# Lecture Notes: Local features

Subhransu Maji

March 19, 2025

## Contents



## 1 Overview

In the next few lectures, we will explore techniques for detecting features in images using convolution operations. These features, such as corners and blobs, play a crucial role in computer vision applications, including panoramic image stitching (Figure 1), 3D reconstruction, motion tracking, and robot navigation. Additionally, they are widely used in image retrieval from large databases and object recognition tasks.

We will then explore the concepts of invariance and equivariance (also referred to as covariance) in the context of local features. We then introduce a feature that is equivariant to scale changes in the blob detector. Unlike the corner detector, which identifies keypoints but does not provide scale information, the blob detector estimates the local scale, enabling feature matching across images captured at different scales. The blob detector serves as the foundation for the widely used Scale-Invariant Feature Transform (SIFT), a powerful feature extraction method commonly applied in 3D geometry estimation and image matching. We will then discuss ways to represent and match local features including the SIFT descriptor.

## 2 Characteristics of Good Features

A good feature should satisfy the following properties:

- **Repeatability**: The same feature should be detectable in multiple images despite geometric and photometric transformations.
- **Saliency**: Each feature should be distinctive.
- **Compactness and Efficiency**: The number of detected features should be significantly smaller than the number of pixels in the image.

- **Locality**: A feature should occupy a relatively small region of the image, making it robust to clutter and occlusion.

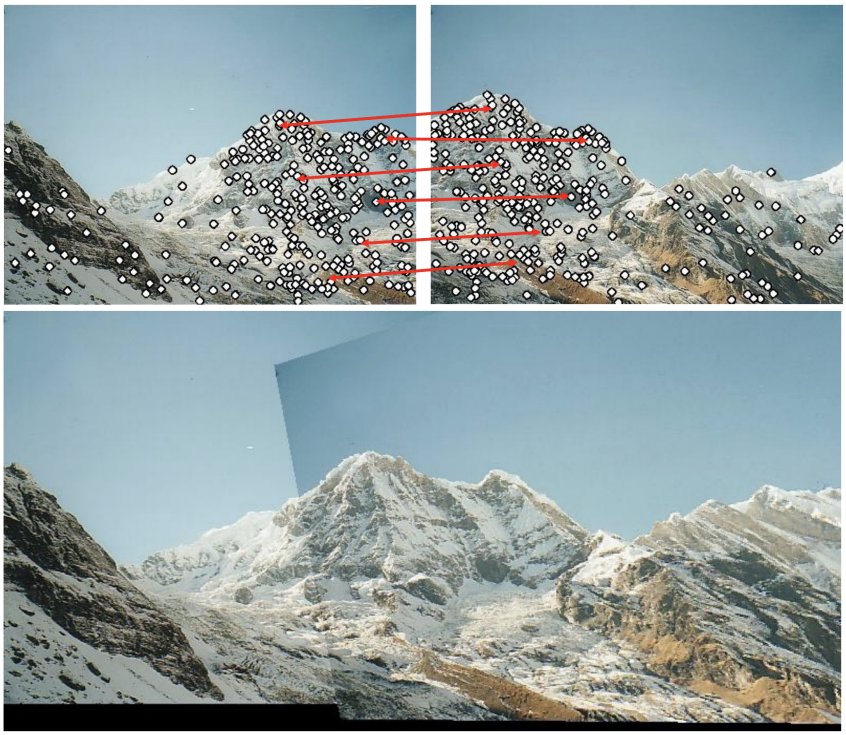


Figure 1: Example of panorama-stitching. Features are first extracted from two images at the top then matched with each other. The two images are then transformed and aligned to be stitched together.

# 3 Corners

Corners can be intuitively identified by analyzing small image regions as show in Figure 2:

- Shifting a small window in any direction over a corner results in a significant change in intensity.
- This property distinguishes corners from edges, where changes occur only along the perpendicular direction, and from flat regions, where no significant change occurs in any direction.

## 3.1 Mathematical Formulation

We aim to model the change in appearance of a window $W$ centered at $(x, y)$ when it is shifted by $(u, v)$ in an image $I$:

$$E(u, v) = \sum_{(x,y)\in W} [I(x + u, y + v) - I(x, y)]^2 \tag{1}$$

A corner is a location where this function varies significantly in all directions $(u, v)$.
Using a first-order Taylor approximation for small shifts $(u, v)$:

$$I(x + u, y + v) \approx I(x, y) + I_x u + I_y v \tag{2}$$

Substituting this into $E(u, v)$, we get:

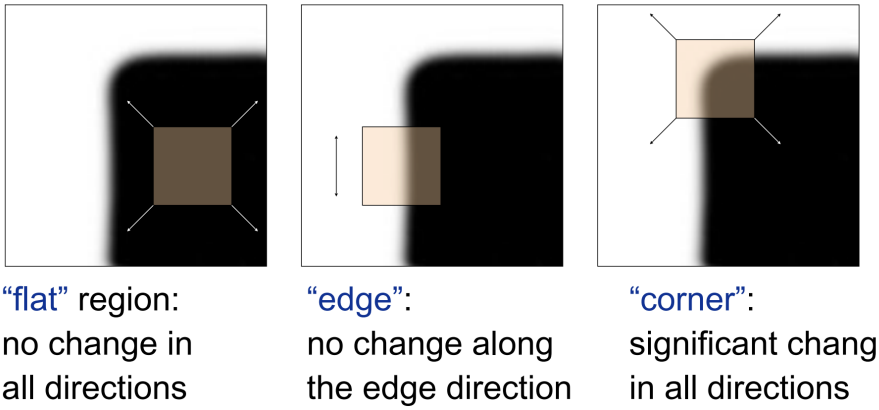



Figure 2: Identifying corners. (Left) The intensity does not change when the window is shifted in a "flat" region. (Middle) The intensity does not change when the window is shifted along the edge direction. (Right) The intensity of the window changes when shifted in arbituary direction.

$$\begin{aligned} E(u,v) &= \sum_{(x,y)\in W} [I(x+u,y+v) - I(x,y)]^2 \\ &\approx \sum_{(x,y)\in W} [I_x u + I_y v]^2 \\ &= \sum_{(x,y)\in W} [I_x^2 u^2 + 2I_x I_y uv + I_y^2 v^2] \end{aligned}$$

This quadratic approximation can be rewritten as:

$$E(u,v) = \begin{bmatrix} u & v \end{bmatrix} \mathbf{M} \begin{bmatrix} u \\ v \end{bmatrix}$$

where $\mathbf{M}$ is the second moment matrix computed from image derivatives:

$$\mathbf{M} = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

### 3.2 Interpreting the Second Moment Matrix

The function $E(u,v)$ is locally approximated by a quadratic form. We analyze its shape to determine in which directions the intensity changes most.

First, consider the special case where gradients are either purely horizontal or vertical. In this case, the second moment matrix simplifies to a diagonal form:

$$\mathbf{M} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

where $a$ and $b$ represent the intensity changes in the horizontal and vertical directions, respectively. A point is considered a corner if both $a$ and $b$ are large.

In general, we can rotate the coordinate system at each pixel so that the gradients become axis-aligned. This results in:

$$\mathbf{M} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R \tag{3}$$

where $\lambda_1, \lambda_2$ are the eigenvalues of $\mathbf{M}$, and $R$ is a rotation matrix. The eigenvalues determine the magnitude of change in two perpendicular directions as shown in Figure 3 and 4.

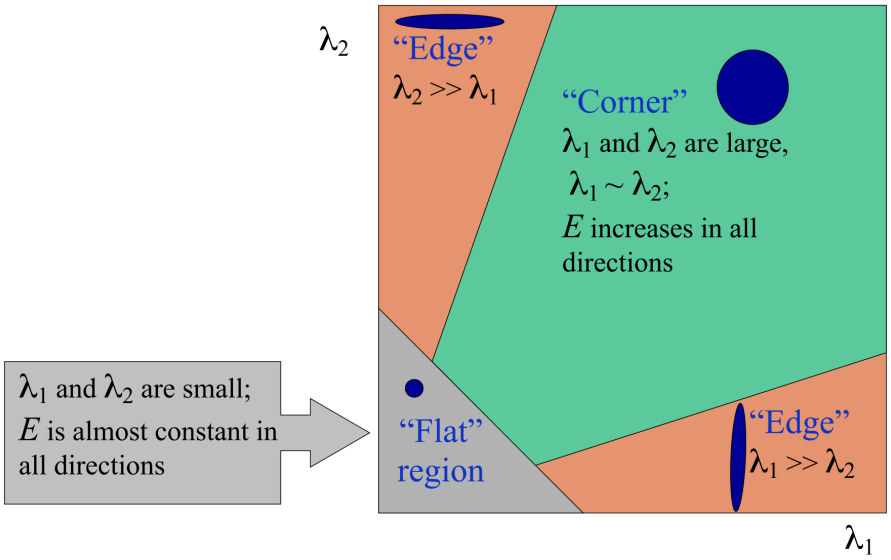



Figure 3: Interpreting the second moment matrix. The eigenvalues $\lambda_1, \lambda_2$ of the second moment matrix determine the mangnitude of change in two perpendicular directions.

### 3.3 Harris Corner Detector

The Harris corner detector has the following steps

1. Compute partial derivatives $I_x, I_y$ at each pixel by convolving the image $I$ with derivative filters.
2. Compute second moment matrix $M$ in a box / Gaussian window around each pixel. Each entry of the M is a weighted sum of $I_x^2, I_x I_y, I_y^2$ around each pixel which can be efficiently computed using convolution with a box / Gaussian filter $w$.

$$\mathbf{M} = \begin{bmatrix} \sum_{(x,y)\in W} w(x,y) I_x^2 & \sum_{(x,y)\in W} w(x,y) I_x I_y \\ \sum_{(x,y)\in W} w(x,y) I_y I_x & \sum_{(x,y)\in W} w(x,y) I_y^2 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

3. Compute the corner response function $R$ at each pixel. The Harris corner detector computes the response as:

$$R = \det(M) - \alpha \text{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2. \tag{4}$$

   The constant $\alpha$ is set between 0.04 and 0.06 and $\lambda_1, \lambda_2$ are the eigenvalues of $M$. Instead of computing these you can use the following identites to compute the response function directly

$$\det(M) = \lambda_1 \lambda_2 = (ad - bc)$$
$$\text{trace}(M) = \lambda_1 + \lambda_2 = (a + d)$$

   To compute R as:

$$R = (ad - bc) - \alpha(a + d)^2 \tag{5}$$

4. Threshold $R$.
5. Find local maxima of the thresholded response.

The function can be implemented in Python efficiently. Use convolutions for Step 1 and to compute the entries of the matrix $M$ in Step 2. The response can be computed across all function using broadcasting without requiring loops.

## 4 Invariance and Covariance

- Invariance: A feature is considered invariant to a transformation if the transformation does not alter its properties.

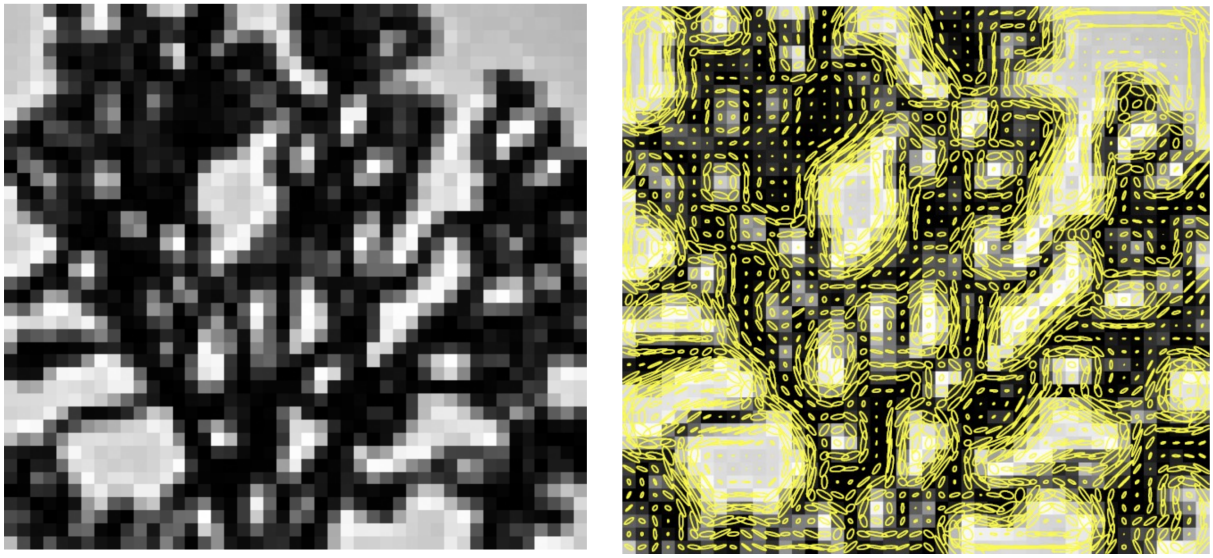


Figure 4: Visualization of the second moment matrix for each pixel in the image. The rotation matrix $R$ determines the orientation of the ellipses whereas the eigenvalues, $\lambda_1, \lambda_2$, of M determines the axis lengths of the ellipses, which implies the intensity of changes. (Right) Pixels in the "flat" region have small ellipses; pixels on the edge have elongated ellipses, with one axis significantly greater than the other, aligning with the direction of the edge; pixels at the corner have large ellipses that almost look like a circle, indicating strong intensity change in both orthogonal directions.

- Covariance or equivariance: A feature is considered equivariant (or covariant) with respect to a transformation if its properties change in a predictable manner under that transformation.

In the context of local feature detection, we aim for features to be invariant to photometric transformations while remaining covariant to geometric transformations.

### 4.1 Understanding the Properties of the Corner Detector

- Affine intensity changes: Photometric transformations such as affine intensity changes follow the form $I \to aI + b$. Since the corner detector operates on image derivatives, intensity shifts $I \to I + b$ have no effect. However, intensity scaling $I \to aI$ also scales the derivatives and the response functions, as shown in Figure 5a. As a result, the corner detector is partially invariant to affine intensity changes.

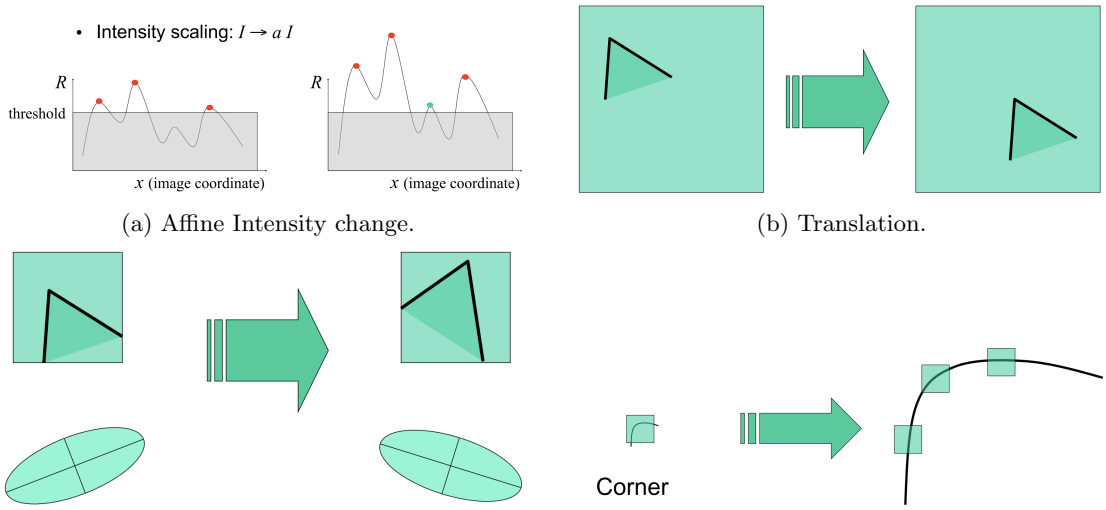
- Translation: Since image derivatives and the window function used for computing the corner response are shift-invariant (as they are based on convolutions), the response shifts by the same amount as the image translation (Figure 5b). Therefore, the corner detector is covariant with respect to translation.

- Rotation: When the image is rotated, the second moment ellipse also rotates, but its shape (i.e., its eigenvalues) remains unchanged as shown in Figure 5c. Consequently, the corner detector is covariant with respect to rotation.

- Scaling: Scaling affects the corner detector because, depending on the window size, a structure that appears as a corner at one scale may resemble an edge when the image is scaled up (Figure 5d). Thus, the corner detector is neither invariant nor covariant with respect to image scale.

## 5 Feature Detection with Scale Selection

How can we design a detector which has a characteristic scale that matches the image scale. The blob detector, which is the scale-invariance part of the SIFT (scale-invariant feature transform) achieves this. The key high-level ideas of the SIFT descriptor are:




(a) Affine Intensity change.

(b) Translation.

(c) Rotation

(d) Scaling a corner

Figure 5: The properties of the corner detector. Subfigure (a) shows that certain points on the derivative function may exceed the threshold for identifying corners due to intensity scaling, thus corner detector is partially invariant to affine intensity change. Subfigure (b) shows that the corner detector is covariant with respect to translation since the corner location shifts by the same amount as the the translation. Subfigure (c) shows that the corner detector is covariant with respect to rotation as the location of the corner also rotates. Subfigure (d) shows that a scaled corner may appear as an edge in the fixed size corner detector, thus the corner detector is neither invariant nor covariant with respect to image scale.

- First, we find points of interest, known as "keypoints" in the image. These will be *local extrema* (minima or maxima) of the "blob detector" across scale space (discussed below). The key difference between corner (e.g. Harris) and blob detection is that blobs offer scale selection.
- Some of these local extrema points are thrown out because they are unstable. This means that a very small change to the image (changing one pixel by one brightness value, for example), may change whether the point is a local extremum. Points in smooth regions of the image and along "ridges" (like images of folding curtains) are typically the types of points that are thrown out. You can filter unstable points by thresholding the blob detection score.
- After a keypoint is found (it is at a local extremum and it is not unstable), its "scale" is defined to be the radius of the blob detector for which the detection score was highest. If it was found by the largest blob detector, it will have a large scale.
- Then a keypoint *orientation* is assigned. This is the, or one of the, dominant orientations in a patch around the keypoint. The dominant orientation is found by looking at a histogram of the gradient orientations in the patch, and picking the orientations with the most values in the histogram.
- Finally, with keypoints that have scales and orientations, we put a set of 4x4 bins down at the given orientation and scale, and build sixteen histograms of local gradient magnitudes. Since each histogram has 8 bins, this will give us a total of 4x4x8 values for on SIFT *descriptor*. Understand what role the number of spatial and orientation bins in the SIFT descriptor.

You should understand all of the above points about SIFT keypoints and descriptors. Here are some additional details you need to understand to apply SIFT features.

## 5.1 Blob Detection – the Math

The basic idea for scale selection is to have multiple copies of a blob detector, one for each scale, and pick the scale that corresponds to the extrema of the response across scales at each location. A blob detector for a scale $\sigma$ is given by

$$\nabla^2_{\text{norm}} g = \sigma^2 \left( \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right)$$

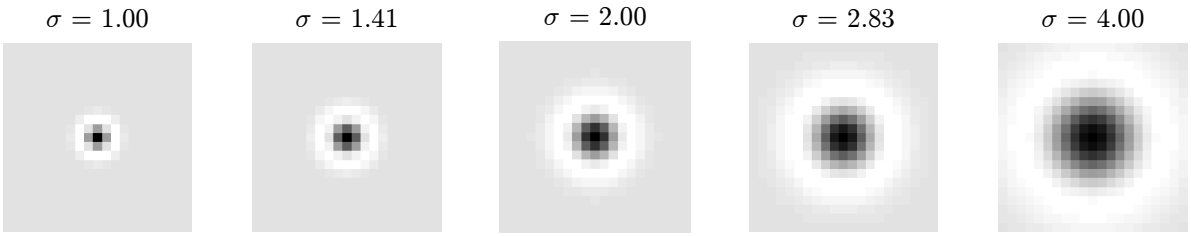



Figure 6: Scale normalized Laplacian-of-Gaussian filters for various $\sigma$.

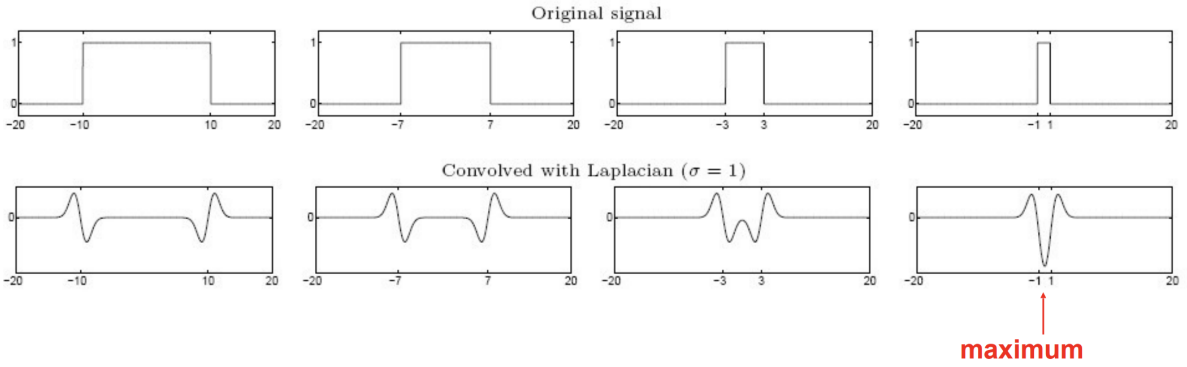



Figure 7: Blob response as the superposition of two edges. Consider the above plots as the cross-section of a blobs original signal (top) and the response signal after convolved with Laplacian (bottom). The Laplacian response reaches the maximum at the center of the blob when the size of Laplacian matches the size of the blob, which is when the two edge responses superpositions

Here $g$ is a Gaussian with a variance $\sigma$, and $\frac{\partial^2 g}{\partial x^2}, \frac{\partial^2 g}{\partial y^2}$ are partial double derivatives of $g$ with respect to $x$ and $y$. This is also called as the scale normalized Laplacian-of-Gaussian (LoG) filter since we are multiplying the LoG by $\sigma^2$. This normalization is needed because the response of the LoG filter scales inversely with $\sigma^2$, and thus by scaling the filter by $\sigma^2$ the responses across scales are comparable.

A set of blob detectors are creating successively scaled copies of the scale normalized LoG filters. Starting with $G_1$ with $\sigma_1 = 1$, the next filter $G_2$ is obtained by setting

$$\sigma_2 = \sigma_1 * r$$

for a fixed ratio $r$. In general

$$\sigma_{k+1} = \sigma_k * r.$$

Figure 6 shows five successive filters with a ratio $r = \sqrt{2}$. Visually these filters resemble blobs of increasing sizes. Given a set of filters $G_1, G_2, \ldots G_K$ we can convolve an image $I$ to obtain a score map for each scale

$$S_k = I \otimes G_k.$$

Here $\otimes$ is the convolution operation.

**Assigning a scale.** The scale of the blob is obtained by the local exterma over scales. A pixel in the stack of images $S_1, S_2, ..., S_K$ is a *local extremum* if it is larger than its 26 neighbors or smaller than its 26 neighbors.

**Assigning an orientation.** Using the partial derivative filters described in our previous work on filtering, you can compute the partial derivatives of an image at each point, and hence form the gradient of the image at each point. You can compute the magnitude of the gradient simply by using the vector magnitude formula. Assuming the magnitude is not zero, you can compute the gradient orientation as

$$\theta = \arctan\left(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x}\right).$$

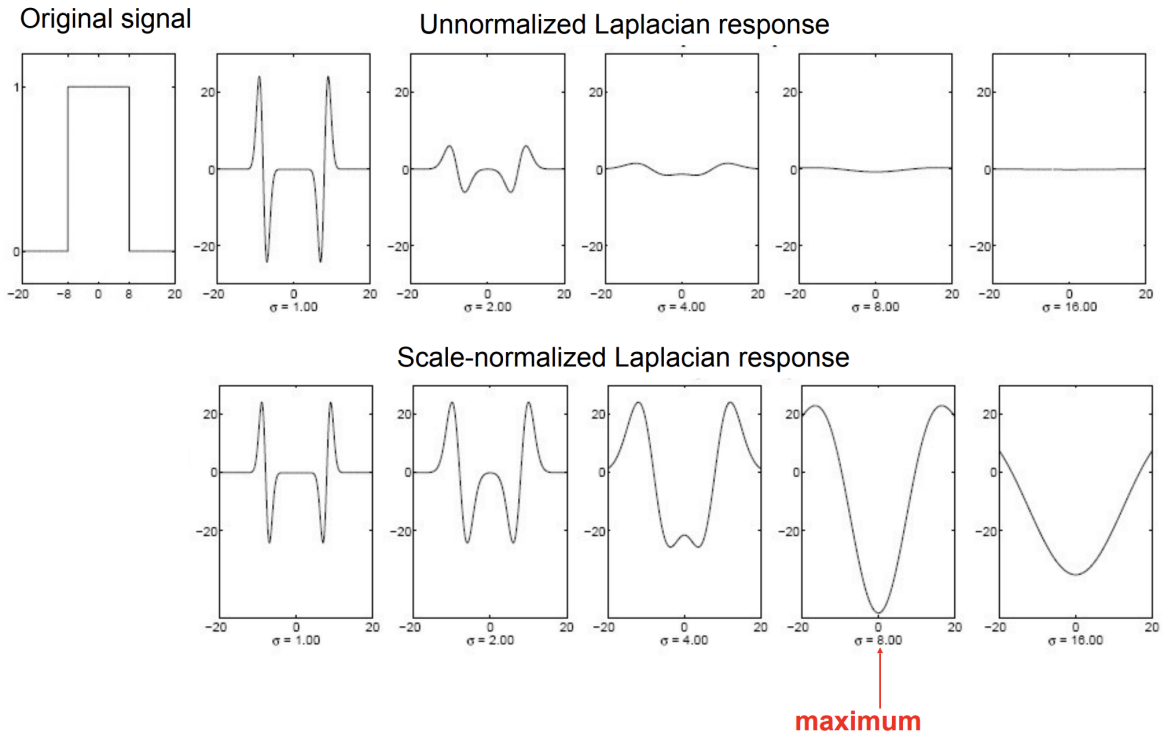



Figure 8: Unnormalized Laplacian response vs Scale-nromalized Laplacian response.

The domainant orientation of the pixels within the blob can be used to define a coordinate system. This allows a representation that is covariant to image rotations. SIFT features can be computed by spatially binning the pixels in the patch and obtaining a orientation histogram. SIFT was specifically defined to create descriptors for image points such that if the same object were seen again, the descriptor for points on that object would be highly similar to the previous view of that object.

### 5.2 Understanding the Properties of the Blob Detector

The blob detector has the following properties:

- It is partially invariant to affine intensity changes for the same reasons as the corner detector.
- It is covariant with respect to translation due to the use of convolutions.
- It is covariant with respect to image scale due to the scale selection mechanism of the blob detector.
- It is covariant with respect to rotation since the response to the blob filter remains unchanged under image rotation due to its rotational symmetry. Assigning a dominant orientation allows the extracted features to be invariant to image rotation.

## 6 From Feature Detection to Description

To find correspondences between images, we need to identify regions and compare their feature descriptors. For region detection, we can use either the corner detector or the blob detector, depending on whether we want to match images across different scales.

The simplest feature descriptor directly represents the image patch corresponding to each detected feature using a vector of raw intensity values.

- For a corner detector, the patch size can be set proportional to the window size (or the Gaussian sigma) used in the detection step.
- For a blob detector, a patch of size proportional to the characteristic scale of the blob can be extracted. Different-sized blobs can be compared by resizing them to a canonical size (e.g., $64 \times 64$ patches).

**Patch Comparison Methods** The extracted patches can be compared using the following measures:

- Sum of squared differences (SSD) – A distance measure that is not invariant to intensity changes

$$\text{SSD}(u, v) = \sum_i (u_i - v_i)^2 \tag{6}$$

- Normalized correlation – A similarity measure that is invariant to affine intensity transformations

$$\rho(\mathbf{u}, \mathbf{v}) = \frac{\sum_i (u_i - \bar{u})(v_i - \bar{v})}{\sqrt{(\sum_i (u_i - \bar{u})^2)\,(\sum_i (v_i - \bar{v})^2)}} \tag{7}$$

However these matching score can be affected by:

- Small deformations. Shifting the patch by just a few pixels can significantly alter the similarity score.
- Small changes in intensity. SSD is also sensitive to the small changes in intensity values. Normalized correlation is robust to this.

Since the feature extraction can lead to inaccuracies in the position and scale estimation due to image noise and discrete nature of the scale-space analysis, we need representations that are more robust. This motivates the SIFT descriptor.

## 6.1 The SIFT Descriptor

The Scale-Invariant Feature Transform (SIFT) descriptor is computed using the following steps:

1. Divide the image patch into $4 \times 4$ sub-patches.
2. Compute a histogram of gradient orientations (with 8 reference angles) within each sub-patch.
3. Normalize the descriptor to unit length.

The resulting descriptor has $4 \times 4 \times 8 = 128$ dimensions.

**Advantages Over Raw Pixel Values**

- Gradients are less sensitive to illumination changes.
- Pooling gradients over sub-patches provides robustness to small shifts while still preserving some spatial information.