Achieving scale covariance

- Goal: independently detect corresponding regions in scaled versions of the same image
- Need scale selection mechanism for finding characteristic region size that is covariant with the image transformation

Blob detection in 2D

Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

\[ \nabla^2 g_\sigma = \frac{\partial^2 g_\sigma}{\partial x^2} + \frac{\partial^2 g_\sigma}{\partial y^2} \]

\( \sigma \) of Gaussian controls the radius of the operator

Scale-normalized

\[ \nabla^2_{\text{normalized}} g_\sigma = \sigma^2 \left( \frac{\partial^2 g_\sigma}{\partial x^2} + \frac{\partial^2 g_\sigma}{\partial y^2} \right) \]

need this to make filter response insensitive to the scale \( \sigma \)

LoG Blob Finding and Scale

Laplacian of Gaussian (LoG) filter extrema locate “blobs”
maxima = dark blobs on light background
minima = light blobs on dark background

Scale of blob (size, radius in pixels) is determined by the sigma parameter of the LoG filter.

Scale Selection

Scale Selection Principle (T. Lindeberg):

In the absence of other evidence, assume that a scale level, at which some (possibly non-linear) combination of normalized derivatives assumes a local maximum over scales, can be treated as reflecting a characteristic length of a corresponding structure in the data.

What are normalized derivatives?

Example (using 2nd order derivatives):

\[ \sigma \nabla^2 f = \sigma^2 \left( \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right) \]

“Laplacian” operator.
Scale selection

- At what scale does the Laplacian achieve a maximum response to a binary circle of radius $r$?

To get maximum response, the zeros of the Laplacian have to be aligned with the circle.

The Laplacian is given by (up to scale):

$$\nabla^2 f = \left( \frac{x^2}{\sigma^2} + \frac{y^2}{\sigma^2} \right) e^{-\frac{x^2+y^2}{2\sigma^2}} - 3 e^{-\frac{x^2+y^2}{2\sigma^2}}.$$

Therefore, the maximum response occurs at $\sigma = r / \sqrt{2}$.

Scale-space blob detector

1. Convolve image with scale-normalized Laplacian at several scales
2. Find maxima of squared Laplacian response in scale-space

Characteristics of scale

- We define the characteristic scale of a blob as the scale that produces peak of Laplacian response in the blob center.

Scale-space blob detector: Example

Efficient implementation

Approximating the Laplacian with a difference of Gaussians:

$$L = \sigma^2 (G_x(x, y, \sigma) + G_y(x, y, \sigma))$$

(Laplacian)

$$\text{DoG} = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)

Why do this: 2D Gaussian filter is separable into two 1D filters, making it more efficient to compute.

Structure

Per-pixel transformations

Interest points: edges, corners, blobs

Feature Descriptors

Templates

- Intensity, gradients, etc.

Histograms

- Color, texture, SIFT descriptors, etc.

or combinations of both

Feature Descriptors

Most features descriptors can be thought of either:

Textons

An attempt to characterize texture

Replace each pixel with integer representing the texture ‘type’

Computing Textons

Take a bank of filters and apply to lots of images

Cluster in filter space

For new pixel, filter surrounding region with same filter bank, and assign to nearest cluster
**Bag of words descriptor**

- Compute visual features in image
- Compute descriptor around each
- Find closest match in library and assign index
- Compute histogram of these indices over the region
- Dictionary computed using K-means

**Sift Descriptor**

Goal: produce a scale and rotation invariant vector that describes the region around an interest point.

- All calculations are relative to the orientation and scale of the keypoint
- Makes descriptor invariant to rotation and scale

**SIFT Detector/Descriptor**

Let’s look “under the hood” at one of the most popular image feature detectors.

**References:**
Generating a Gaussian Pyramid

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

Using a Half-Octave Pyramid

From Crowley et.al., "Fast Computation of Characteristic Scale using a Half-Octave Pyramid." General idea: cascaded filtering using \([1 4 6 4 1]\) kernel (sigma = 1) to generate pyramid with two images per octave (power of 2 change in resolution). When we reach a full octave, downsample the image.

Half Octave Gaussian and DOG

<table>
<thead>
<tr>
<th>Level</th>
<th>Image Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>G2</td>
</tr>
<tr>
<td>G3,L3</td>
<td>G4,L4</td>
</tr>
<tr>
<td>G5,L5</td>
<td></td>
</tr>
</tbody>
</table>

Find Extrema in Space and Scale

Hint: when finding maxima or minima at level \(L\), we must DownSample or UpSample as necessary to make DOG images at level \(L-1\) and \(L+1\) the same size as \(L\).

Filtering out edge responses

Either using Harris corner matrix (2x2 matrix computed from first partial image derivatives)

\[
H = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}
\]

Or use Hessian matrix (2x2 matrix computed from second partial image derivatives)

\[
M = \sum_{x,y} w(x,y) \begin{bmatrix} Dxx & Dxy \\ Dxy & Dyy \end{bmatrix}
\]

Lowe’s notation, Section 4.1
Filtering out edge responses

Would be removed by Harris method

Would be removed by Hessian method

Since the goal is filtering out edges, I like this one better

Any Significant Difference?

Not much difference

Yellow = extrema, blue = removed by Harris corners, red = removed by Hessian

Computing Patch Orientation

Following experimentation with a number of approaches to assigning a local orientation, the following approach was found to give the most stable results. The scale of the keypoint is used to select the Gaussian smoothed image, $I$, with the closest scale, and all computations must be performed at a scale-invariant transform. For each image sample, $I_{x,y}$, the gradient magnitude, $m$, and orientation, $\theta$, is precomputed in the following differences:

$$m = \sqrt{(I_{x,y} \cdot I_{x,y})^2 + (I_{x,y} \cdot I_{x,y})^2}$$

$$\theta = \tan^{-1}\left(\frac{I_{x,y} \cdot I_{x,y}}{I_{x,y} \cdot I_{x,y}}\right)$$

An orientation histogram is formed from the gradient orientations at all sample points within a circular window around the keypoint. Each sample added to the histogram is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a radius three times that of the scale of the keypoint. The orientation histogram has 36 bins covering the 360-degree range of orientations.

Peaks in the orientation histogram correspond to dominant directions of local gradients. The highest local peak in the histogram is detected, and then any other local peak that is within 90% of the highest peak is used to form a keypoint with that orientation. Therefore, for locations with multiple peaks of similar magnitude, there will be multiple keypoints created at the same location and scale, but different orientations. Only about 15% of points are assigned to triple configurations, but these contribute significantly to the stability of matching. Finally, a patch is fit to the 3 histogram values around each peak to interpolate the peak position for better accuracy.

Which Gaussian is Closest in Scale?

G(i) is closest in scale to L

Sigma = 1.18 $\sigma$

(Crowley paper)

Sigma = $\sigma$

Sigma = $\sqrt{2}\sigma$ = 1.1412$\sigma$

Computing Orientation

Following experimentation with a number of approaches to assigning a local orientation, the following approach was found to give the most stable results. The scale of the keypoint is used to select the Gaussian smoothed image, $I$, with the closest scale, and all computations must be performed at a scale-invariant transform. For each image sample, $I_{x,y}$, the gradient magnitude, $m$, and orientation, $\theta$, is precomputed in the following differences:

$$m = \sqrt{(I_{x,y} \cdot I_{x,y})^2 + (I_{x,y} \cdot I_{x,y})^2}$$

$$\theta = \tan^{-1}\left(\frac{I_{x,y} \cdot I_{x,y}}{I_{x,y} \cdot I_{x,y}}\right)$$

An orientation histogram is formed from the gradient orientations at all sample points within a circular window around the keypoint. Each sample added to the histogram is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a radius three times that of the scale of the keypoint. The orientation histogram has 36 bins covering the 360-degree range of orientations.

Peaks in the orientation histogram correspond to dominant directions of local gradients. The highest local peak in the histogram is detected, and then any other local peak that is within 90% of the highest peak is used to form a keypoint with that orientation. Therefore, for locations with multiple peaks of similar magnitude, there will be multiple keypoints created at the same location and scale, but different orientations. Only about 15% of points are assigned to triple configurations, but these contribute significantly to the stability of matching. Finally, a patch is fit to the 3 histogram values around each peak to interpolate the peak position for better accuracy.
Gradient Magnitude and Angle

Computing Orientation

Following experimentation with a number of approaches to assigning a local orientation, the following approach was found to give the most reliable results. The scale of the keypoint is used to select the Gaussian smoothed image, $G$, with the closest scale, as all computations must be performed on a scale-oriented feature. For each image sample, $(x,y)$, the gradient magnitude, $\mathbf{m}$, and orientation, $\phi$, is precomputed using differences:

$$ m = \sqrt{(\mathbf{L}_{x,y})^2 + (\mathbf{L}_{y,x})^2} $$

$$ \phi = \arctan(\frac{\mathbf{L}_{y,x}}{\mathbf{L}_{x,y}}) $$

An orientation histogram is formed from the gradient orientations at all sample points within a circular window around the keypoint. Each sample added to the histogram is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a size three times that of the scale of the keypoint. The orientation histogram has 36 bins covering the 180-degree range of orientations. Peaks in the orientation histogram correspond to dominant directions of local gradients. The highest local peak in the histogram is detected, and then any other local peak that is within 10% of the highest peak is used to create a keypoint with that orientation. Therefore, for locations with multiple peaks of similar magnitude, there will be multiple keypoints created at the same location and scale, but different orientations. Only about 1% of points per sample multiple orientations, but these contribute significantly to the stability of matching. Finally, a patch is fit to the 18 histograms values around each peak to interpolate the peak position for better accuracy.

Sampling/Weighting over a Circular Region

Gaussian-weighted circular window == 2D Gaussian kernel.

Weight magnitude by Gaussian kernel.

What size kernel?

Lowe suggests sigma value of Gaussian kernel should be 3 times larger than scale of the keypoint. Design decision: make the gaussian kernel have sigma = 3, in the pixel coordinate system of $G(I)$, the image from the Gaussian pyramid that magnitude and angle were computed with.

Plus or minus 3 sigma, with sigma = 3, gives a width of 19 pixels.

Example

Example (continued)

Example (continued)
Example (continued)

Computing Orientation

Following experimentation with a number of approaches to assigning a local orientation, the following approach was found to give the most stable results. The scale of the keypoint is used to select the Gaussian-smoothed image $I_\sigma$ with the closest scale, as all computations must be performed at a scale consistent with $I_\sigma$. For each image sample, $I_{\text{sample}}$, the gradient magnitude, $m$, and orientation, $\theta$, is computed using gradient differences:

$$m = \sqrt{(I_{\text{sample}} - I_{\text{sample-1}})^2 + (I_{\text{sample}} - I_{\text{sample-2}})^2}$$

$$\theta = \tan^{-1}((I_{\text{sample}} - I_{\text{sample-1}})/(I_{\text{sample}} - I_{\text{sample-2}}))$$

An orientation histogram is formed from the gradient orientations of all sampled points within a circular window around the keypoint. Each sample added to the histogram is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a three times that of the scale of the keypoint. The orientation histogram has 36 bins covering the 180-degree range of orientations.

A peak in the orientation histogram corresponds to a dominant direction of local gradients. The highest local peak in the histogram is detected, and then any other local peak that is within 99% of the highest peak is used to create a keypoint with that orientation. Therefore, for locations with multiple peaks of similar magnitude, there will be multiple keypoints created at the same location and scale, but different orientations. Only about 1% of points are assigned multiple orientations, but these contribute significantly to the stability of matching.

There may be multiple orientations.

In this case, generate duplicate SIFT patches, one with orientation at 25 degrees, one at 155 degrees.

Design decision: one might want to limit number of possible multiple peaks to two.

Forming the Sift Descriptor

1. Compute image gradients
2. Pool into local histograms
3. Correlate histograms
4. Normalize histograms

Spatial Histogram of Gradients

Computed on rotated and scaled version of window according to computed orientation & scale
- resample the window
Based on gradients weighted by a Gaussian of variance half the window (for smooth falloff)
Spatial Histogram of Gradients
4x4 array of gradient orientation histograms
- each orientation “increment” is weighted by magnitude
8 orientations x 4x4 array = 128 dimensions
Motivation: some sensitivity to spatial layout, but not too much.

Ensure smoothness
Gaussian weight
Trilinear interpolation
- a given gradient contributes to 8 bins:
  4 in space times 2 in orientation

Reduce effect of illumination
128-dim vector normalized to 1
Threshold gradient magnitudes to avoid excessive influence of high gradients
- after normalization, clamp gradients >0.2
- renormalize

SIFT Invariance and covariance properties
- Laplacian blob response (detection) is invariant w.r.t. rotation and scaling
- Blob location covariant w.r.t. rotation and scale
- Normalized patch is invariant wrt rotation/scale
- Normalized vector is insensitive to illumination
- Not invariant/covariant with respect to affine transformations (such as due to small changes in viewing angle)

Achieving affine covariance
Consider the second moment matrix of the window containing the blob:
\[ M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = R \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R \]
Recall:
\[ [u \ v] M [u \ v]^T = \text{const} \]
This ellipse visualizes the “characteristic shape” of the window

Affine adaptation example
Scale-invariant regions (blobs)
Affine adaptation example

Affine-adapted blobs

Affine adaptation

• Problem: the second moment “window” determined by weights \( w(x,y) \) must match the characteristic shape of the region

• Solution: iterative approach
  • Use a circular window to compute second moment matrix
  • Perform affine adaptation to find an ellipse-shaped window
  • Recompute second moment matrix using new window and iterate

Iterative affine adaptation


http://www.robots.ox.ac.uk/~vgg/research/affine/

Affine covariance

• Affinely transformed versions of the same neighborhood will give rise to ellipses that are related by the same transformation

• What to do if we want to compare these image regions?

• Affine normalization: transform these regions into same-size circles

Affine normalization

• Problem: There is no unique transformation from an ellipse to a unit circle
  • We can rotate or flip a unit circle, and it still stays a unit circle

Eliminating rotation ambiguity

• To assign a unique orientation to circular image windows:
  • Create histogram of local gradient directions in the patch
  • Assign canonical orientation at peak of smoothed histogram
From covariant regions to invariant features

- Extract affine regions
- Normalize regions
- Eliminate rotational ambiguity
- Compute appearance descriptors

S. Lazebnik, UNC