More on pyramids....

Binomial distribution and Pascal’s Triangle

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

..and so on...

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

Pascal’s Triangle

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

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

Image from Robert Collins
Look at odd-length rows of Pascal’s triangle:

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

\[\frac{[1 2 1]}{4} \text{ - approximates Gaussian with } \sigma = \frac{1}{\sqrt{2}}\]

\[\frac{[1 4 6 4 1]}{16} \text{ - approximates Gaussian with } \sigma = 1\]

and so on...

An easy way to generate integer-coefficient Gaussian approximations.
\[ \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} / 16 \]

\[ \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} / 16 \times \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} / 16 \]
How to produce two sigma’s at every pyramid level

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.

**Effective σ**

- Image start
- \( g(n, σ = 1) \)
- Level 0
  - \( p_0(n, 0) \)
  - \( g(n, σ = 1) \)
  - \( σ = 1 \)
- Level 1
  - \( g(n, σ = 1) \)
  - \( p_0(n, 0) \)
  - \( g(n, σ = 1) ≠ g(n, σ = 1) \)
  - \( σ = √2 \)
- Level 2
  - \( g(n, σ = 1) \)
  - \( p_0(n, 0) \)
  - \( g(n, σ = 1) ≠ g(n, σ = 1) \)
  - \( σ = 2 \)
- Level 3
  - \( g(n, σ = 1) \)
  - \( p_0(n, 0) \)
  - \( g(n, σ = 1) ≠ g(n, σ = 1) \)
  - \( σ = 2\sqrt{2} \)
- Level 4
  - \( g(n, σ = 1) \)
  - \( p_0(n, 0) \)
  - \( g(n, σ = 1) ≠ g(n, σ = 1) \)
  - \( σ = 4 \)
Remember that LoG = DoG!

\[ \nabla^2 G(r, \sigma_{\text{lap}}) = \frac{r^2 - \sigma_{\text{lap}}^2}{\sigma_{\text{lap}}^5 \sqrt{2\pi}} e^{-\frac{1}{2} \frac{r^2}{\sigma_{\text{lap}}^2}} \]

The difference of Gaussians is:

\[ \text{DOG}(r, \sigma_{\text{dog}}) = \frac{1}{\sigma_1 \sqrt{2\pi}} e^{-\frac{1}{2} \frac{r^2}{\sigma_1^2}} - \frac{1}{\sigma_{\text{dog}} \sqrt{2\pi}} e^{-\frac{1}{2} \frac{r^2}{\sigma_{\text{dog}}^2}} \]

\( \sigma_1 = \sqrt{2}\sigma_{\text{dog}} \)
DoG vs LoG (Crowley, 2002)

Figure 2: Comparisons of real Laplacian versus real DoG and binomial DoG for $\sigma_{\text{dog}} = \sqrt{2}$ and $\sigma_1 = \sqrt{2}\sigma_{\text{dog}} = 2$ and $\sigma_{\text{lap}} = 1.7$
Effective $\sigma$ of the Laplacian as Diff of Gaussians
Reconstruction of the image from laplacian pyramid

- function [ img ] = pyrReconstruct( pyr )
- %PYRRECONSTRUCT Uses a Laplacian pyramid to reconstruct a image
- % IMG = PYRRECONSTRUCT(PYR) PYR should be a 1*level cell array containing
- % the pyramid, SIZE(PYR{i}) = SIZE(PYR{i-1})*2-1
- % Yan Ke @ THUEE, xjed09@gmail.com

- for p = length(pyr)-1:-1:1
  - pyr{p} = pyr{p}+pyr_expand(pyr{p+1}); %% upsampled and smoothed at level (p+1)
- end
- img = pyr{1};

- end
Laplacian Scale Space
\[ \text{pyr}\{p\} = \text{pyr}\{p\} + \text{pyr\_expand}(\text{pyr}\{p+1\}); \]
Back to SIFT: where is a SIFT keypoint?

- Definition of a keypoint: Maximum in the 3x3x3 ($x, y, \sigma$) region of the point.
Selected $\sigma$ is visualized with a circle

Denoting the support region of the feature
Detector is rotation invariant

Because Laplacian is isotropic and a maximum in \((x,y,\sigma)\) is invariant to rotations.
Descriptor invariance

• Since the intrinsic scale is detected (circle size) all circles will be normalized to a 16x16 region.
The descriptor should be also rotation invariant

- 1st Step: Find dominant orientation for the patch
The descriptor should be rotation invariant

- 1st Step: Find dominant orientation for the patch
- 2nd Step: Rotate patch to point along x-axis
To extract a feature descriptor from a cell

- Compute Image Gradients
To extract a feature descriptor from a cell

- Compute Image Gradients
- Accumulate gradients along cells
To extract a feature descriptor from a cell

- Compute Image Gradients
- Accumulate gradients along cells
- Form image descriptor
As a matter of fact it is a 4x4 grid of histograms at each keypoint.
The descriptor is an 128x1 vector which together with $\sigma, \theta$ characterize the keypoint.
Example of SIFT detections and feature extraction
Example of SIFT detections and feature extraction

Input Image

Example Detections

Extracted Feature Descriptors
Using SIFT for image matching

Original Image Pair
Using SIFT for image matching

Original Image Pair

Matched features
Create Image Mosaic

1. Get an image pair

2. Establish correspondences between matching features
Create Image Mosaic

3. Keep only consistent matches (inliers)

4. Compute homography and warp 2\textsuperscript{nd} image
Create Image Mosaic

5. Repeat to extend the mosaic
Find Location

Query Image

We want to find a match in a dataset of given images
Find Location

Query Image
Find Location

Good Match

Medium Match

Bad Match
SIFT Features

• SIFT detector can automatically
  • Select scale
  • Compute dominant rotation

• SIFT descriptor
  • Is a grid of histogram of gradient orientations
  • On a region normalized with respect to scale and rotation