AI6121 Computer Vision

Image Features – Points

Contents and Learning Objectives

1. Introduction
2. Feature Point Detection
3. Feature Point Description
4. Feature Matching
5. Applications

1. Introduction

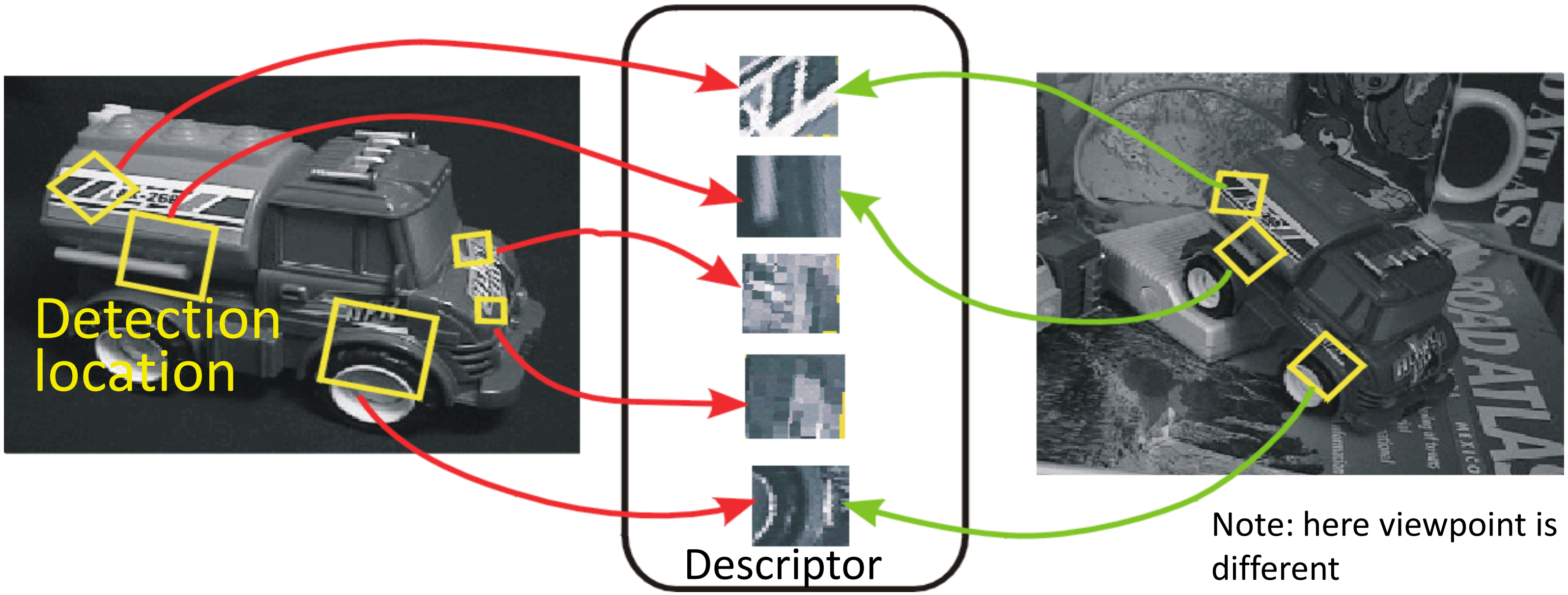
Find the two objects from the image:

1. Introduction

How to perform panorama by stitching multiple images together?

1. Introduction

- **Detection**: detect same scene points independently in both images
- **Description**: encode local neighboring window to describe the scene point
  - Note how scale & rotation of window are the same in both image (but computed independently)
- **Matching (Correspondence)**: find most similar descriptor in other image



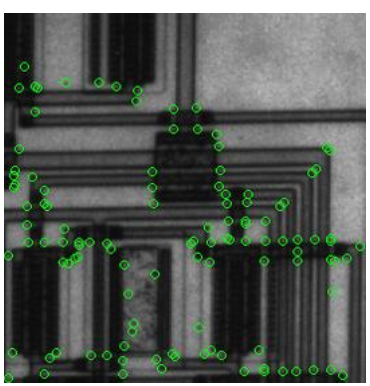
Note: here viewpoint is different

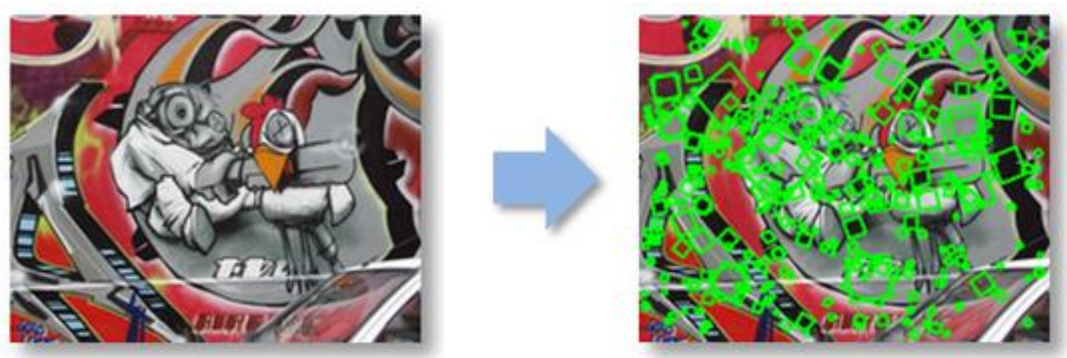
David G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV, 2004.

1. Introduction: What are features?

The definition of features varies. In computer vision and image processing, a feature is a piece of information which is relevant for solving the computational task related to a certain application. Features may be specific structures in the image such as points, edges or objects.

Local features refer to a pattern or distinct structure in an image, such as a point, edge, or small image patch. They are usually associated with an image patch that differs from its immediate surroundings by texture, color, or intensity. Examples of local features are blobs, corners, and edge pixels.





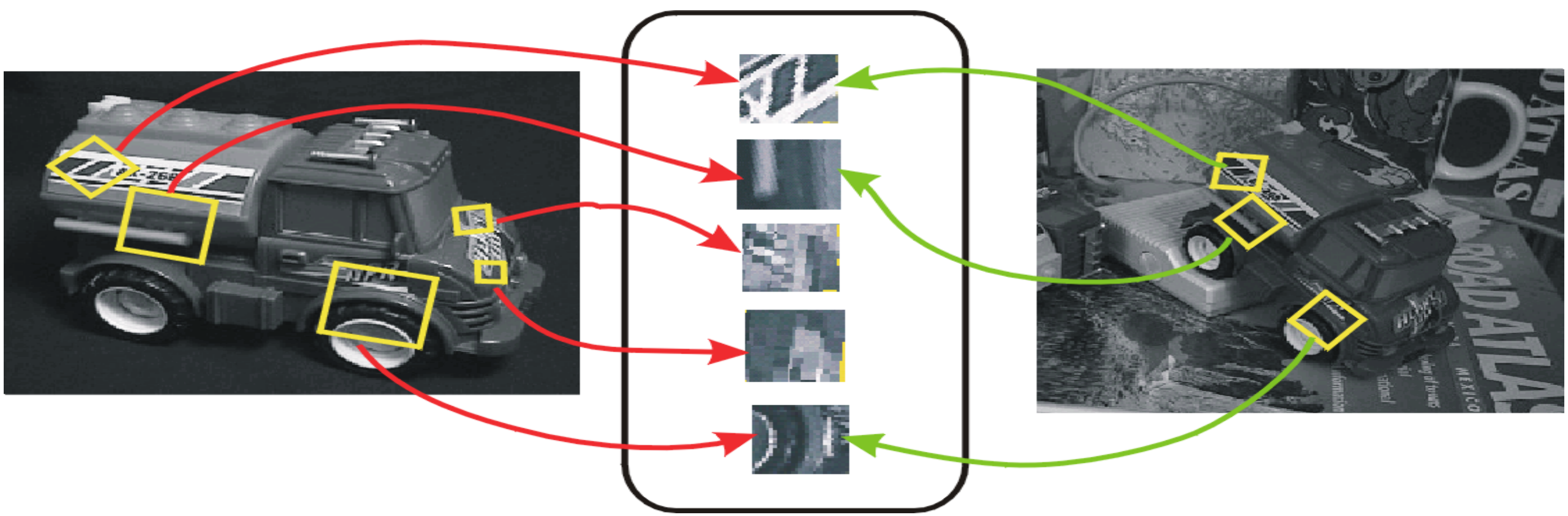
David G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV, 2004.

1. Why Feature Detection and Description

Feature detection and feature description is often the starting point in many computer vision tasks.

Images features are used in two fundamental ways:

- To localize anchor points for use in image stitching, 3-D reconstruction, etc.
- To represent image contents compactly for image classification, object detection and recognition, without requiring image segmentation.



1. Introduction: What makes a good feature?

Good image features usually exhibit the following properties:

Repeatable detections: When given two images of the same scene, most features that the detector finds in both images are the same. The features are robust to changes in viewing conditions and noise.

Distinctive: The neighborhood around the feature center varies enough to allow for a reliable comparison between the features.

Localizable: The feature has a unique location assigned to it. Changes in viewing conditions do not affect its location.

1. Introduction: What makes a good feature?

Repeatable

No chance to match

Distinctive

Feature description

Localizable

Minimal occlusion

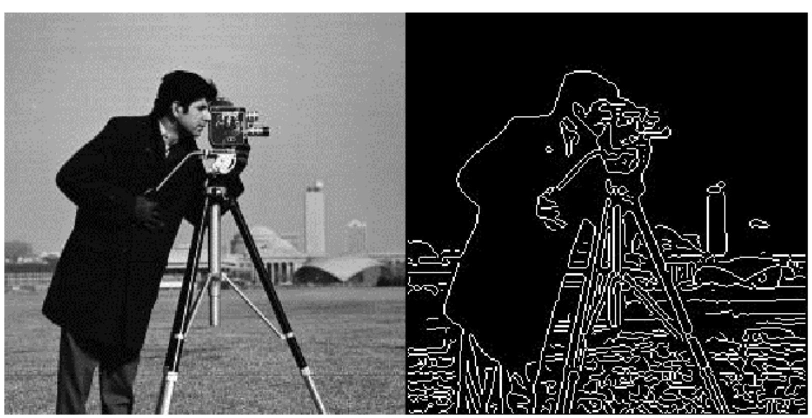
2. Feature Detection

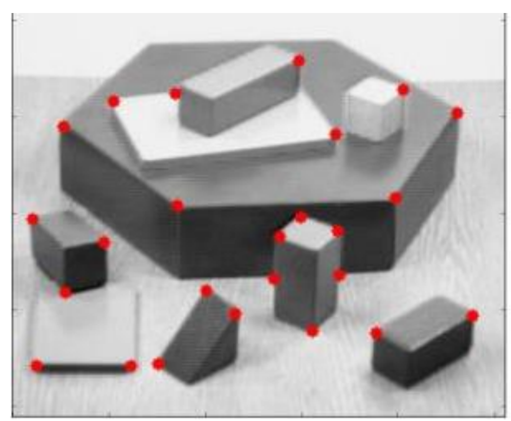
There are different types of features:

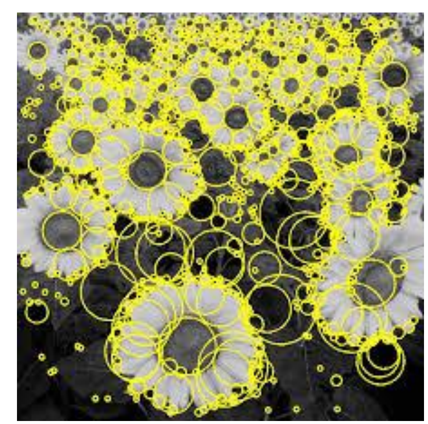
Edges: Edge are points where there is a boundary between two image regions. They are often defined as sets of points that have a strong gradient magnitude.

Corners: Corners (or interest points) refer to point-like features in an image, which have a local two dimensional structure. They can be detected by finding rapid changes in the direction of edges, or high levels of curvature in the image gradient.

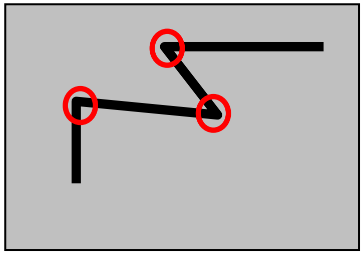
Blobs: Blobs provide a complementary description of image structures in terms of *regions*, as opposed to corners that are more point-like. Blob detectors can detect areas in an image which are too smooth to be detected by a corner detector.





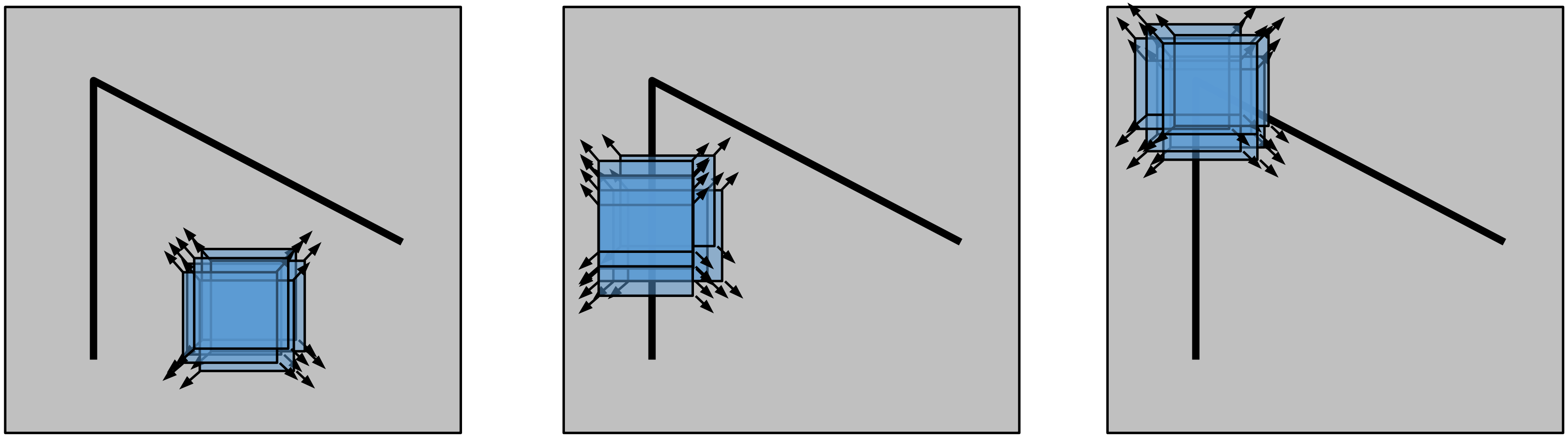


2. Feature Detection – Harris Corner



Harris corner detector

- The points should be easily localized by checking a small window.
- Shifting a window in any direction should give a large change in pixels intensities in window.



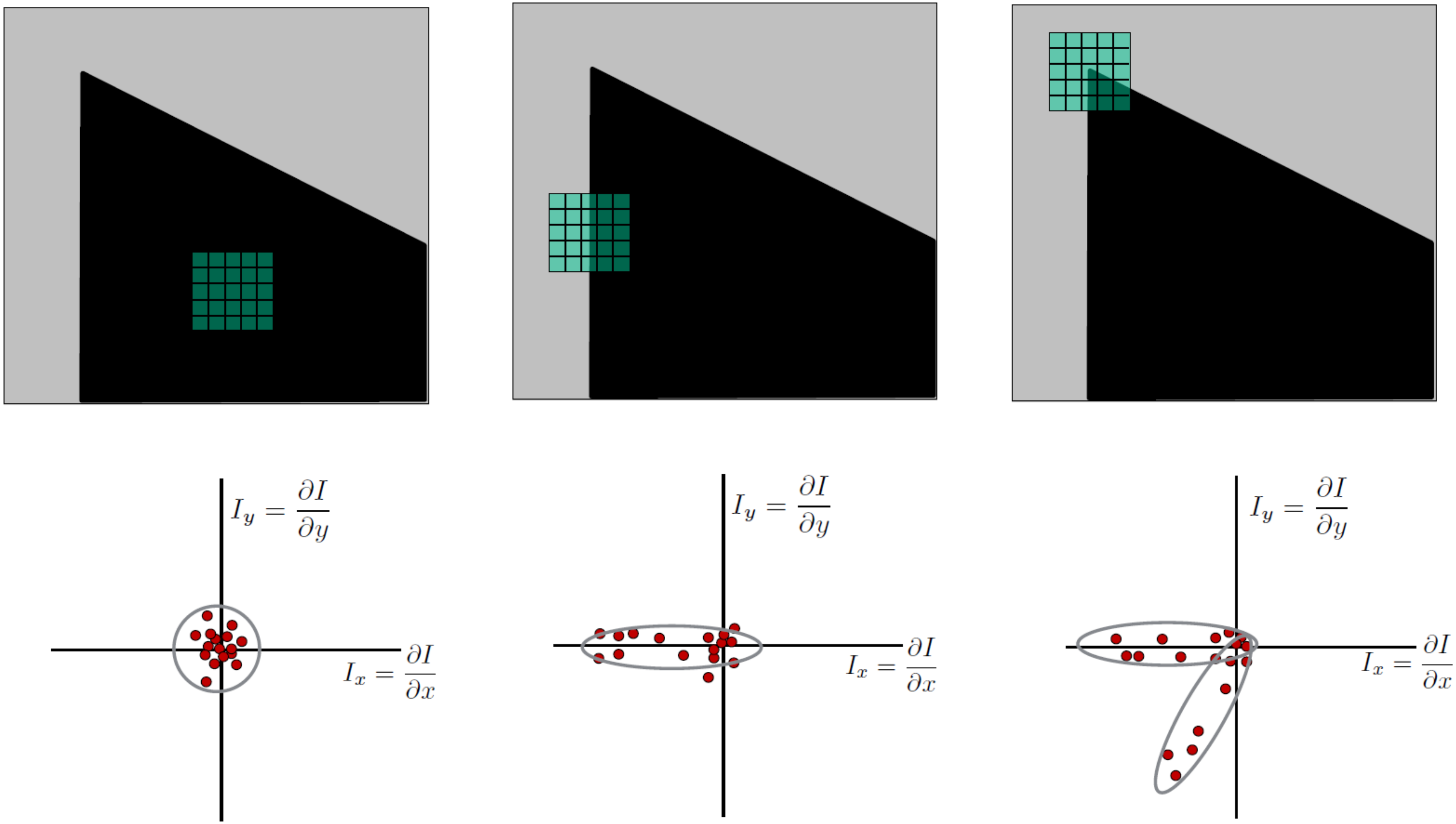
Flat region: no change in all directions

Edge: no change along the edge direction

Corners: significant change in all directions

2. Feature Detection – Harris Corner

The graph illustrates the pixel values when the region window moves.



2. Feature Detection – Harris Corner

Window-averaged squared change of intensity induced by shifting the image data by [u,v]:

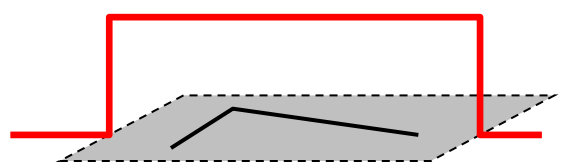
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

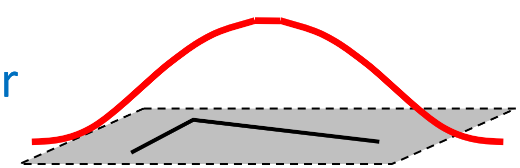
Original Intensity

Window function w(x,y) =



1 in window, 0 outside

or



Gaussian

2. Feature Detection – Harris Corner

When *u* and *v* are small, *I(x+u, y+v)* can be expanded with Talor Series expansion:

$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \approx I(x,y) + [I_x \;\; I_y] \begin{bmatrix} u \\ v \end{bmatrix}$$

$$I_x = \frac{\partial I}{\partial x} \qquad I_y = \frac{\partial I}{\partial y}$$

I_x and I_y are usually first subtracted by their means before further computing.

2. Feature Detection – Harris Corner

The image change can thus be derived as follows:

$$E(u,v) \approx \sum_{x,y} w(x,y)[I(x,y) + uI_x + vI_y - I(x,y)]^2 \quad = \sum_{x,y} w(x,y)[uI_x + vI_y]^2$$

$$= (u \quad v) \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

$$E(u,v) \cong [u,v] \; M \; \begin{bmatrix} u \\ v \end{bmatrix}$$

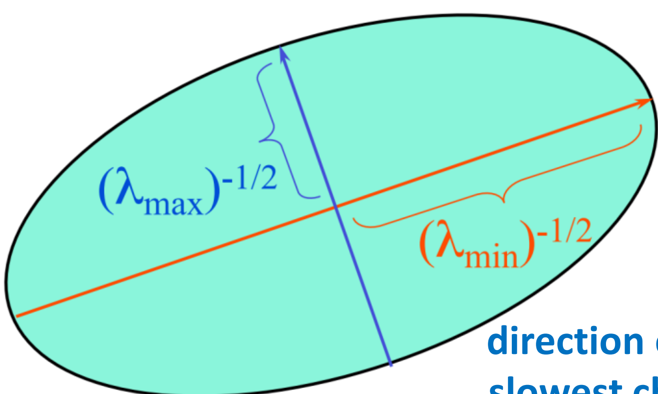
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

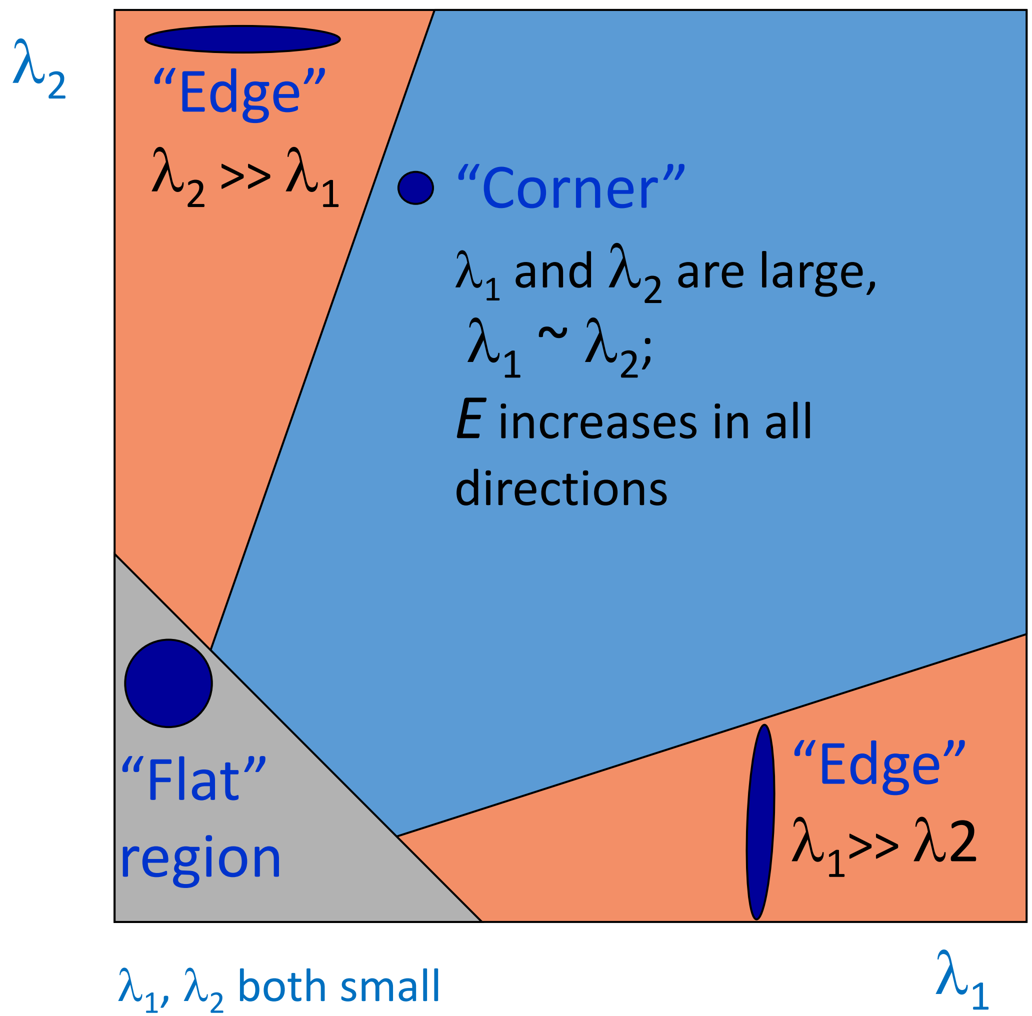
M is a 2×2 matrix which can be computed from image derivatives. It's often called structure tensor.

2. Feature Detection – Harris Corner

Image pixels can be classified based on the two eigenvalues of M: λ_1 and λ_2.

λ_1, λ_2: Eigenvalues of M





2. Feature Detection – Harris Corner

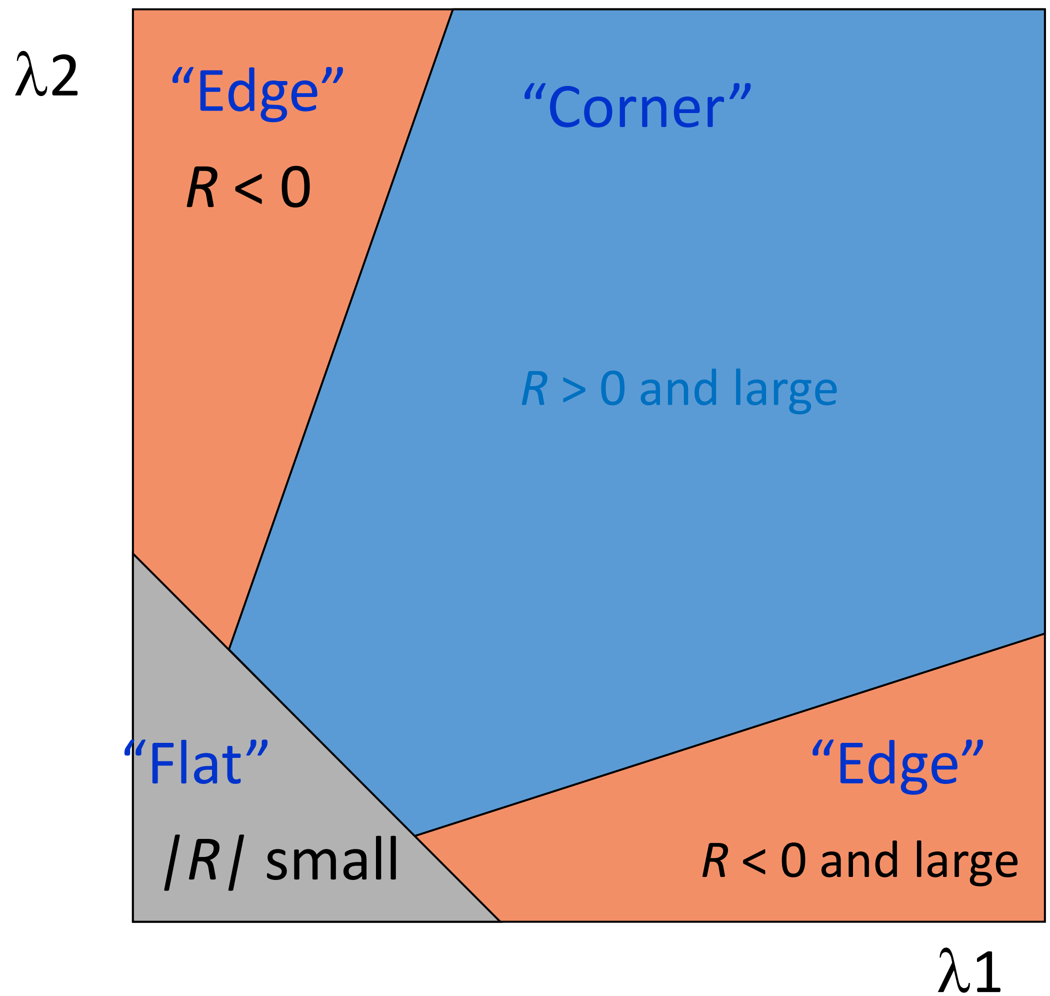
Image pixels can be classified based on the two eigenvalues of *M:* λ_1 and λ_2.

Measure of corner response:

$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

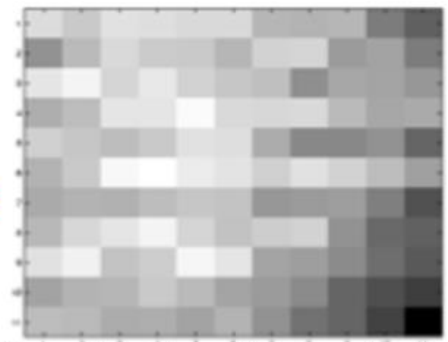
$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

(k – empirical constant,
k = 0.04-0.06)



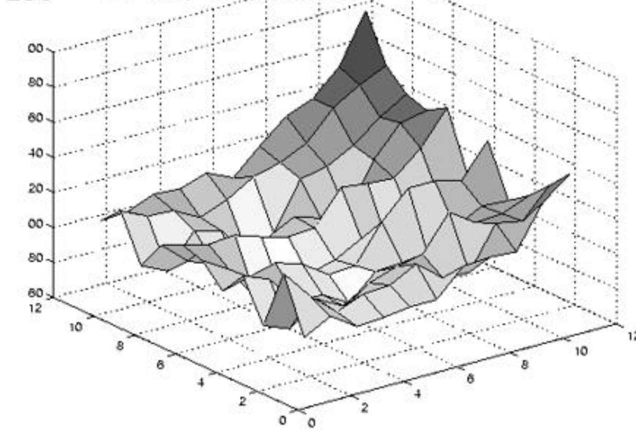
2. Feature Detection – Harris Corner

Image patch



(contrast auto-scaled)

200 Error surface



λ_1 and λ_2 are both small
(vertical scale exaggerated)

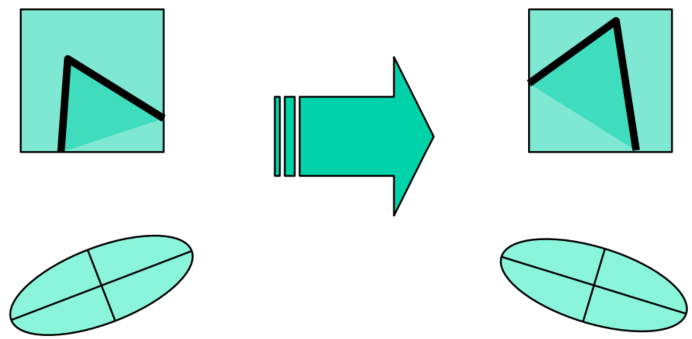
$$E(u,v) \cong [u,v] \; M \; \begin{bmatrix} u \\ v \end{bmatrix}$$

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

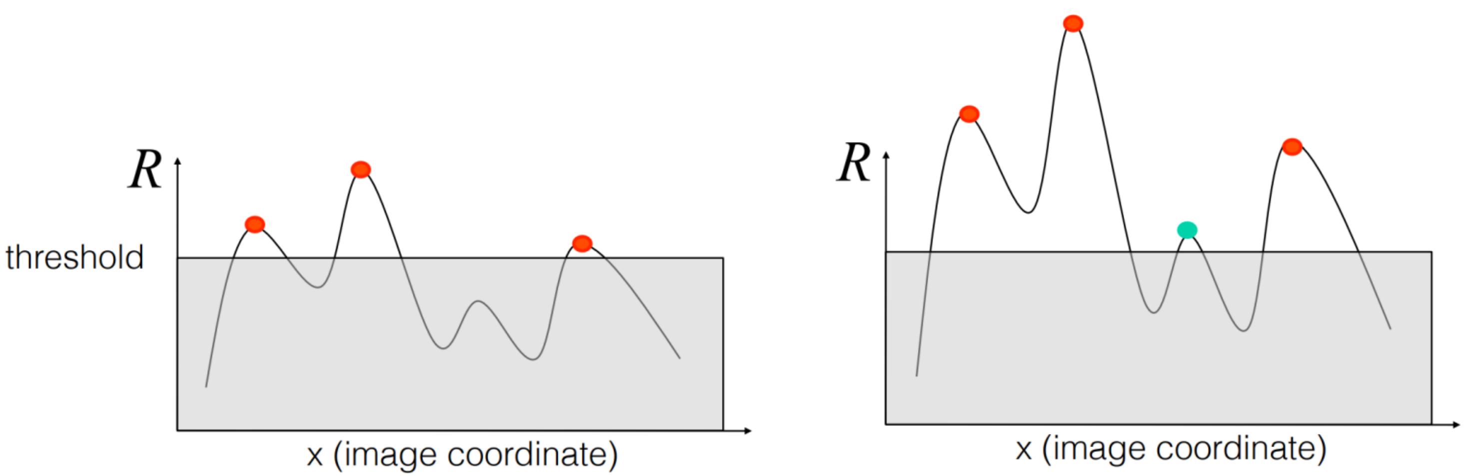
2. Feature Detection – Harris Corner

2. Feature Detection – Harris Corner

The corner response *R* is invariant to rotation. The ellipse rotates but its shape (eigenvalues) remains the same.

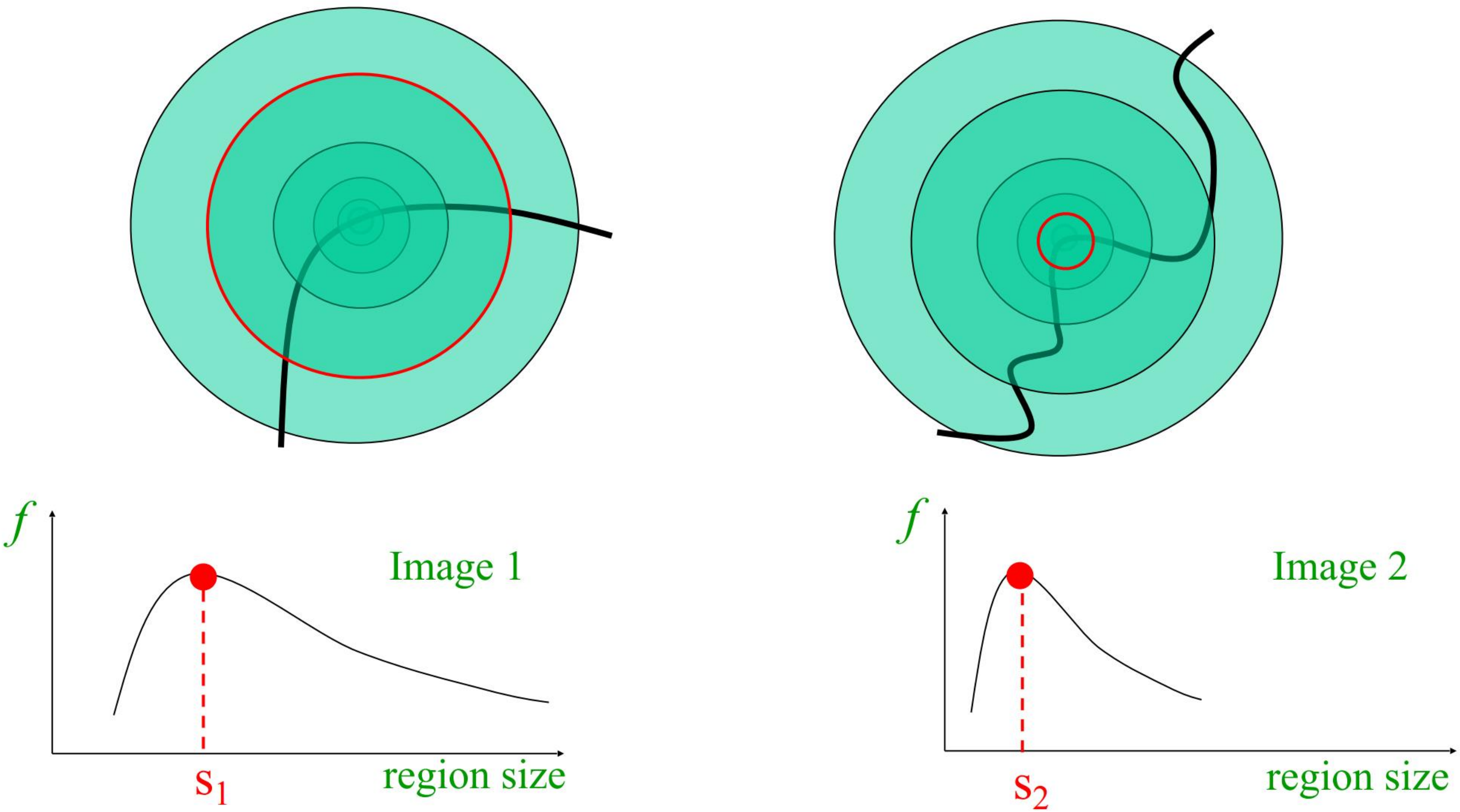


The corner response *R* is partially invariant to intensity changes.



2. Feature Detection – Harris Corner

The corner response *R* changes with respect to scales.



2. Feature Detection – Scale Space Extrema

Laplacian of Gaussian locates edges and corners well. SIFT uses Difference of Gaussian (DoG) to approximate scale-normalized Laplacian of Gaussian to detect scale-space extrema.

Given an image $I(x, y)$, its scale space is defined by $L(x, y, \sigma)$ which is produced by convoluting $I(x, y)$ with a variable-scale Gaussian:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Where * denotes convolution and

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

The DoG thus becomes

$$\begin{aligned} D(x, y, \sigma) &= \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned}$$

2. Feature Detection – Scale Space Extrema

Laplacian of Gaussian $\nabla^2 G$ is not scale-invariant because of the σ^2 in the denominator of the Gaussian function. From a heat diffusion equation we have

$$\sigma \nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

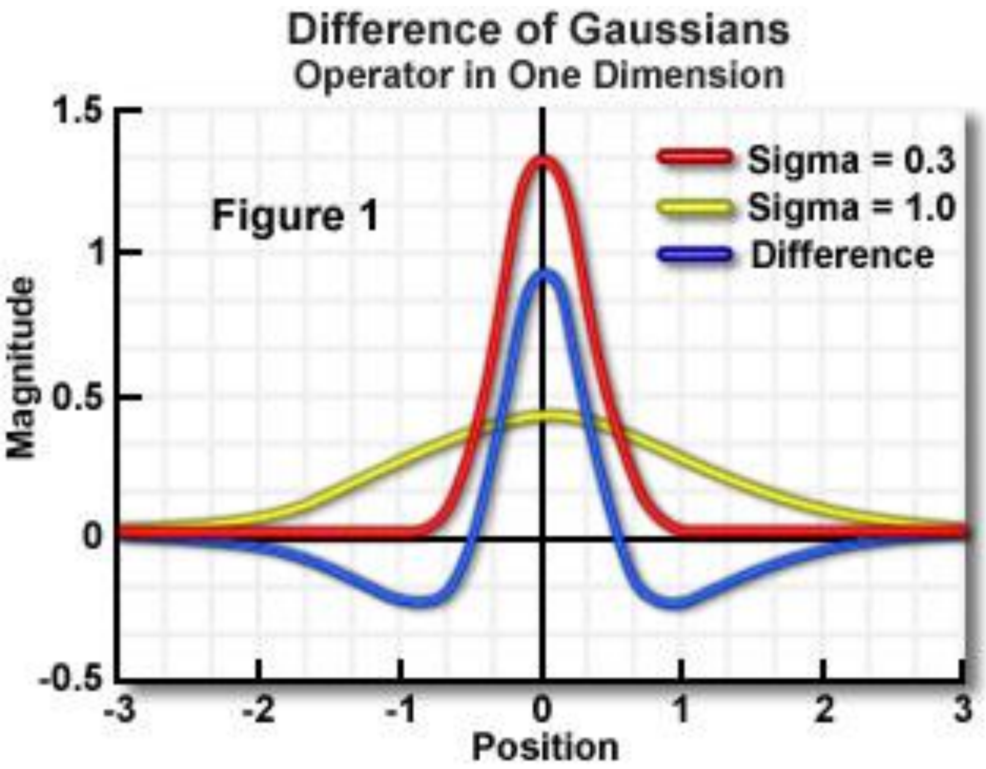
Therefore

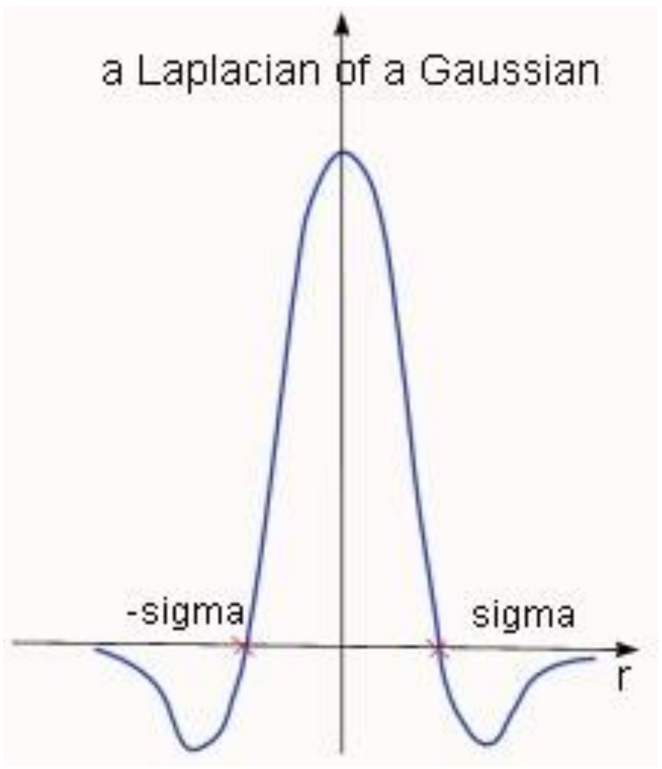
$$G(x, y, k\sigma) - G(x, y, \sigma) = (k-1)\sigma^2 \nabla^2 G$$

The DoG can thus be approximated by a scale-normalized Laplacian of Gaussian (σ^2 is cancelled), up to a scaling factor $k-1$ which does not affect the detection of scale-space extrema.

2. Feature Detection – Scale Space Extrema

The graphs below also show that DoG approximiates Laplacian of Gaussian well, up to a scaling factors.





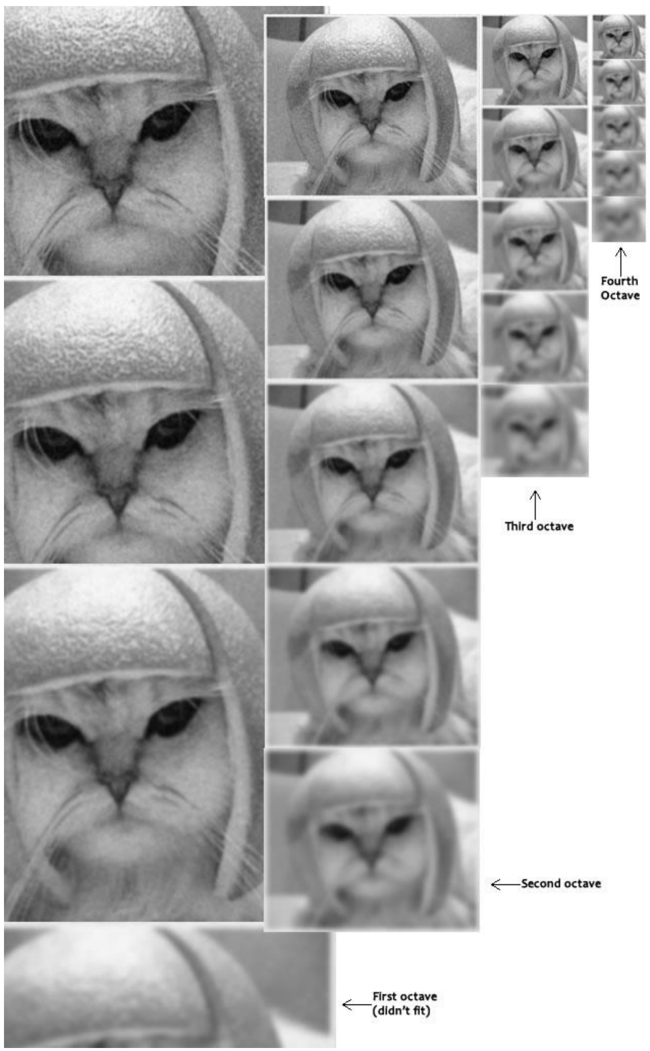
2. Feature Detection – Scale Space Extrema

SIFT constructs a Gaussian image pyramid by smoothing the image by using Gaussian of different scale σ. The smoothing is performed over multiple octaves each of which consists of several images of the same size but with increasing Gaussian scales $k\sigma$.

scale →

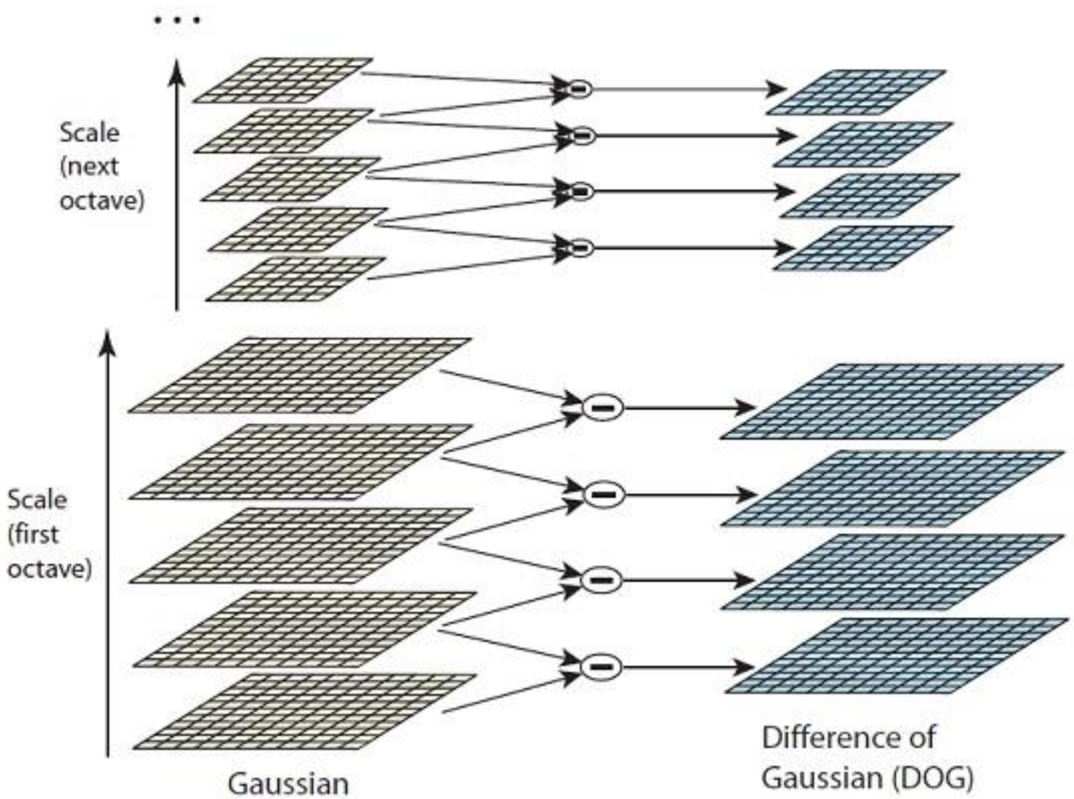
octave				
0.707107	1.000000	1.414214	2.000000	2.828427
1.414214	2.000000	2.828427	4.000000	5.656854
2.828427	4.000000	5.656854	8.000000	11.313708
5.656854	8.000000	11.313708	16.000000	22.627417

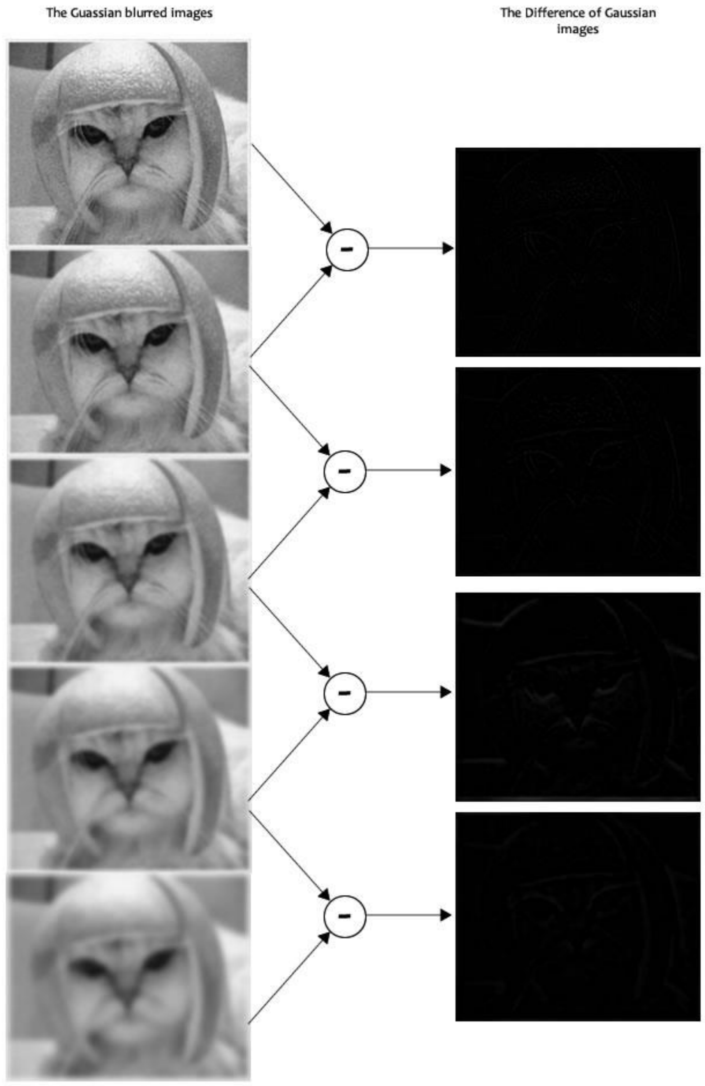
Images in the next-level octave is down-sampled by 50% in both image width and image height, forming a Gaussian pyramid.



2. Feature Detection – Scale Space Extrema

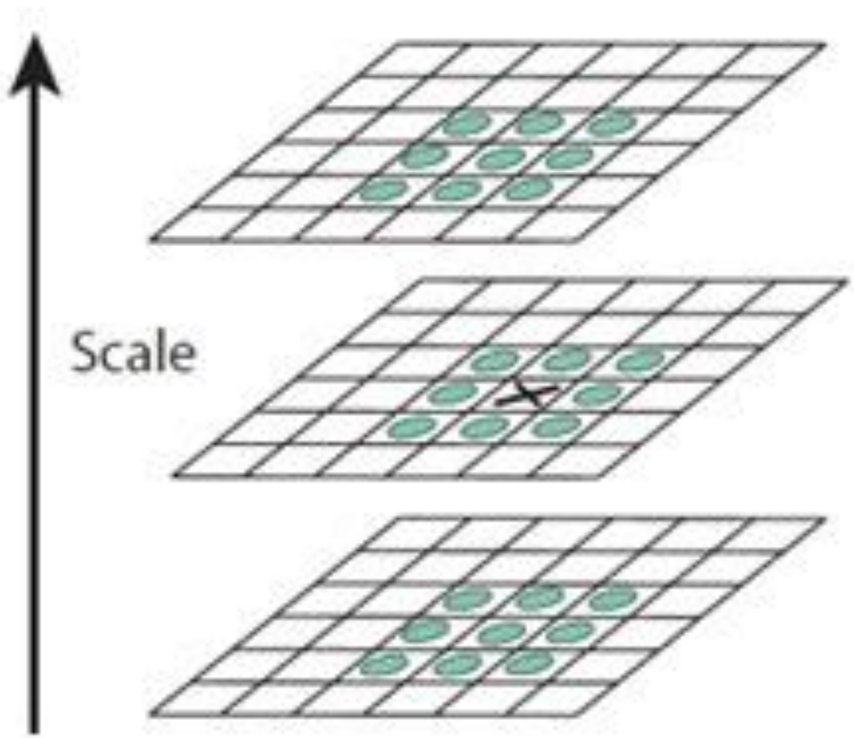
SIFT also constructs a DoG pyramid by computing the DoG for each pair of neighboring images within each octave of the Gaussian pyramid.

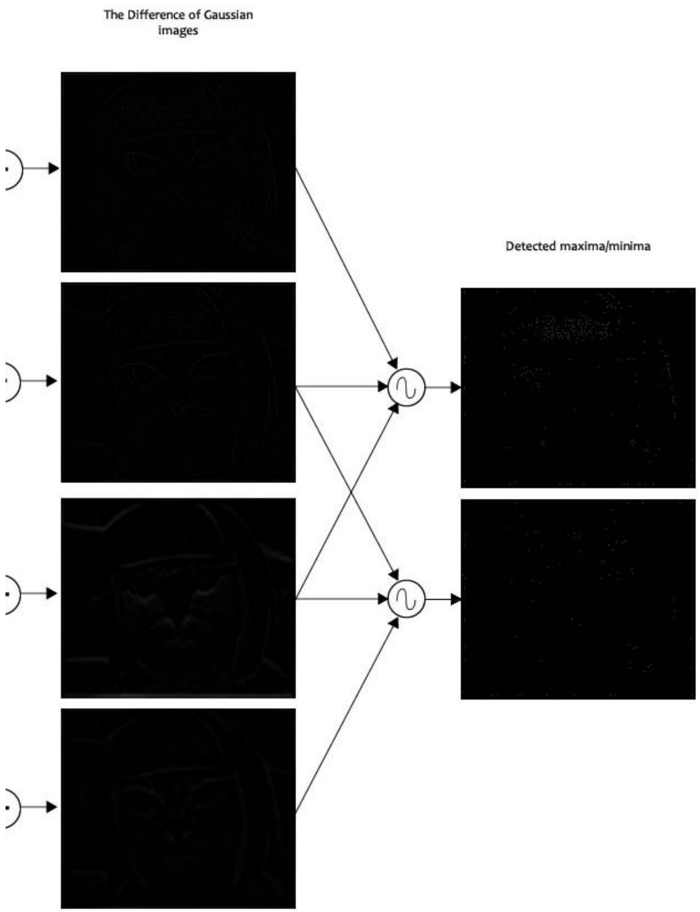




2. Feature Detection – Scale Space Extrema

The extrema (or key points) is detected if it is larger than the 8 neighbours within the same DoG map, as well as the 9 neighbours of the previous and next neighboring DoG maps.



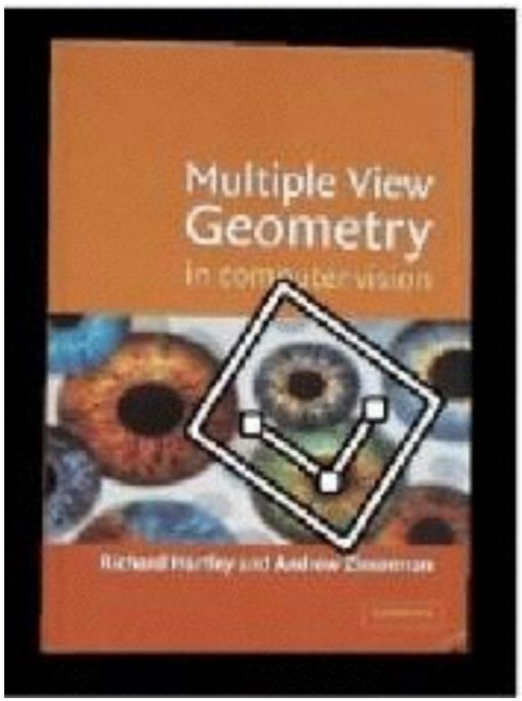


3. Feature Description

If we know where the good features are, how do we match them?

We need good feature descriptor to describe the image patch around feature points. Patches with similar contents should have similar descriptors. Challenges include:

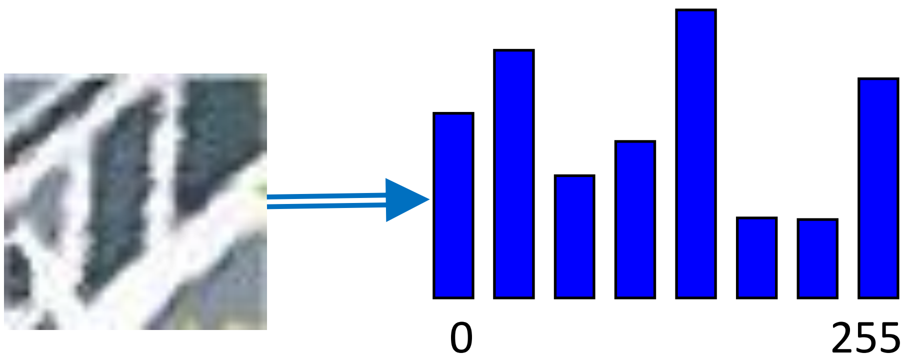
- Photometric transformations: illumination changes
- Geometric transformations: features may have different scales and perspective



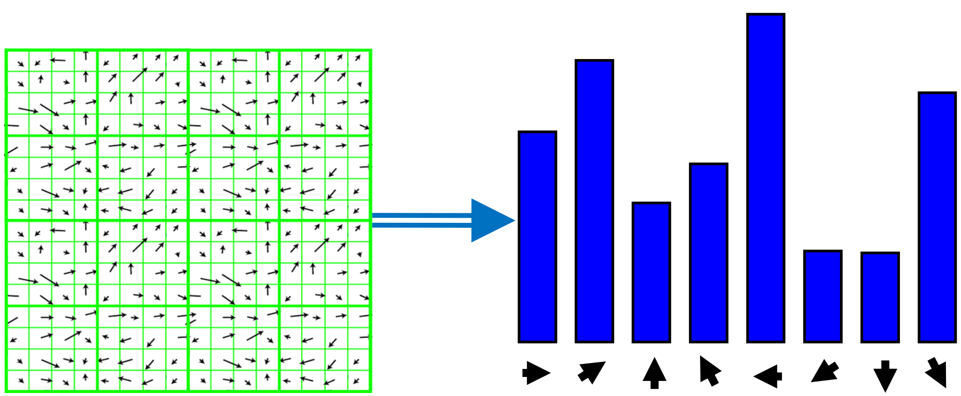
3. Feature Description

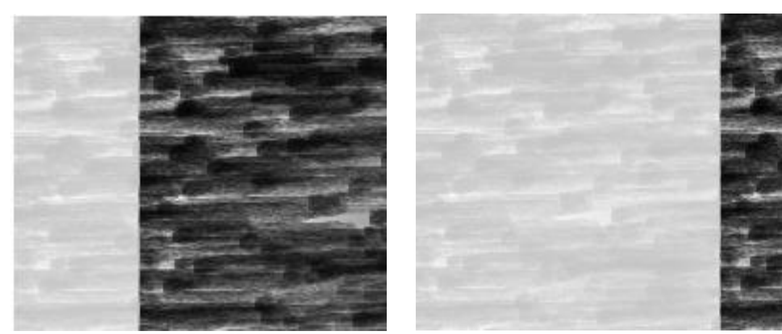
Different descriptors measure different similarity with different invariance in illumination, colors, texture, etc. Different applications require different invariance therefore require different descriptors

Using pixel intensity directly won't work.



Using pixel gradients directly won't work either.

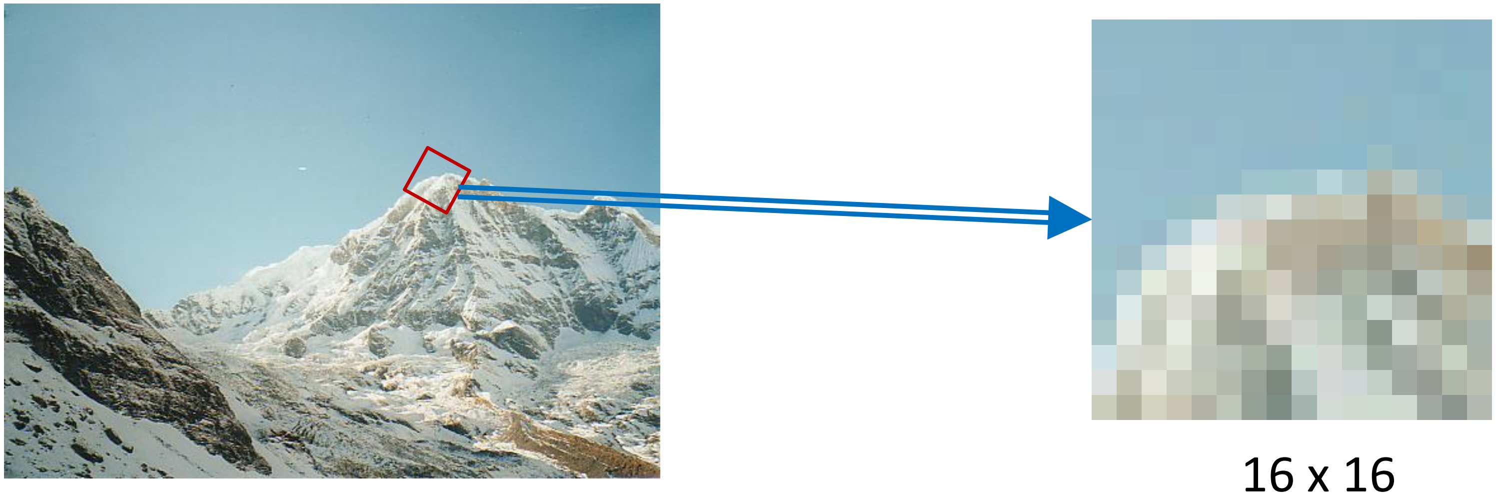




3. Feature Description

SIFT describes a feature point by dividing its neighboring region (a patch) into cells and computing gradients in each cells.

Step 1: Warp the image to the correct orientation and scale, and then extract the feature as 16x16 pixels



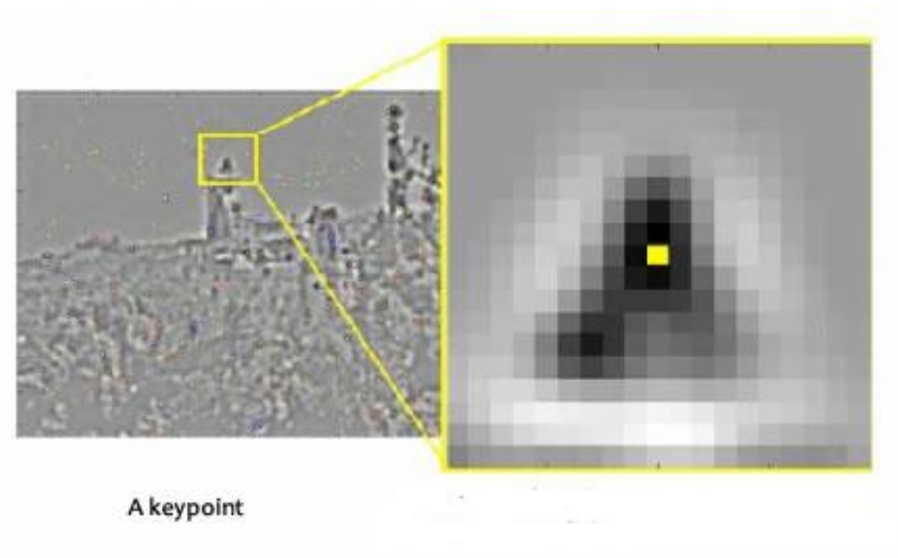
16 x 16

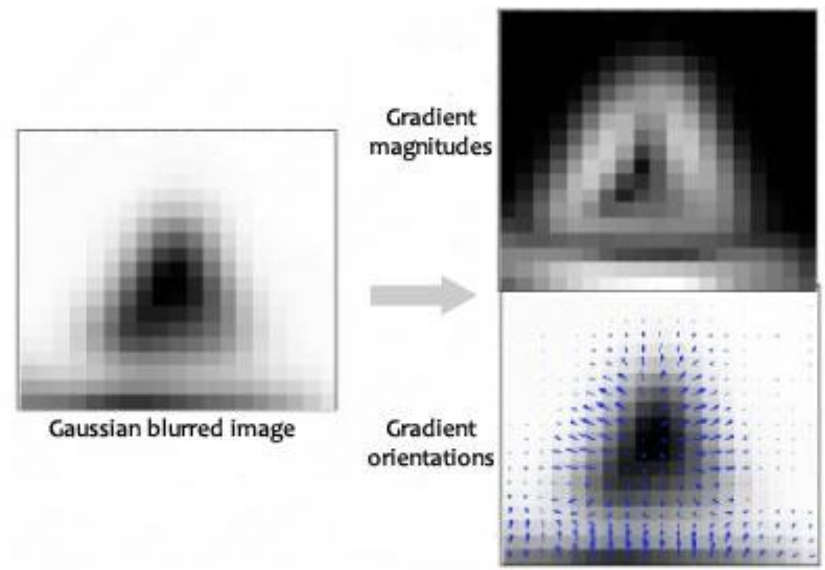
3. Feature Description

Besides scale invariance, SIFT determines a dominant orientation for each valid feature point for achieving rotation invariance.

It is achieved by collecting and binning gradient orientations of neighboring pixels, and select the most prominent (or frequent) one.

The size of the `orientation collection region' depends on the keypoint scale ($1.5 * \sigma$). The bigger the scale, the bigger the collection region.





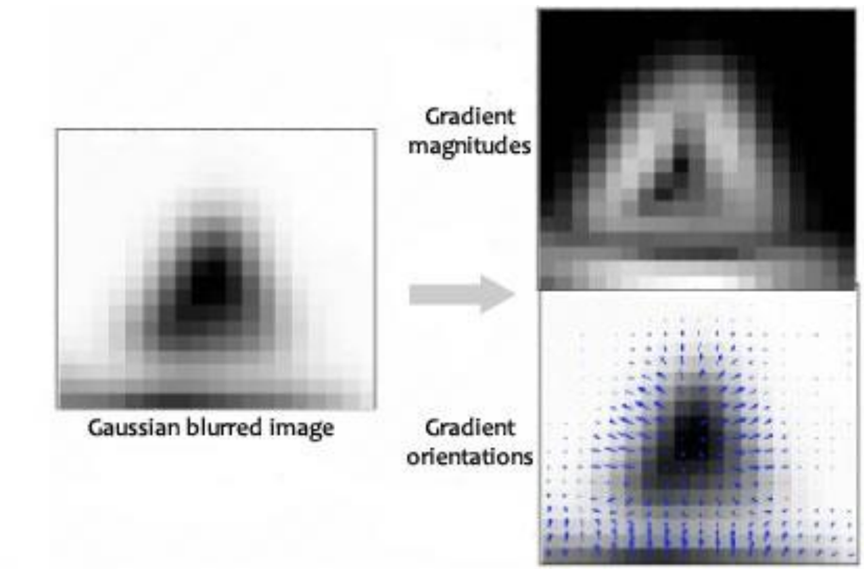
3. Feature Description

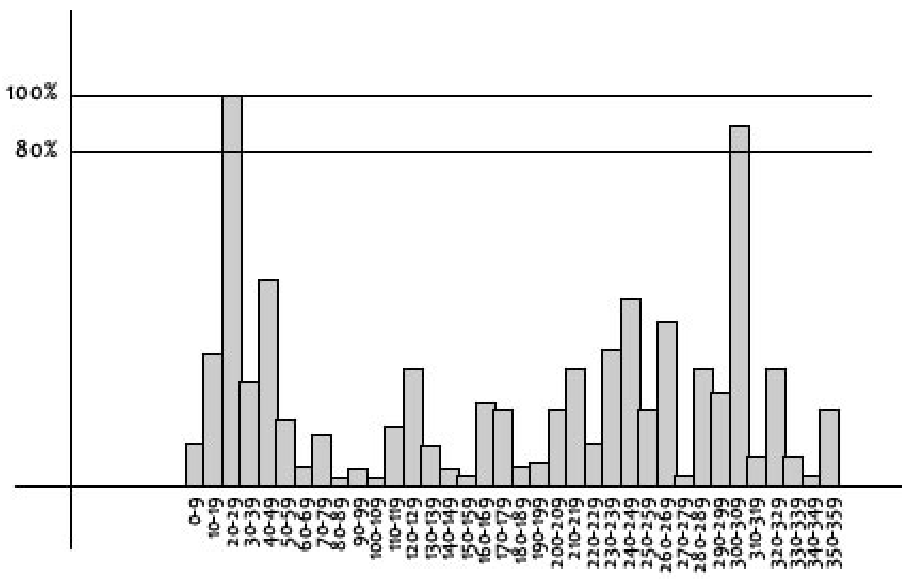
In orientation collection, each neighboring pixel has a gradient amplitude and orientation

$$m(x,y) = \sqrt{\left(L(x+1,y) - L(x-1,y)\right)^2 + \left(L(x,y+1) - L(x,y-1)\right)^2}$$

$$\theta(x,y) = tan^{-1}\left(\left(L(x,y+1) - L(x,y-1)\right) / \left(L(x+1,y) - L(x-1,y)\right)\right)$$

The gradient angle is binned to a histogram, scaled by its amplitude. The extrema orientation is determined by the peak angle.



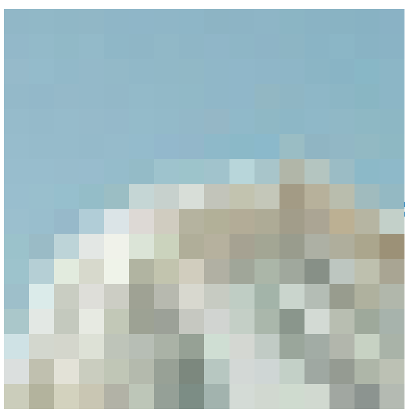


3. Feature Description

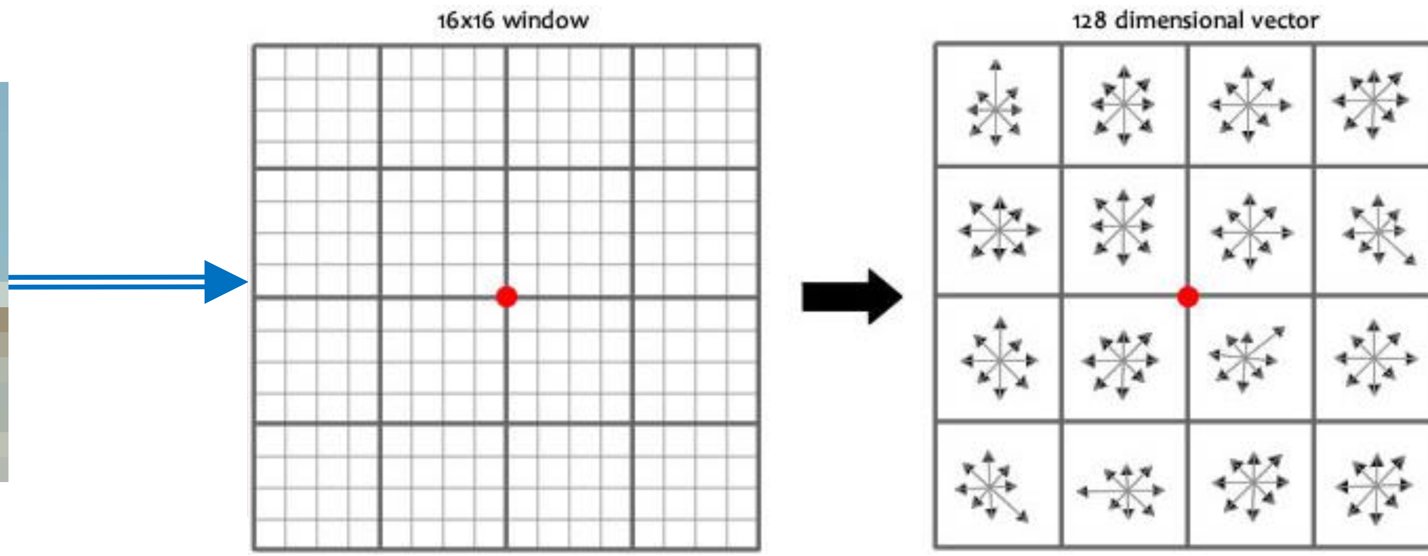
Step 2: Compute the gradient for each pixel (direction and magnitude)

Step 3: Divide the pixels into 16, 4x4 squares

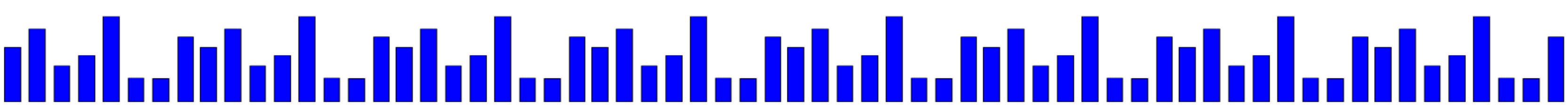
Step 4: For each square, compute gradient direction histogram over 8 directions.



16 x 16

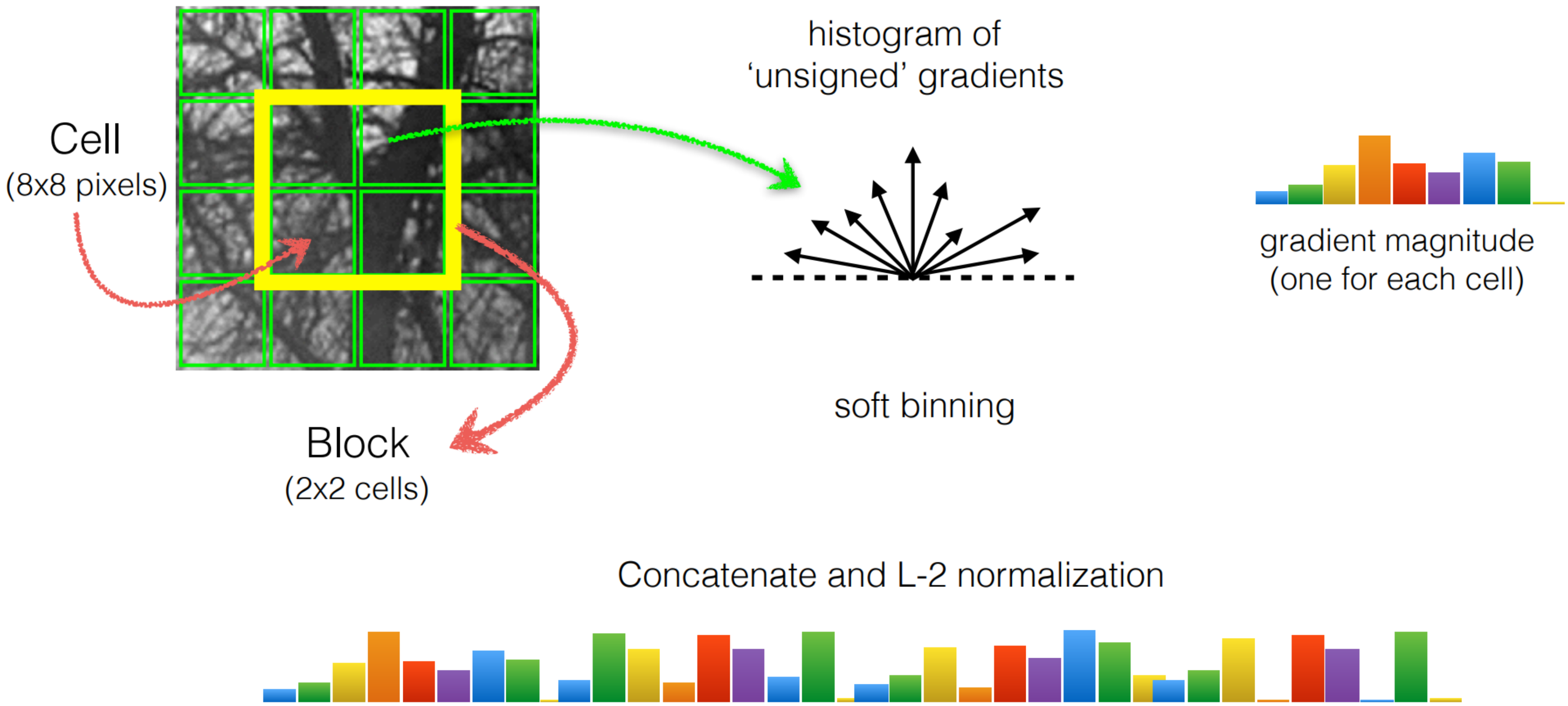


The result: 128 dimensions feature vector.



3. Feature Description

Histograms of Oriented Gradients (HoG) is another widely adopted feature descriptor.



3. Feature Description

It has been successfully applied for human detection.

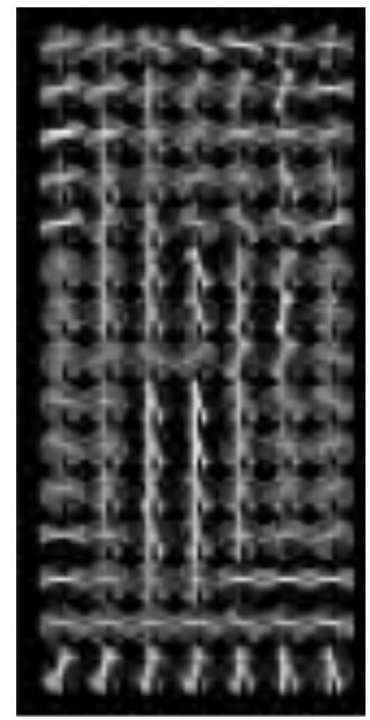
1 cell step size

128 pixels
16 cells
15 blocks

64 pixels
8 cells
7 blocks

15 x 7 x 4 x 9 =
3780

visualization



3. Feature Description

HOG blocks appear quite similar to the SIFT descriptors, but

- HOG blocks are computed in dense grids at some single scale without orientation alignment.
- SIFT descriptors are computed at sparse, scale-invariant key image points and are rotated to align orientation.

4. Feature Matching

Given a database of feature points from interested objects, a simple way of matching feature points from an image is by

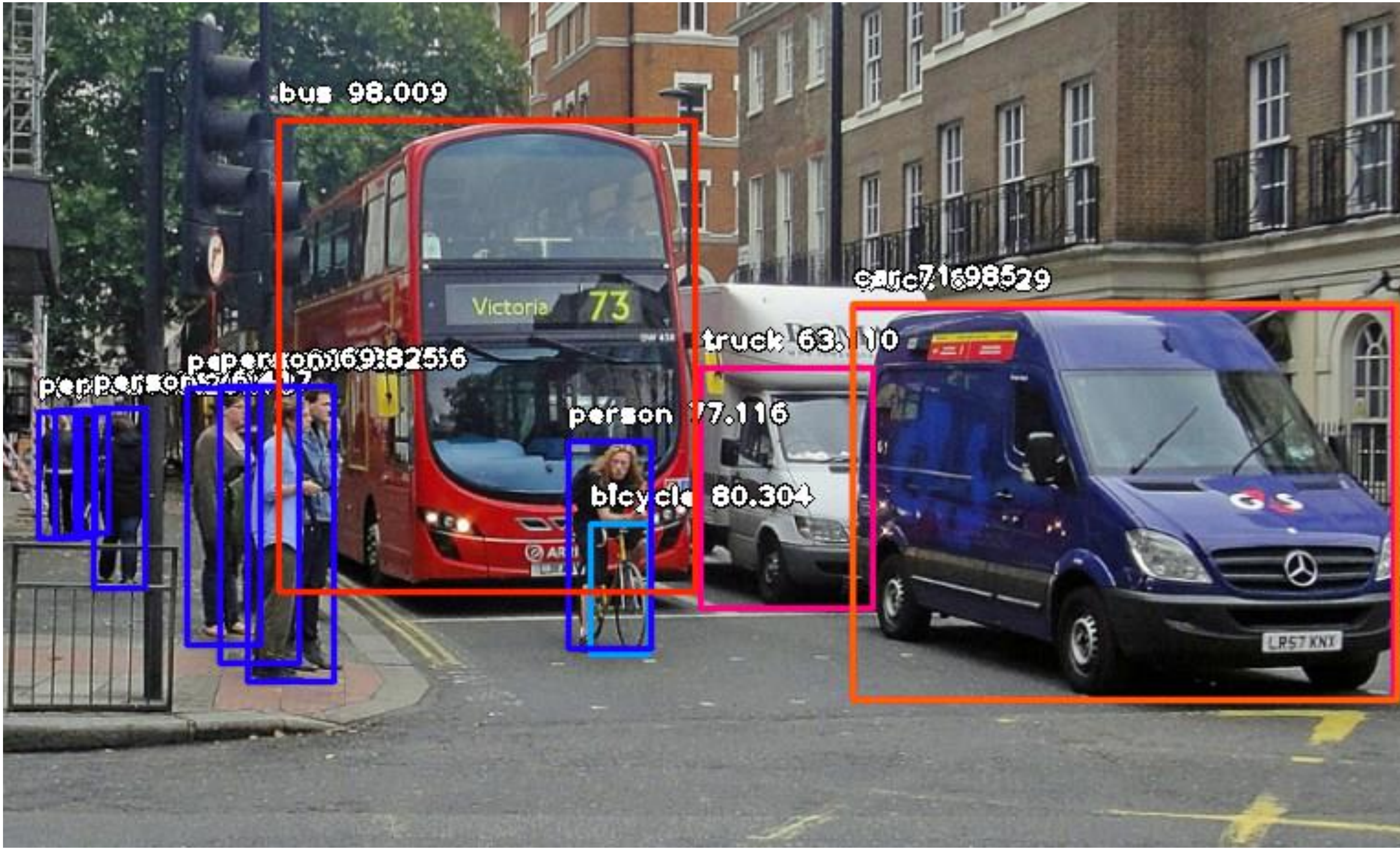
1. Define distance function that compares two descriptors
2. Test all the features in L_2, find the one with the minimum distance
3. Simple approach is $SSD(f_1, f_2)$ (i.e. squared L_2 distance) between entries of the two descriptors

4. Feature Matching

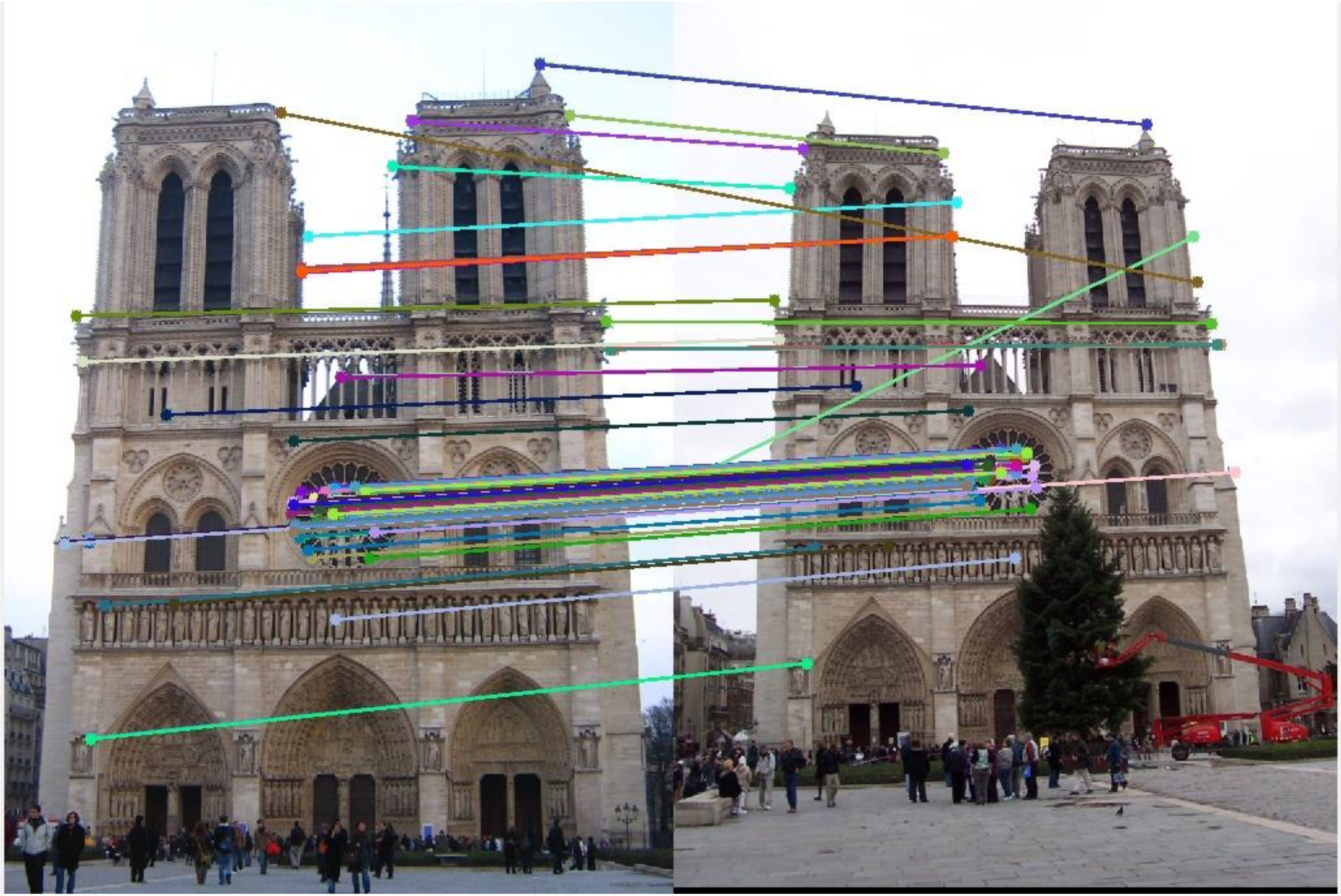
Feature points in an image may be detected from image background which introduces false match. One way to suppress such mismatches is to apply a distance threshold.

However, a global threshold may not work well as different descriptors have different discrimination power. A more effective measure is obtained by comparing the distance of the closest neighbour to that of the second-closest neighbour.

5. Applications – Object Detection



5. Applications – 3D Reconstruction



5. Applications – Image Stitching

Summary

1. Basics about features
2. Feature detection
3. Feature description
4. Feature matching
5. Applications