Lecture 10: Pyramids and Scale Space

Recall

• Cascaded Gaussians
  – Repeated convolution by a smaller Gaussian to simulate effects of a larger one.
  
• $G^*(G*f) = (G*G)*f$ [associativity]

\[
G_{\sigma_1} * G_{\sigma_2} = G_{\sigma} \quad \sigma^2 = \sigma_1^2 + \sigma_2^2
\]

Example: Cascaded Convolutions

\[
\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 1 \\ 1 & 3 & 3 & 1 \end{bmatrix}
\]

\[
\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 1 & 3 & 3 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 3 & 3 & 1 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}
\]

\[
\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 3 & 3 & 1 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 1 & 5 & 10 & 5 & 1 \end{bmatrix}
\]

..and so on…

Pascal’s Triangle

Aside: Binomial Approximation

\[
a_r = \frac{n!}{r!(n-r)!} = \binom{n}{r}
\]

- $n$ = number of elements in the 1D filter minus 1
- $r$ = position of element in the filter kernel (0, 1, 2, …)

Pascal’s Triangle Coefficients

Aside: Binomial Approximation

Look at odd-length rows of Pascal’s triangle:

\[
\begin{bmatrix} 1 \end{bmatrix} * \begin{bmatrix} 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \end{bmatrix}
\]

\[
\begin{bmatrix} 1 & 2 & 1 \\ 1 & 3 & 3 & 1 \\ 1 & 4 & 6 & 4 & 1 \\ 1 & 5 & 10 & 5 & 1 \end{bmatrix}
\]

\[
\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} / 4 \approx \text{approximates Gaussian with } \sigma = 1/\sqrt{2}
\]

\[
\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} / 16 \approx \text{approximates Gaussian with } \sigma = 1
\]

An easy way to generate integer-coefficient Gaussian approximations.

From Homework 2

<table>
<thead>
<tr>
<th>Row</th>
<th>( \sigma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/2</td>
</tr>
<tr>
<td>2</td>
<td>1/\sqrt{2}</td>
</tr>
<tr>
<td>3</td>
<td>( \sigma )</td>
</tr>
<tr>
<td>4</td>
<td>( \sigma )</td>
</tr>
<tr>
<td>5</td>
<td>( \sigma )</td>
</tr>
<tr>
<td>6</td>
<td>( \sigma )</td>
</tr>
</tbody>
</table>

\[
G_{\sigma_1} * G_{\sigma_2} = G_{\sigma} \quad \sigma^2 = \sigma_1^2 + \sigma_2^2
\]
More about Cascaded Convolutions
(for the mathematically inclined)

Fun facts:
The distribution of the sum of two random variables
$X + Y$ is the convolution of their two distributions

Given $N$ i.i.d. random variables, $X_1 \ldots X_N$, the distribution
of their sum approaches a Gaussian distribution (aka the
central limit theorem)

Therefore:
The repeated convolution of a (nonnegative)
filter with itself takes on a Gaussian shape.

Gaussian Smoothing at
Different Scales

Idea for Today:
Form a Multi-Resolution Representation

Pyramid Representations

Because a large amount of smoothing limits
the frequency of features in the image, we do
not need to keep all the pixels around!

Strategy: progressively reduce the number of
pixels as we smooth more and more.

Leads to a “pyramid” representation if we
subsample at each level.
Gaussian Pyramid

- Synthesis: Smooth image with a Gaussian and downsample. Repeat.
- Gaussian is used because it is self-reproducing (enables incremental smoothing).
- Top levels come “for free”. Processing cost typically dominated by two lowest levels (highest resolution).

Emphasis: Smaller Images have Lower Resolution

Generating a Gaussian Pyramid

Basic Functions:
- Blur (convolve with Gaussian to smooth image)
- DownSample (reduce image size by half)
- Upsample (double image size)

Downsample

By the way: Subsampling is a bad idea unless you have previously blurred/smoothed the image! (because it leads to aliasing)
To Elaborate: Thumbnails

original image 262x195  
downsampling (left)

vs. smoothed then
downsampling (right)

131x97  
65x48  
32x24

original image 262x195  
downsampling (left)  
vs. smoothed then
downsampling (right)

131x97

original image 262x195  
downsampling (left)  
vs. smoothed then
downsampling (right)

65x48

original image 262x195  
downsampling (left)  
vs. smoothed then
downsampling (right)

32x24

Upsample

How to fill in the empty values?  
Interpolation:  
• initially set empty pixels to zero  
• convolve upsampling image with Gaussian filter!  
e.g. 5x5 kernel with sigma = 1.  
• Must also multiply by 4. Explain why.

Specific Example


General idea: cascaded filtering using \([1 4 6 4 1]\) kernel to generate a pyramid with two images per octave (power of 2 change in resolution). When we reach a full octave, downsample the image.
Basic idea: different scales are appropriate for describing different objects in the image, and we may not know the correct scale/size ahead of time.

Example: Detecting “Blobs” at Different Scales.

But first, we have to talk about detecting blobs at one scale...