central moment: variance
  - $3^{rd}$ central moment: skewness
  - $4^{th}$ central moment: kurtosis
Moment Invariants ("Hu Moments")

- Normalized central moments

\[ \eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma}, \quad \gamma = \frac{p + q}{2} + 1 \]

- From those, a set of invariant moments can be defined for object description.

\[ \phi_1 = \eta_{20} + \eta_{02} \]
\[ \phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \]
\[ \phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \]
\[ \phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \]

- Robust to translation, rotation & scaling, but don’t expect wonders (still summary statistics).
Moment Invariants

\[
\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2]
+ (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\]

\[
\phi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
+ 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})
\]

\[
\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2]
+ (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\]

Often better to use \( \log_{10}(\phi_i) \) instead of \( \phi_i \) directly...
Axis of Least Second Moment

• Invariance to orientation?
  - Need a common alignment

  ![Image of alignment]

  Axis for which the squared distance to 2D object points is *minimized* (maximized).

  Compute Eigenvectors of 2\textsuperscript{nd} moment matrix (Matlab: eig(A))

  \[
  \begin{bmatrix}
  \mu_{20} & \mu_{11} \\
  \mu_{11} & \mu_{02}
  \end{bmatrix}
  = VDV^T
  = \begin{bmatrix}
  v_{11} & v_{12} \\
  v_{21} & v_{22}
  \end{bmatrix}
  \begin{bmatrix}
  \lambda_1 & 0 \\
  0 & \lambda_2
  \end{bmatrix}
  \begin{bmatrix}
  v_{11} \\
  v_{21}
  \end{bmatrix}
  \]

B. Leibe
Summary: Binary Image Processing

• Pros
  - Fast to compute, easy to store
  - Simple processing techniques
  - Can be very useful for constrained scenarios

• Cons
  - Hard to get “clean” silhouettes
  - Noise is common in realistic scenarios
  - Can be too coarse a representation
  - Cannot deal with 3D changes
Demo “Haribo Classification”

Code will be available on L2P…
You Can Do It At Home...

Accessing a webcam in Matlab:

```matlab
function out = webcam
% uses "Image Acquisition Toolbox",
adaptorName = 'winvideo';
vidFormat = 'I420_320x240';
vidObj1= videoinput(adaptorName, 1, vidFormat);
set(vidObj1, 'ReturnedColorSpace', 'rgb');
set(vidObj1, 'FramesPerTrigger', 1);
out = vidObj1 ;

cam = webcam();
img=getsnapshot(cam);
```
Course Outline

- Image Processing Basics
  - Image Formation
  - Binary Image Processing
  - Linear Filters
  - Edge & Structure Extraction
  - Color

- Segmentation

- Local Features & Matching

- Object Recognition and Categorization

- 3D Reconstruction

- Motion and Tracking

B. Leibe
Motivation

- Noise reduction/image restoration
- Structure extraction
Topics of This Lecture

- **Linear filters**
  - What are they? How are they applied?
  - Application: smoothing
  - Gaussian filter
  - What does it *mean* to filter an image?

- **Nonlinear Filters**
  - Median filter

- **Multi-Scale representations**
  - How to properly rescale an image?

- **Filters as templates**
  - Correlation as template matching
Common Types of Noise

- **Salt & pepper noise**
  - Random occurrences of black and white pixels

- **Impulse noise**
  - Random occurrences of white pixels

- **Gaussian noise**
  - Variations in intensity drawn from a Gaussian ("Normal") distribution.

- **Basic Assumption**
  - *Noise is i.i.d. (independent & identically distributed)*
Gaussian Noise

\[
 \begin{align*}
 f(x, y) &= \overline{f}(x, y) + \eta(x, y) \\
 \eta(x, y) &\sim \mathcal{N}(\mu, \sigma)
\end{align*}
\]

\[
 \text{Ideal Image} \quad \text{Noise process} \quad \text{Gaussian i.i.d. ("white") noise:}
\]

\[
 \text{sigma} = \text{randn(size(im))}
\]

\[
 \text{output} = \text{im} + \text{noise}
\]

Image Source: Martial Hebert

Slide credit: Kristen Grauman

B. Leibe
First Attempt at a Solution

• Assumptions:
  - Expect pixels to be like their neighbors
  - Expect noise processes to be independent from pixel to pixel ("i.i.d. = independent, identically distributed")

• Let’s try to replace each pixel with an average of all the values in its neighborhood...
Moving Average in 2D

\[ F[x, y] \quad G[x, y] \]
Moving Average in 2D

\[ F[x, y] \]

\[ G[x, y] \]

Source: S. Seitz
Moving Average in 2D

\[ F[x, y] \]  \hspace{1cm}  \[ G[x, y] \]

Source: S. Seitz
Moving Average in 2D

\[ F[x, y] \]

\[ G[x, y] \]
**Moving Average in 2D**

\[
F[x, y] = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 90 & 0 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 90 & 0 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 90 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

\[
G[x, y] = \begin{bmatrix}
0 & 10 & 20 & 30 & 30 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]
Moving Average in 2D

\[
F(x, y)
\]

\[
G(x, y)
\]
Correlation Filtering

- Say the averaging window size is $2k+1 \times 2k+1$:

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

- Now generalize to allow different weights depending on neighboring pixel’s relative position:

$$G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v]$$

Loop over all pixels in neighborhood around image pixel $F[i, j]$
Correlation Filtering

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v]
\]

- This is called cross-correlation, denoted \( G = H \otimes F \)

- Filtering an image
  - Replace each pixel by a weighted combination of its neighbors.
  - The filter “kernel” or “mask” is the prescription for the weights in the linear combination.

Slide credit: Kristen Grauman
Convolution

- Convolution:
  - Flip the filter in both dimensions (bottom to top, right to left)
  - Then apply cross-correlation

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i - u, j - v]
\]

\[
G = H \ast F
\]

*Notation for convolution operator*
Correlation vs. Convolution

- **Correlation**
  \[
  G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v]F[i + u, j + v]
  \]
  
  \[G = H \otimes F\]

- **Convolution**
  \[
  G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v]F[i - u, j - v]
  \]
  
  \[G = H \star F\]

- **Note**
  - If \(H[-u,-v] = H[u,v]\), then correlation \(\equiv\) convolution.

Matlab:
- `filter2`
- `imfilter`

Matlab:
- `conv2`

Note the difference!
Shift Invariant Linear System

• Shift invariant:
  - Operator behaves the same everywhere, i.e. the value of the output depends on the pattern in the image neighborhood, not the position of the neighborhood.

• Linear:
  - Superposition: \( h \ast (f_1 + f_2) = (h \ast f_1) + (h \ast f_2) \)
  - Scaling: \( h \ast (kf) = k(h \ast f) \)
Properties of Convolution

- Linear & shift invariant
- Commutative: \( f \ast g = g \ast f \)
- Associative: \( (f \ast g) \ast h = f \ast (g \ast h) \)
  - Often apply several filters in sequence: \(((a \ast b_1) \ast b_2) \ast b_3)\)
  - This is equivalent to applying one filter: \(a \ast (b_1 \ast b_2 \ast b_3)\)
- Identity: \( f \ast e = f \)
  - for unit impulse \( e = [\ldots, 0, 0, 1, 0, 0, \ldots] \).
- Differentiation: \( \frac{\partial}{\partial x} (f \ast g) = \frac{\partial f}{\partial x} \ast g \)
Averaging Filter

- What values belong in the kernel $H[u,v]$ for the moving average example?

\[ F[x, y] \ast H[u, v] = G[x, y] \]

\[ G = H \ast F \]

"box filter"
Smoothing by Averaging

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

“Ringing” artifacts!

Original

Filtered

Slide credit: Kristen Grauman

Image Source: Forsyth & Ponce
Smoothing with a Gaussian

Original

Filtered

Image Source: Forsyth & Ponce
Smoothing with a Gaussian - Comparison

Original

Filtered

Image Source: Forsyth & Ponce
Gaussian Smoothing

- Gaussian kernel
  \[ G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \]

- Rotationally symmetric

- Weights nearby pixels more than distant ones
  - This makes sense as 'probabilistic' inference about the signal

- A Gaussian gives a good model of a fuzzy blob
**Gaussian Smoothing**

- What parameters matter here?
- **Variance** $\sigma$ of Gaussian
  - Determines extent of smoothing

\[ \sigma = 2 \text{ with 30x30 kernel} \]
\[ \sigma = 5 \text{ with 30x30 kernel} \]
Gaussian Smoothing

- What parameters matter here?
- **Size** of kernel or mask
  - Gaussian function has infinite support, but discrete filters use finite kernels
  - Rule of thumb: set filter half-width to about $3\sigma$!

$$\sigma = 5 \text{ with } 10 \times 10 \text{ kernel}$$

$$\sigma = 5 \text{ with } 30 \times 30 \text{ kernel}$$

Slide credit: Kristen Grauman
Gaussian Smoothing in Matlab

\[
\begin{align*}
\text{>> } & \text{ hsize} = 10; \\
\text{>> } & \text{ sigma} = 5; \\
\text{>> } & \text{ h = fspecial('gaussian' hsize, sigma);} \\
\text{>> } & \text{ mesh(h);} \\
\text{>> } & \text{ imagesc(h);} \\
\text{>> } & \text{ outim = imfilter(im, h);} \\
\text{>> } & \text{ imshow(outim);} \\
\end{align*}
\]
Effect of Smoothing

More noise →

\[ \sigma = 0.05 \quad \sigma = 0.1 \quad \sigma = 0.2 \]

- Wider smoothing kernel
- No smoothing

Slide credit: Kristen Grauman

Image Source: Forsyth & Ponce
Efficient Implementation

• Both, the BOX filter and the Gaussian filter are separable:
  - First convolve each row with a 1D filter
    \[ g(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \]
  - Then convolve each column with a 1D filter
    \[ g(y) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{y^2}{2\sigma^2}\right) \]

• Remember:
  - Convolution is linear - associative and commutative
    \[ g_x \ast g_y \ast I = g_x \ast (g_y \ast I) = (g_x \ast g_y) \ast I \]
Filtering: Boundary Issues

• What is the size of the output?

• **MATLAB:** `filter2(g, f, shape)`
  - `shape = 'full'`: output size is sum of sizes of `f` and `g`
  - `shape = 'same'`: output size is same as `f`
  - `shape = 'valid'`: output size is difference of sizes of `f` and `g`
Filtering: Boundary Issues

- How should the filter behave near the image boundary?
  - The filter window falls off the edge of the image
  - Need to extrapolate
  - Methods:
    - Clip filter (black)
    - Wrap around
    - Copy edge
    - Reflect across edge

Source: S. Marschner
Filtering: Boundary Issues

- How should the filter behave near the image boundary?
  - The filter window falls off the edge of the image
  - Need to extrapolate
  - Methods (MATLAB):
    - Clip filter (black): `imfilter(f,g,0)`
    - Wrap around: `imfilter(f,g,'circular')`
    - Copy edge: `imfilter(f,g,'replicate')`
    - Reflect across edge: `imfilter(f,g,'symmetric')`
Topics of This Lecture

• Linear filters
  - What are they? How are they applied?
  - Application: smoothing
  - Gaussian filter
  - What does it mean to filter an image?

• Nonlinear Filters
  - Median filter

• Multi-Scale representations
  - How to properly rescale an image?

• Filters as templates
  - Correlation as template matching
Why Does This Work?

- A small excursion into the Fourier transform to talk about spatial frequencies...

\[ 3 \cos(x) + 1 \cos(3x) + 0.8 \cos(5x) + 0.4 \cos(7x) + \ldots \]

Source: Michal Irani
The Fourier Transform in Cartoons

- A small excursion into the Fourier transform to talk about spatial frequencies...

\[ 3 \cos(x) + 1 \cos(3x) + 0.8 \cos(5x) + 0.4 \cos(7x) + \ldots \]

Frequency spectrum

“high”  “low”  “high”

Frequency coefficients

Source: Michal Irani
Fourier Transforms of Important Functions

• Sine and cosine transform to...

Image Source: S. Chenney
Fourier Transforms of Important Functions

- Sine and cosine transform to “frequency spikes”

- A Gaussian transforms to...
Fourier Transforms of Important Functions

• Sine and cosine transform to “frequency spikes”

• A Gaussian transforms to a Gaussian

• A box filter transforms to...

B. Leibe

Image Source: S. Chenney
Fourier Transforms of Important Functions

- Sine and cosine transform to “frequency spikes”

- A Gaussian transforms to a Gaussian

- A box filter transforms to a sinc

All of this is symmetric!

\[ \text{sinc}(x) = \frac{\sin x}{x} \]
Duality

• The better a function is localized in one domain, the worse it is localized in the other.

• This is true for any function
Effect of Convolution

- Convolving two functions in the image domain corresponds to taking the product of their transformed versions in the frequency domain.

\[ f \star g \rightarrow \mathcal{F} \cdot G \]

- This gives us a tool to manipulate image spectra.
  - A filter attenuates or enhances certain frequencies through this effect.
Effect of Filtering

- Noise introduces high frequencies. To remove them, we want to apply a “low-pass” filter.

- The ideal filter shape in the frequency domain would be a box. But this transfers to a spatial sinc, which has infinite spatial support.

- A compact spatial box filter transfers to a frequency sinc, which creates artifacts.

- A Gaussian has compact support in both domains. This makes it a convenient choice for a low-pass filter.
Low-Pass vs. High-Pass

Original image

Low-pass filtered

High-pass filtered

Image Source: S. Chenney
Quiz: What Effect Does This Filter Have?

Source: D. Lowe
Sharpening Filter

Original

Sharpening filter
– Accentuates differences with local average

Source: D. Lowe
Sharpening Filter

before

after

B. Leibe

Source: D. Lowe
Application: High Frequency Emphasis

Original

High pass Filter

High Frequency Emphasis

High Frequency Emphasis + Histogram Equalization

Slide credit: Michal Irani
Topics of This Lecture

• **Linear filters**
  - What are they? How are they applied?
  - Application: smoothing
  - Gaussian filter
  - What does it *mean* to filter an image?

• **Nonlinear Filters**
  - Median filter

• **Multi-Scale representations**
  - How to properly rescale an image?

• **Image derivatives**
  - How to compute gradients robustly?
Non-Linear Filters: Median Filter

• Basic idea
  - Replace each pixel by the median of its neighbors.

• Properties
  - Doesn’t introduce new pixel values
  - Removes spikes: good for impulse, salt & pepper noise
  - Linear?

Slide credit: Kristen Grauman
Median Filter

Salt and pepper noise

Plots of a row of the image
Median Filter

- The Median filter is **edge preserving**.

<table>
<thead>
<tr>
<th>INPUT</th>
<th>MEDIAN</th>
<th>MEAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>..</td>
<td>...</td>
<td>......</td>
</tr>
<tr>
<td>..</td>
<td>...</td>
<td>......</td>
</tr>
<tr>
<td>..</td>
<td>...</td>
<td>......</td>
</tr>
</tbody>
</table>
Median vs. Gaussian Filtering

Gaussian

Median

Slide credit: Svetlana Lazebnik
Topics of This Lecture

- **Linear filters**
  - What are they? How are they applied?
  - Application: smoothing
  - Gaussian filter
  - What does it *mean* to filter an image?

- **Nonlinear Filters**
  - Median filter

- **Multi-Scale representations**
  - How to properly rescale an image?

- **Filters as templates**
  - Correlation as template matching
Motivation: Fast Search Across Scales

B. Leibe

Image Source: Irani & Basri
Image Pyramid

Low resolution

High resolution
How Should We Go About Resampling?

Let’s resample the checkerboard by taking one sample at each circle.

In the top left board, the new representation is reasonable. Top right also yields a reasonable representation.

Bottom left is all black (dubious) and bottom right has checks that are too big.
Fourier Interpretation: Discrete Sampling

- Sampling in the spatial domain is like multiplying with a spike function.

- Sampling in the frequency domain is like...

\[ ? \]
Fourier Interpretation: Discrete Sampling

- Sampling in the spatial domain is like multiplying with a spike function.

- Sampling in the frequency domain is like convolving with a spike function.

Source: S. Chenney
Sampling and Aliasing

Signal → Fourier Transform → Magnitude Spectrum

Sample → Sampled Signal → Fourier Transform → Magnitude Spectrum

Copy and Shift

Accurately Reconstructed Signal → Inverse Fourier Transform → Magnitude Spectrum

Cut out by multiplication with box filter
Sampling and Aliasing

- Nyquist theorem:
  - In order to recover a certain frequency $f$, we need to sample with at least $2f$.
  - This corresponds to the point at which the transformed frequency spectra start to overlap (the Nyquist limit).

Image Source: Forsyth & Ponce
Sampling and Aliasing

- Signal
- Fourier Transform
- Magnitude Spectrum
- Sample
- Copy and Shift
- Sampled Signal
- Fourier Transform
- Magnitude Spectrum
- Inaccurately Reconstructed Signal
- Inverse Fourier Transform
- Magnitude Spectrum
- Cut out by multiplication with box filter

Image Source: Forsyth & Ponce
Aliasing in Graphics

Disintegrating textures
Resampling with Prior Smoothing

- Note: We cannot recover the high frequencies, but we can avoid artifacts by smoothing before resampling.

B. Leibe

Image Source: Forsyth & Ponce
The Gaussian Pyramid

\[ G_0 = \text{Image} \]

\[ G_1 = (G_0 \ast \text{gaussian}) \downarrow 2 \]

\[ G_2 = (G_1 \ast \text{gaussian}) \downarrow 2 \]

\[ G_3 = (G_2 \ast \text{gaussian}) \downarrow 2 \]

\[ G_4 = (G_3 \ast \text{gaussian}) \downarrow 2 \]

Source: Irani & Basri
Gaussian Pyramid - Stored Information

All the extra levels add very little overhead for memory or computation!
Summary: Gaussian Pyramid

- **Construction**: create each level from previous one
  - Smooth and sample

- **Smooth with Gaussians**, in part because
  - a Gaussian*Gaussian = another Gaussian
  - $G(\sigma_1) * G(\sigma_2) = G(\sqrt{\sigma_1^2 + \sigma_2^2})$

- Gaussians are low-pass filters, so the representation is redundant once smoothing has been performed.
  - $\Rightarrow$ There is no need to store smoothed images at the full original resolution.
The Laplacian Pyramid

Gaussian Pyramid

\[ L_i = G_i - \text{expand}(G_{i+1}) \]

\[ G_i = L_i + \text{expand}(G_{i+1}) \]

Laplacian Pyramid

\[ L_n = G_n \]

Why is this useful?

Source: Irani & Basri
Laplacian ~ Difference of Gaussian

DoG = Difference of Gaussians
Cheap approximation - no derivatives needed.
Topics of This Lecture

• **Linear filters**
  - What are they? How are they applied?
  - Application: smoothing
  - Gaussian filter
  - What does it *mean* to filter an image?

• **Nonlinear Filters**
  - Median filter

• **Multi-Scale representations**
  - How to properly rescale an image?

• **Filters as templates**
  - Correlation as template matching
Note: Filters are Templates

- Applying a filter at some point can be seen as taking a dot-product between the image and some vector.
- Filtering the image is a set of dot products.

- Insight
  - Filters look like the effects they are intended to find.
  - Filters find effects they look like.
Where’s Waldo?

Template

Scene

Slide credit: Kristen Grauman

B. Leibe
Where’s Waldo?

Detected template

Template

Slide credit: Kristen Grauman

B. Leibe
Where’s Waldo?

Detected template

Correlation map

Slide credit: Kristen Grauman
Correlation as Template Matching

- Think of filters as a dot product of the filter vector with the image region
  - Now measure the angle between the vectors
    \[ a \cdot b = |a| |b| \cos \theta \]
    \[ \cos \theta = \frac{a \cdot b}{|a| |b|} \]
  - Angle (similarity) between vectors can be measured by normalizing length of each vector to 1.
Summary: Mask Properties

- **Smoothing**
  - Values positive
  - Sum to 1 $\Rightarrow$ constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove “high-frequency” components; “low-pass” filter

- **Filters act as templates**
  - Highest response for regions that “look the most like the filter”
  - Dot product as correlation
Summary Linear Filters

• Linear filtering:
  ➢ Form a new image whose pixels are a weighted sum of original pixel values

• Properties
  ➢ Output is a shift-invariant function of the input (same at each image location)

Examples:
• Smoothing with a box filter
• Smoothing with a Gaussian
• Finding a derivative
• Searching for a template

Pyramid representations
• Important for describing and searching an image at all scales
References and Further Reading

- Background information on linear filters and their connection with the Fourier transform can be found in Chapters 7 and 8 of