Filters

CMP 719 – Computer Vision
Pinar Duygulu
Hacettepe University
Today’s topics

• Image Formation

• Image filters in spatial domain
  – Filter is a mathematical operation of a grid of numbers
  – Smoothing, sharpening, measuring texture

• Image filters in the frequency domain
  – Filtering is a way to modify the frequencies of images
  – Denoising, sampling, image compression

• Templates and Image Pyramids
  – Filtering is a way to match a template to the image
  – Detection, coarse-to-fine registration
Images as functions

Source: S. Seitz
Images as functions

• We can think of an image as a function, $f$, from $\mathbb{R}^2$ to $\mathbb{R}$:
  • $f(x, y)$ gives the intensity at position $(x, y)$
  • Realistically, we expect the image only to be defined over a rectangle, with a finite range:
    – $f: [a,b] \times [c,d] \rightarrow [0, 255]$  

• A color image is just three functions pasted together. We can write this as a “vector-valued” function:

$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$

Source: S. Seitz
Digital images

- In computer vision we operate on **digital (discrete)** images:
  - **Sample** the 2D space on a regular grid
  - **Quantize** each sample (round to nearest integer)
- Image thus represented as a matrix of integer values.

Adapted from S. Seitz
Images as discrete functions

- Cartesian Coordinates

\[
f[n, m] = \begin{bmatrix}
\vdots & \vdots & \vdots \\
\cdots & f[-1, 1] & f[0, 1] & f[1, 1] \\
\cdots & f[-1, 0] & f[0, 0] & f[1, 0] \\
f[-1, -1] & f[0, -1] & f[1, -1] & \cdots \\
\vdots & \vdots & \vdots & \cdots
\end{bmatrix}
\]

Source: Fei Feli Li, Stanford University
Today’s topics

• Image Formation

• Image filters in spatial domain
  – Filter is a mathematical operation of a grid of numbers
  – Smoothing, sharpening, measuring texture

• Image filters in the frequency domain
  – Filtering is a way to modify the frequencies of images
  – Denoising, sampling, image compression

• Templates and Image Pyramids
  – Filtering is a way to match a template to the image
  – Detection, coarse-to-fine registration
Zebras vs. Dalmatians

Both zebras and dalmatians have black and white pixels in about the same number
  – if we shuffle the images point-wise processing is not affected

Need to measure properties relative to small neighborhoods of pixels
  - find different image patterns
We want to remove unwanted sources of variation, and keep the information relevant for whatever task we need to solve.
Filters

• **Filtering:**
  – Form a new image whose pixels are a combination of original pixel values
  - compute function of local neighborhood at each position

• **Goals:**
  • Extract useful information from the images
    Features (textures, edges, corners, distinctive points, blobs...)
  • Modify or enhance image properties:
    super-resolution; in-painting; de-noising, resizing
  • Detect patterns
    Template matching

Source: Fei Feli Li, Stanford University; James Hays, Brown
Smooth/Sharpen Images...  Find edges...  Find waldo...

Source: Darrell, Berkeley
De-noising

Salt and pepper noise

Super-resolution

In-painting

Source: Fei Fei Li, Stanford University
Common types of noise

- **Salt and 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

Source: Darrell, Berkeley

Source: S. Seitz
Gaussian noise

$$f(x, y) = \hat{f}(x, y) + \eta(x, y)$$

Gaussian i.i.d. ("white") noise:
$$\eta(x, y) \sim N(\mu, \sigma)$$

```matlab
>> noise = randn(size(im)).*sigma;
```

```matlab
>> output = im + noise;
```
First attempt at a solution

• Let’s replace each pixel with an average of all the values in its neighborhood

• Assumptions:
  – Expect pixels to be like their neighbors
  – Expect noise processes to be independent from pixel to pixel

Source: Darrell, Berkeley
First attempt at a solution

- Let’s replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:

Source: Darrell, Berkeley
Source: S. Marschner
Weighted Moving Average

• Can add weights to our moving average
• *Weights* $[1, 1, 1, 1, 1] / 5$
Weighted Moving Average

• Non-uniform weights \([1, 4, 6, 4, 1] / 16\)

Source: Darrell, Berkeley
Source: S. Marschner
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]

Source: Darrell, Berkeley

Source: S. Seitz
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]

Source: S. Seitz

Source: Darrell, Berkeley
Correlation filtering

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

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

Attribute uniform weight Loop over all pixels in neighborhood around to each pixel image pixel $F[i,j]$

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

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

Non-uniform weights

Source: Darrell, Berkeley
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 with a linear combination of its neighbors.

The filter “kernel” or “mask” \(H[u, v]\) is the prescription for the weights in the linear combination.

Source: Darrell, Berkeley
Averaging Filter

original

adapted from Darrell and Freeman, MIT
Averaging Filter

adapted from Darrell and Freeman, MIT
Averaging Filter

adapted from Darrell and Freeman, MIT
Averaging Filter

adapted from Darrell and Freeman, MIT
Averaging filter

• What values belong in the kernel $H$ for the moving average example?

$$G = H \otimes F$$

Source: Darrell, Berkeley
Smoothing by averaging

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

Source: Darrell, Berkeley
Example

Source: Martial Hebert, CMU
Example

Source: Martial Hebert, CMU
Example

Source: Martial Hebert, CMU
Smoothing by averaging

Source: Martial Hebert, CMU
Smoothing by averaging

Source: Martial Hebert, CMU
Smoothing by averaging
Smoothing by averaging

Source: Martial Hebert, CMU
Gaussian filter

• What if we want nearest neighboring pixels to have the most influence on the output?

A weighted average that weights pixels at its center much more strongly than its boundaries.

This kernel is an approximation of a Gaussian function:

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

Source: S. Seitz

Source: Darrell, Berkeley
Smoothing with a Gaussian

Source: Darrell, Berkeley
Smoothing with a Gaussian

Source: Martial Hebert, CMU
Smoothing with a Gaussian

Result of blurring using a uniform local model

Produces a set of narrow vertical, horizontal, and vertical bars – ringing effect

Result of blurring using a set of Gaussian weights

Source: David Forsyth, UIUC
Smoothing with a Gaussian

Source: Martial Hebert, CMU
Gaussian filters

• What parameters matter here?

• **Size** of kernel or mask
  
  – Note, Gaussian function has infinite support, but discrete filters use finite kernels

\[ \sigma = 5 \text{ with } 10 \times 10 \text{ kernel} \quad \text{and} \quad \sigma = 5 \text{ with } 30 \times 30 \text{ kernel} \]

Source: Darrell, Berkeley
Gaussian filters

- What parameters matter here?
- **Variance** of Gaussian: determines extent of smoothing

\[ \sigma = 2 \text{ with } 30 \times 30 \text{ kernel} \]

\[ \sigma = 5 \text{ with } 30 \times 30 \text{ kernel} \]

Source: Darrell, Berkeley
Smoothing with a Gaussian

If \( \sigma \) is small: the smoothing will have little effect.

If \( \sigma \) is larger: neighboring pixels will have larger weights resulting in consensus of the neighbors.

If \( \sigma \) is very large: details will disappear along with the noise.

Source: Martial Hebert, CMU
Gaussian filter

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

Source: Torralba, MIT
Gaussian smoothing to remove noise

\[ f(x, y) = \widehat{f}(x, y) + \eta(x, y) \]

Gaussian i.i.d. ("white") noise:
\[ \eta(x, y) \sim N(\mu, \sigma) \]

Source: Martial Hebert, CMU
Gaussian smoothing to remove noise

Source: Martial Hebert, CMU
Smoothing with a Gaussian

The effects of smoothing
Each row shows smoothing with gaussians of different width; each column shows different realizations of an image of gaussian noise.

Source: David Forsyth, UIUC
Smoothing with a Gaussian

• Filtered noise is sometimes useful
  – looks like some natural textures, can be used to simulate fire, etc.

Source: David Forsyth, UIUC
Gaussian kernel

\[ g(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x^2 + y^2)}{2\sigma^2}\right) \]

Gaussian is an approximation to the binomial distribution.

Can approximate Gaussian using binomial coefficients.

\[ a_{nr} \equiv \frac{n!}{r!(n-r)!} \equiv \binom{n}{r} \]

1X3 filter: \( n=(3-1)=2, r=0,1,2 \)

\[
\begin{array}{ccc}
1 & 2 & 1 \\
0.0625 & 0.1250 & 0.0625 \\
0.1250 & 0.2500 & 0.1250 \\
0.0625 & 0.1250 & 0.0625 \\
\end{array}
\]

Source: from Michael Black
Matlab

>> hsize = 10;
>> sigma = 5;
>> h = fspecial('gaussian' hsize, sigma);

>> mesh(h);

>> imagesc(h);

>> outim = imfilter(im, h);
>> imshow(outim);

Source: Darrell, Berkeley
Smoothing with a Gaussian

Parameter $\sigma$ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.

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

Source: Darrell, Berkeley
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

Source: Darrell, Berkeley
Convolution vs. correlation

Convolution

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

\[ G = H \ast F \]

Cross-correlation

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

\[ G = H \otimes F \]

For a Gaussian or box filter, how will the outputs differ?

If the input is an impulse signal, how will the outputs differ?

Source: Darrell, Berkeley
Predict the filtered outputs

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{bmatrix} \ast 0 = ?
\]

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0 \\
\end{bmatrix} \ast 0 = ?
\]

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{bmatrix} \ast \frac{1}{9} \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix} = ?
\]

Source: Darrell, Berkeley
Practice with linear filters

original

?
Practice with linear filters

adapted from Darrell and Freeman, MIT
Practice with linear filters

Original

Filtered (no change)

Source: D. Lowe

Source: Darrell, Berkeley
Impulse

\[ f[m, n] = I \otimes g = \sum_{k, l} h[m - k, n - l]g[k, l] \]

Source: Torralba, MIT
Practice with linear filters

Original

Source: Darrell, Berkeley

Source: D. Lowe
Practice with linear filters

Original

Shifted left by 1 pixel with correlation

Source: Darrell, Berkeley

Source: D. Lowe
**Shifts**

\[ f[m,n] = I \otimes g = \sum_{k,l} h[m-k, n-l] g[k, l] \]
Practice with linear filters

adapted from Darrell and Freeman, MIT
Practice with linear filters

adapted from Darrell and Freeman, MIT
Practice with linear filters

adapted from Darrell and Freeman, MIT
Sharpening

adapted from Darrell and Freeman, MIT
Sharpening

adapted from Darrell and Freeman, MIT
Practice with linear filters

Original

Sharpening filter
- Accentuates differences with local average

Source: Darrell, Berkeley
Source: D. Lowe
Filtering examples: sharpening

before

after

Source: Darrell, Berkeley
Rectangular filter

\[ g[m,n] \ast h[m,n] = f[m,n] \]

Source: Torralba, MIT
What does blurring take away?

- Let’s add it back:
Rectangular filter

\[ g[m,n] \odot h[m,n] = f[m,n] \]

Source: Torralba, MIT
Rectangular filter

\[ g[m,n] \otimes h[m,n] = f[m,n] \]

Source: Torralba, MIT
Integral image

Source: Torralba, MIT
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: $h \ast (k \, f) = k \, (h \ast f)$

Source: Darrell, Berkeley
Properties of convolution

• Linear & shift invariant

• Commutative:
  
  \[ f \ast g = g \ast f \]

• Associative
  
  \[(f \ast g) \ast h = f \ast (g \ast h)\]

• Identity:
  
  unit impulse \( e = [..., 0, 0, 1, 0, 0, ...] \). \( f \ast e = f \)

• Differentiation:
  
  \[
  \frac{\partial}{\partial x} (f \ast g) = \frac{\partial f}{\partial x} \ast g
  \]

Source: Darrell, Berkeley
Separability

- In some cases, filter is separable, and we can factor into two steps:
  - Convolve all rows
  - Convolve all columns

Source: Darrell, Berkeley
Separability

• In some cases, filter is separable, and we can factor into two steps: e.g.,

\[ f \ast (g \ast h) = (f \ast g) \ast h \]

What is the computational complexity advantage for a separable filter of size \( k \times k \), in terms of number of operations per output pixel?

Source: Darrell, Berkeley
Advantages of separability

First convolve the image with a one dimensional horizontal filter
Then convolve the result of the first convolution with a one dimensional vertical filter
For a k x k Gaussian filter, 2D convolution requires $k^2$ operations per pixel
But using the separable filters, we reduce this to 2k operations per pixel.

adapted from Larry Davis, University of Maryland
Seperable Gaussian

\[ g(x) = \frac{1}{\sqrt{2\pi \sigma}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \]

\[ g(y) = \frac{1}{\sqrt{2\pi \sigma}} \exp\left(-\frac{y^2}{2\sigma^2}\right) \]

Product?

\[ g(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \]
Advantages of Gaussians

- Convolution of a Gaussian with itself is another Gaussian
  - so we can first smooth an image with a small Gaussian
  - then, we convolve that smoothed image with another small Gaussian and the result is equivalent to smoother the original image with a larger Gaussian.
  - If we smooth an image with a Gaussian having sd $\sigma$ twice, then we get the same result as smoothing the image with a Gaussian having standard deviation $(2\sigma)^{1/2}$

adapted from Larry Davis, University of Maryland
Effect of smoothing filters

5x5

Additive Gaussian noise

Salt and pepper noise

Source: Darrell, Berkeley
Median filter

- No new pixel values introduced
- Removes spikes: good for impulse, salt & pepper noise

Source: Darrell, Berkeley
Median filter

Salt and pepper noise

Median filtered

Plots of a row of the image

Source: M. Hebert

Source: Darrell, Berkeley
Median filter

- Median filter is edge preserving

Source: Darrell, Berkeley
Boundary issues

• What is the size of the output?
• MATLAB: \texttt{filter2(g, f, shape)}
  – \texttt{shape = ‘full’}: output size is sum of sizes of \( f \) and \( g \)
  – \texttt{shape = ‘same’}: output size is same as \( f \)
  – \texttt{shape = ‘valid’}: output size is difference of sizes of \( f \) and \( g \)
Boundary issues

• What about near the edge?
  – 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
Boundary issues

• What about near the edge?
  – the filter window falls off the edge of the image
  – need to extrapolate
  – methods (MATLAB):
    • clip filter (black): \texttt{imfilter}(f, g, 0)
    • wrap around: \texttt{imfilter}(f, g, ‘circular’)
    • copy edge: \texttt{imfilter}(f, g, ‘replicate’)
    • reflect across edge: \texttt{imfilter}(f, g, ‘symmetric’)

Source: Darrell, Berkeley
Source: S. Marschner
Borders

Source: Torralba, MIT
Today’s topics

• Image Formation

• Image filters in spatial domain
  – Filter is a mathematical operation of a grid of numbers
  – Smoothing, sharpening, measuring texture

• Image filters in the frequency domain
  – Filtering is a way to modify the frequencies of images
  – Denoising, sampling, image compression

• Templates and Image Pyramids
  – Filtering is a way to match a template to the image
  – Detection, coarse-to-fine registration
Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

Source: James Hays, Brown
Why does a lower resolution image still make sense to us? What do we lose?
Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807): Any univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.

• Don’t believe it?
  – Neither did Lagrange, Laplace, Poisson and other big wigs
  – Not translated into English until 1878!

• But it’s (mostly) true!
  – called Fourier Series
  – there are some subtle restrictions

...the manner in which the author arrives at these equations is not exempt of difficulties and...his analysis to integrate them still leaves something to be desired on the score of generality and even rigour.

Source: James Hays, Brown
A sum of sines

Our building block:

\[ A \sin(\omega x + \phi) \]

Add enough of them to get any signal \( f(x) \) you want!

\[ f(\text{target}) = f_1 + f_2 + f_3 + \ldots + f_n + \ldots \]

Source: James Hays, Brown
Filtering in spatial domain

\[
\begin{bmatrix}
1 & 0 & -1 \\
2 & 0 & -2 \\
1 & 0 & -1 \\
\end{bmatrix}
\]

Source: James Hays, Brown
Filtering in frequency domain

Slide: Hoiem

Source: James Hays, Brown
Filtering

Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

Source: James Hays, Brown
Gaussian

Source: James Hays, Brown
Box Filter

Source: James Hays, Brown
Subsampling by a factor of 2

Throw away every other row and column to create a 1/2 size image

Source: James Hays, Brown
Aliasing problem

• 1D example (sinewave):

Source: James Hays, Brown

Source: S. Marschner
Aliasing problem

• 1D example (sinewave):
Subsampling without pre-filtering

1/2

1/4  (2x zoom)

1/8  (4x zoom)

Source: James Hays, Brown
Subsampling with Gaussian pre-filtering

Gaussian 1/2  G 1/4  G 1/8

Source: James Hays, Brown
Today’s topics

• Image Formation

• Image filters in spatial domain
  – Filter is a mathematical operation of a grid of numbers
  – Smoothing, sharpening, measuring texture

• Image filters in the frequency domain
  – Filtering is a way to modify the frequencies of images
  – Denoising, sampling, image compression

• Templates and Image Pyramids
  – Filtering is a way to match a template to the image
  – Detection, coarse-to-fine registration
Template matching

Scene

Template (mask)

A toy example

Source: Darrell, Berkeley
Template matching

Detected template

Template

Source: Darrell, Berkeley
Template matching

Detected template

Correlation map

Source: Darrell, Berkeley
Where’s Waldo?

Scene

Template

Source: Darrell, Berkeley
Where’s Waldo?

Scene

Template

Source: Darrell, Berkeley
Where’s Waldo?

Detected template

Correlation map

Source: Darrell, Berkeley
Template matching

What if the template is not identical to some subimage in the scene?

Source: Darrell, Berkeley
Template matching

Match can be meaningful, if scale, orientation, and general appearance is right.

Source: Darrell, Berkeley
Application

copyright 1998, IEEE
Template matching

• Goal: find \( \begin{array}{c} \text{in image} \end{array} \)

• Main challenge: What is a good similarity or distance measure between two patches?
  – Correlation
  – Zero-mean correlation
  – Sum Square Difference
  – Normalized Cross Correlation

Source: Hays, Brown
Matching with filters

• Goal: find eye in image

• Method 0: filter the image with eye patch

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l]$$

What went wrong?

response is stronger for higher intensity

Source: Hays, Brown
Matching with filters

• Goal: find 🌟 in image

• Method 1: filter the image with zero-mean eye

\[ h[m,n] = \sum_{k,l} (f[k,l] - \bar{f})(g[m+k, n+l]) \]
Matching with filters

• Goal: find \( \begin{array}{c} \text{in image} \end{array} \)

• Method 2: SSD

\[
h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2
\]

Source: Hays, Brown
Matching with filters

- **Goal:** find \( \text{\[image\]} \) in image

- **Method 2: SSD**

  \[
  h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2
  \]

What's the potential downside of SSD?

SSD is sensitive to average intensity.
Matching with filters

• Goal: find \( \text{eye} \) in image

• Method 3: Normalized cross-correlation

\[
h[m, n] = \left( \frac{\sum_{k,l} (g[k,l] - \bar{g})(f[m-k,n-l] - \bar{f}_{m,n})}{\left( \sum_{k,l} (g[k,l] - \bar{g})^2 \right)^{0.5} \left( \sum_{k,l} (f[m-k,n-l] - \bar{f}_{m,n})^2 \right)^{0.5}} \right)
\]

Matlab: `normxcorr2(template, im)`

Source: Hays, Brown
Matching with filters

• Goal: find \(\bigcirc\) in image

• Method 3: Normalized cross-correlation

Source: Hays, Brown
Matching with filters

• Goal: find 🡱 in image

• Method 3: Normalized cross-correlation

Source: Hays, Brown
Q: What is the best method to use?

A: Depends

• SSD: faster, sensitive to overall intensity
• Normalized cross-correlation: slower, invariant to local average intensity and contrast

Source: Hays, Brown
Q: What if we want to find larger or smaller eyes?

Motivation for studying scale.
A: Image Pyramid

adapted from Michael Black, Brown University
Review of Sampling

Image ➔ Gaussian Filter ➔ Low-Pass Filtered Image ➔ Sample ➔ Low-Res Image

Source: Hays, Brown
Gaussian Pyramid

\[ G_4 = (G_3 \ast \text{gaussian}) \downarrow 2 \]
\[ G_3 = (G_2 \ast \text{gaussian}) \downarrow 2 \]
\[ G_2 = (G_1 \ast \text{gaussian}) \downarrow 2 \]
\[ G_1 = (G_0 \ast \text{gaussian}) \downarrow 2 \]

adapted from Michael Black, Brown University
Gaussian pyramid

512  256  128  64  32  16  8

Source: Hays, Brown

Source: Forsyth
Template Matching with Image Pyramids

Input: Image, Template
1. Match template at current scale
2. Downsample image
3. Repeat 1-2 until image is very small
4. Take responses above some threshold, perhaps with non-maxima suppression

Source: Hays, Brown
Coarse-to-fine Image Registration

1. Compute Gaussian pyramid
2. Align with coarse pyramid
3. Successively align with finer pyramids
   - Search smaller range

Why is this faster?

Are we guaranteed to get the same result?

Source: Hays, Brown
Laplacian filter

Source: Hays, Brown

Source: Lazebnik
Laplacian pyramid

Source: Hays, Brown

Source: Forsyth
Computing Gaussian/Laplacian Pyramid

Can we reconstruct the original from the laplacian pyramid?

Source: Hays, Brown

Texture segmentation

Clues from Human Perception

- Early processing in humans filters for various orientations and scales of frequency
- Perceptual cues in the mid-high frequencies dominate perception
- When we see an image from far away, we are effectively subsampling it

Early Visual Processing: Multi-scale edge and blob filters

Source: Hays, Brown