Feature Detection and Matching 2

Chul Min Yeum
Assistant Professor
Civil and Environmental Engineering
University of Waterloo, Canada

CIVE 497 – CIVE 700: Smart Structure Technology

Last updated: 2020-03-02

UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING

SIFT: Scale-Invariant Feature Transform

Distinctive Image Features from Scale-Invariant Keypoints

David G. Lowe
Computer Science Department
University of British Columbia
Vancouver, B.C., Canada
lowe@cs.ubc.ca

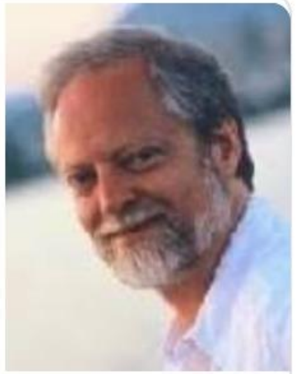
January 5, 2004

Abstract

This paper presents a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. The features are invariant to image scale and rotation, and are shown to provide robust matching across a a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive, in the sense that a single feature can be correctly matched with high probability against a large database of features from many images. This paper also describes an approach to using these features for object recognition. The recognition proceeds by matching individual features to a database of features from known objects using a fast nearest-neighbor algorithm, followed by a Hough transform to identify clusters belonging to a single object, and finally performing verification through least-squares solution for consistent pose parameters. This approach to recognition can robustly identify objects among clutter and occlusion while achieving near real-time performance.

David G. Lowe

Canadian computer scientist



David G. Lowe is a Canadian computer scientist working for Google as a Senior Research Scientist. He was a former professor in the Computer Science Department at the University of British Columbia and New York University. Wikipedia

Known for: Scale-invariant feature transform

Residence: Seattle, Washington, United States

Alma maters: The University of British Columbia, Stanford University (1985, PhD)

Academic advisor: Thomas Binford

Notable student: Ken Perlin

Steps for Extracting SIFT Keypoints

Scale-space extrema detection: Difference-of-Gaussian function is used to identify potential interest points that are invariant to scale and orientation.

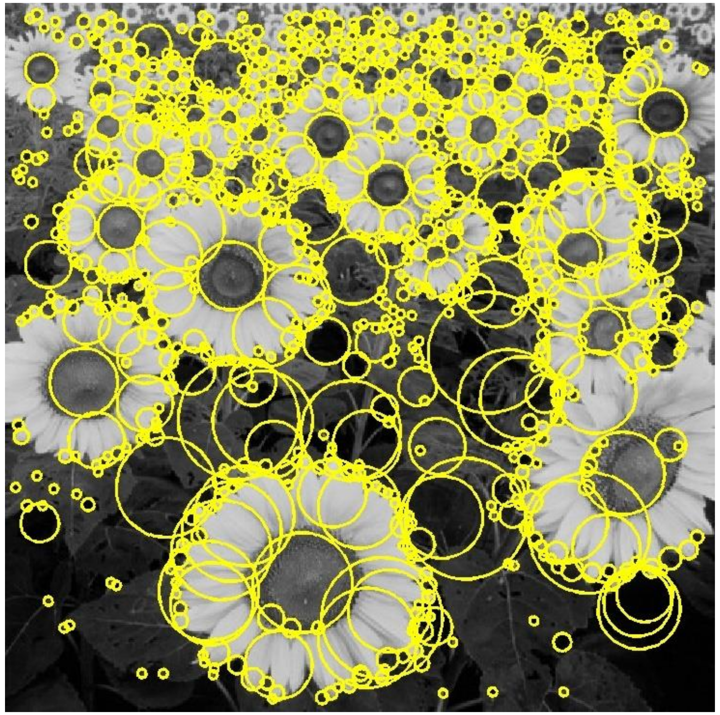
Keypoint localization: At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.

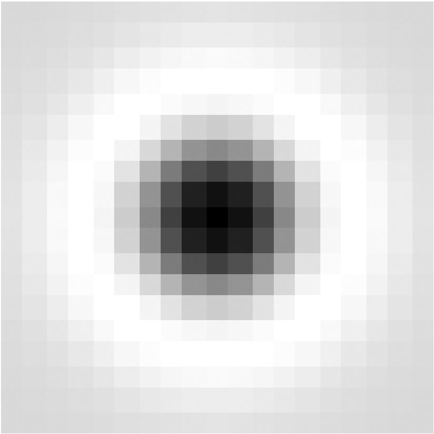
Orientation assignment: One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.

Keypoint descriptor: The local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

Basic Idea

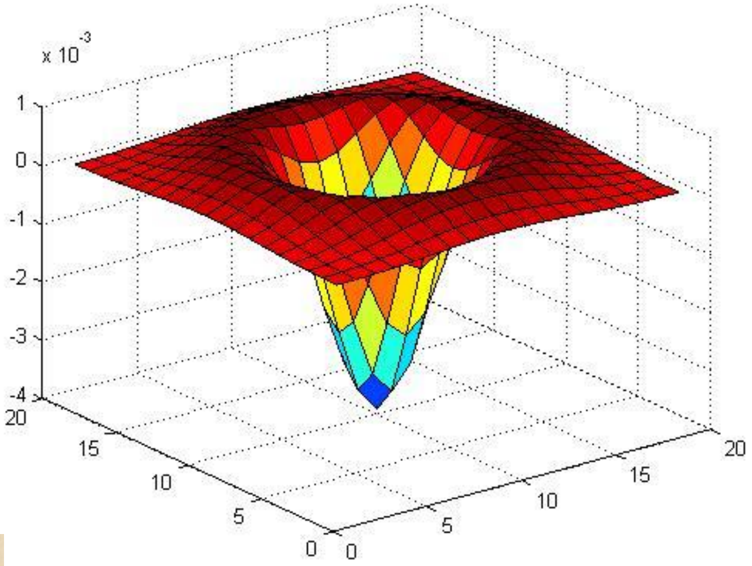
- Convolve the image with a “blob filter” at multiple scales and look for extrema of filter response in the resulting *scale space*

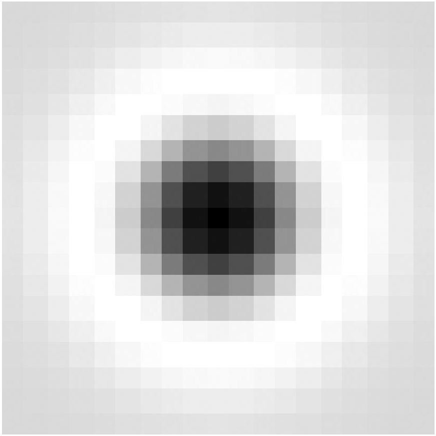




Blob Filter

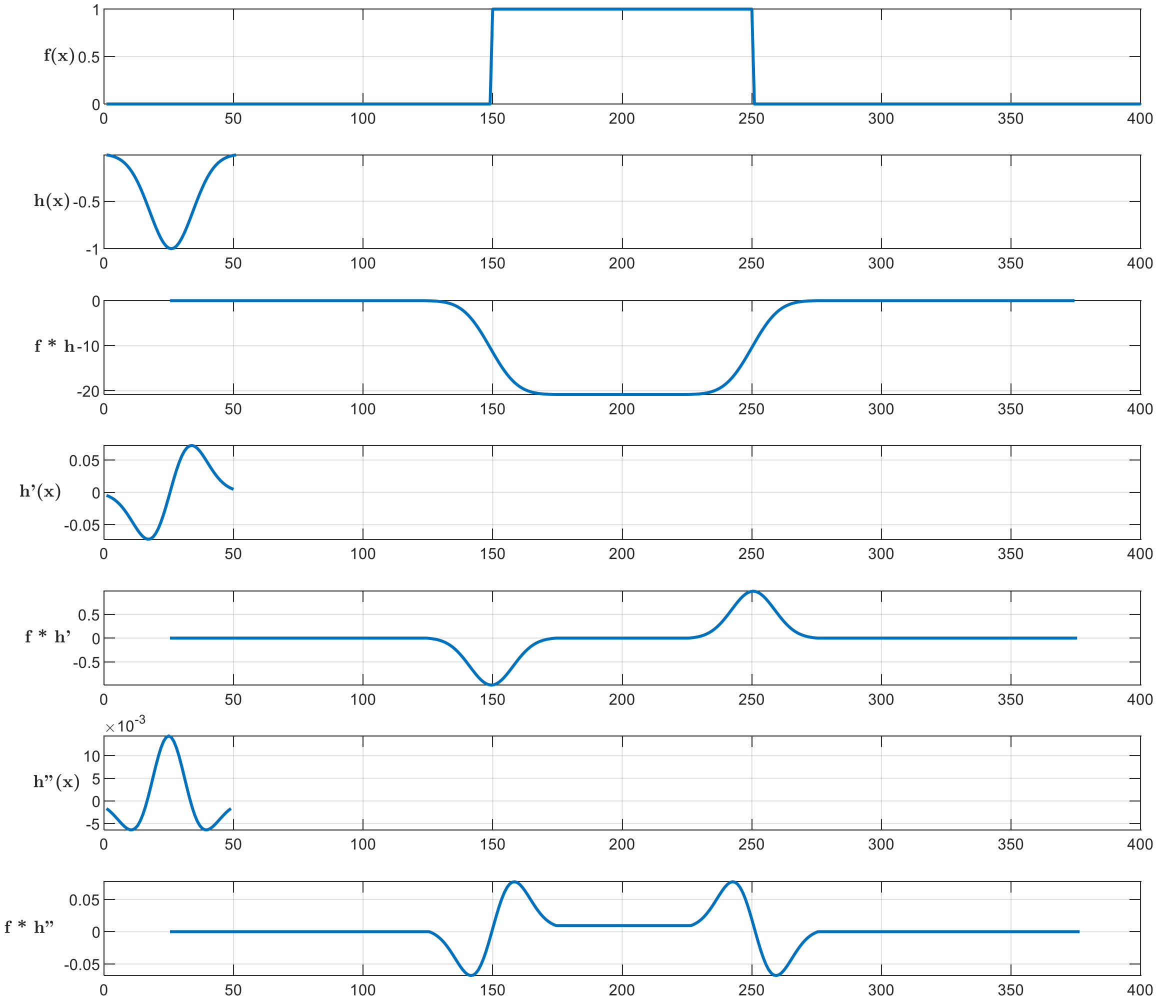
- Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

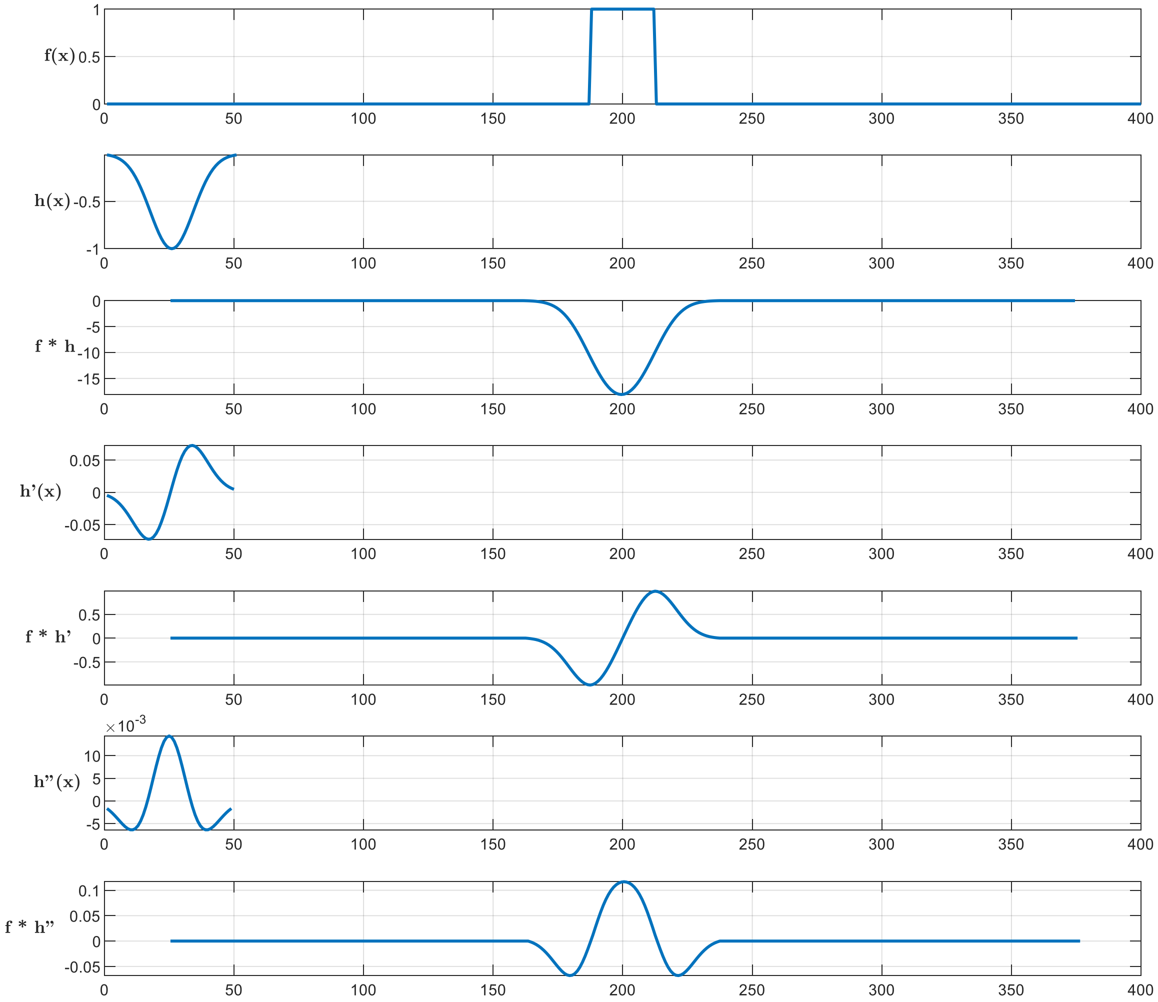




$$\nabla^2 G = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$

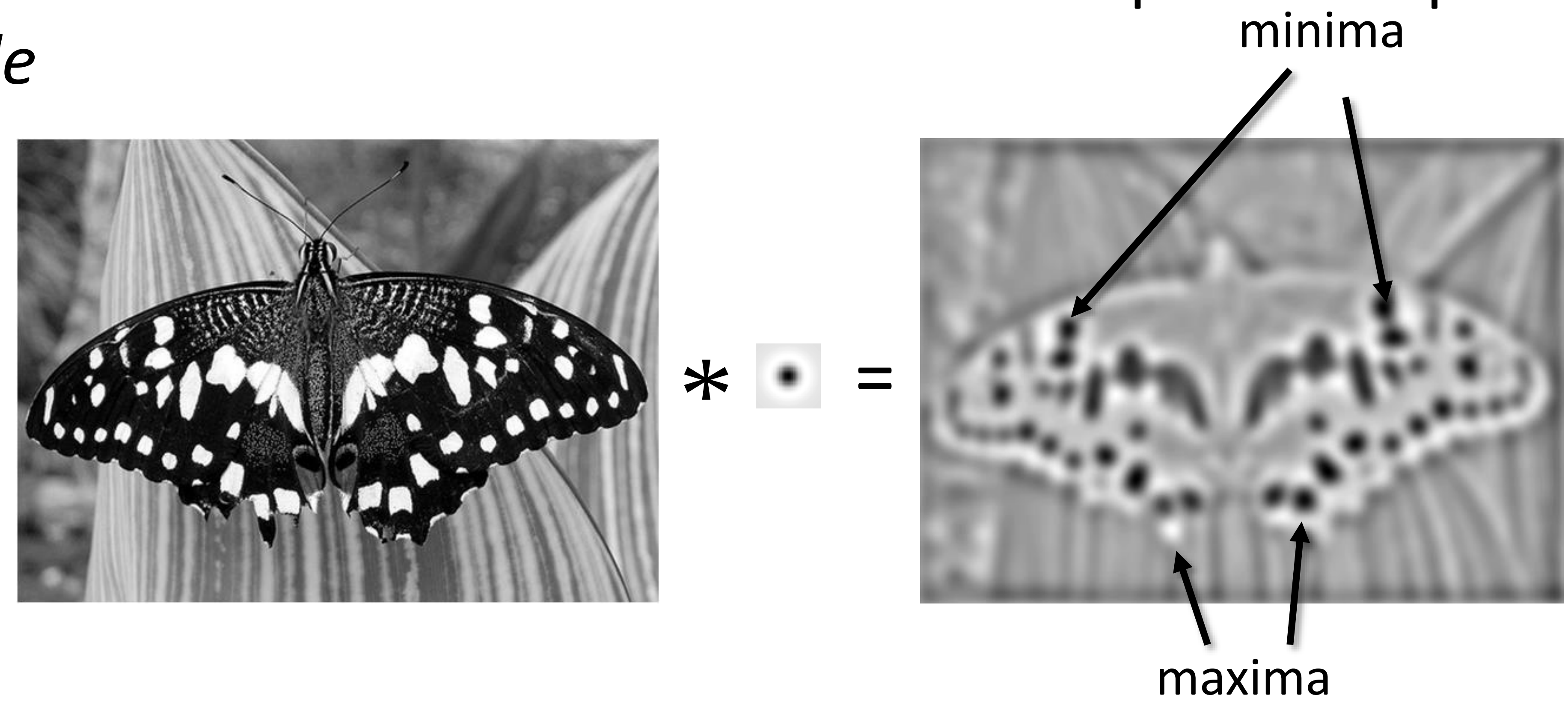
Example: Blob Filter





Blob Detection

- Find maxima *and minima* of blob filter response in space *and scale*



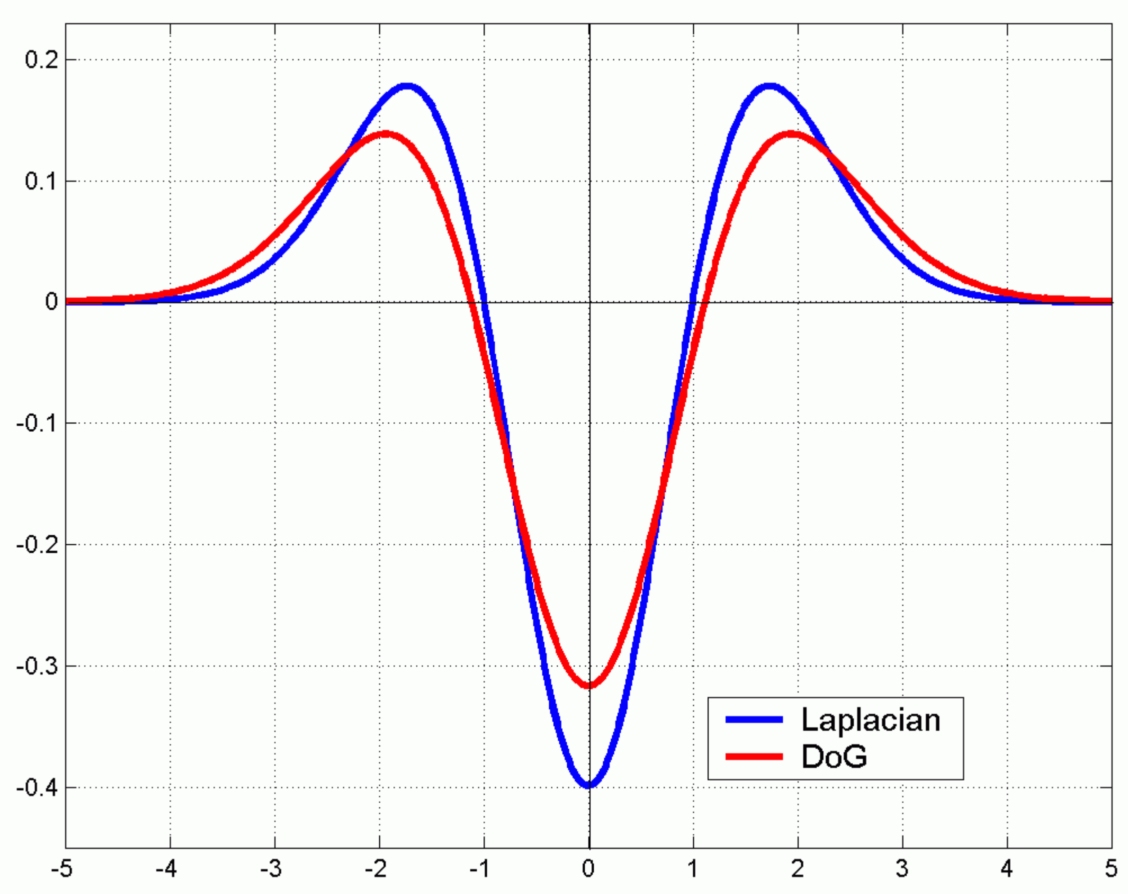
Efficient Implementation

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

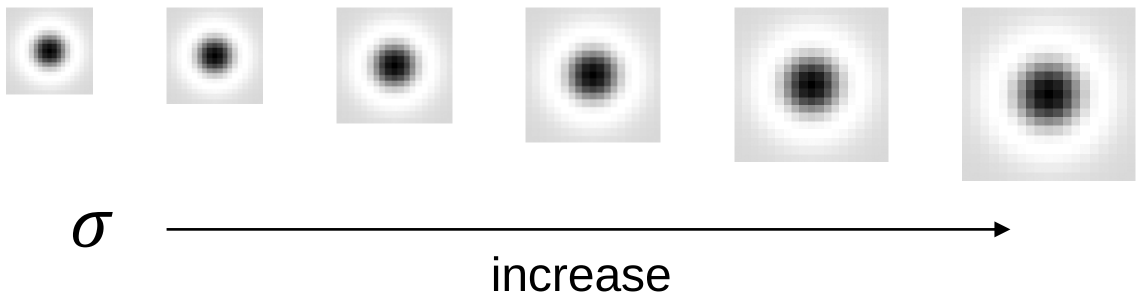
(Laplacian)

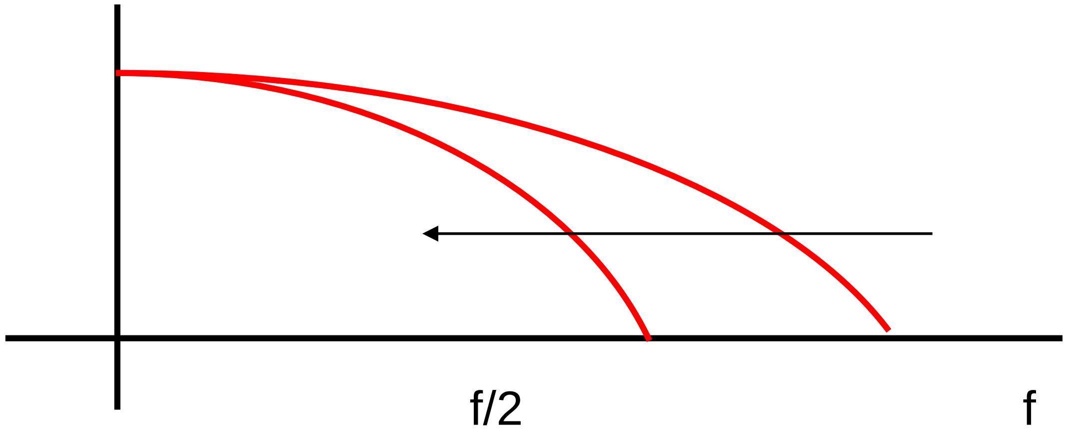
$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

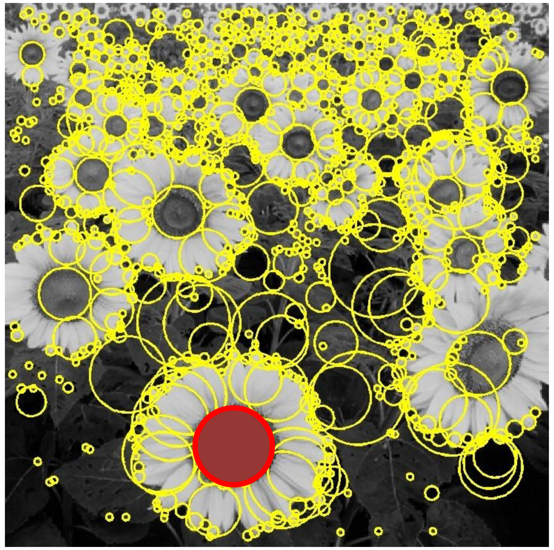
(Difference of Gaussians)

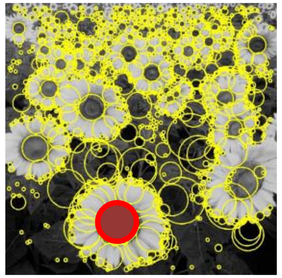


Octave

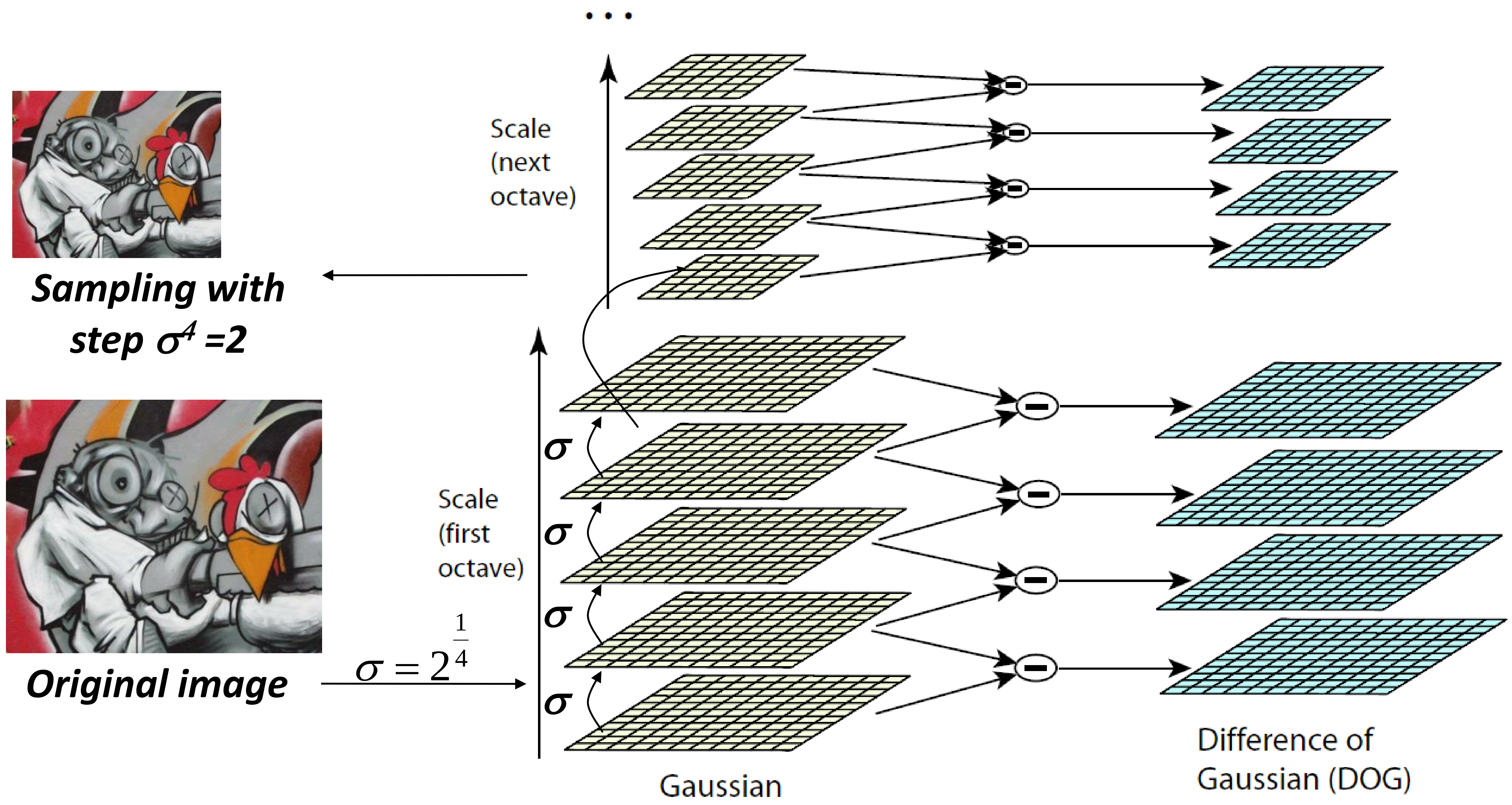








Gaussian Scale Pyramid



Example: Sunflower

Gaussian Smoothing

sigma = sigma0 * 1.270

sigma = sigma0 * 1.600

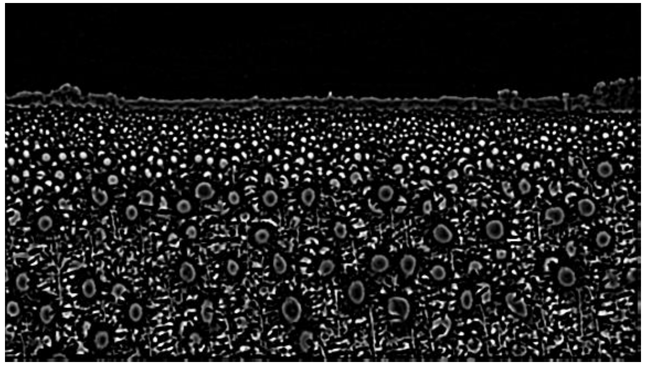
sigma = sigma0 * 2.016

sigma = sigma0 * 2.540

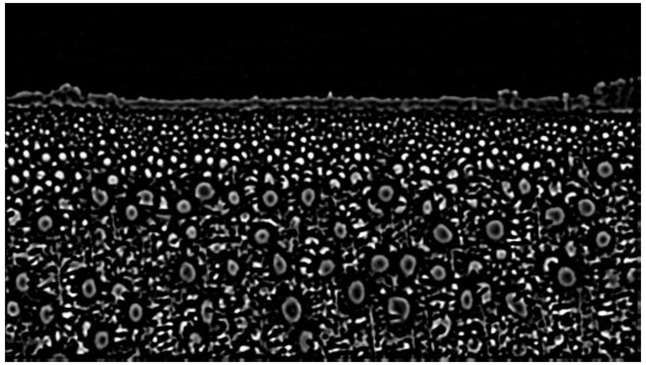
sigma = sigma0 * 3.200

Difference of Gaussian

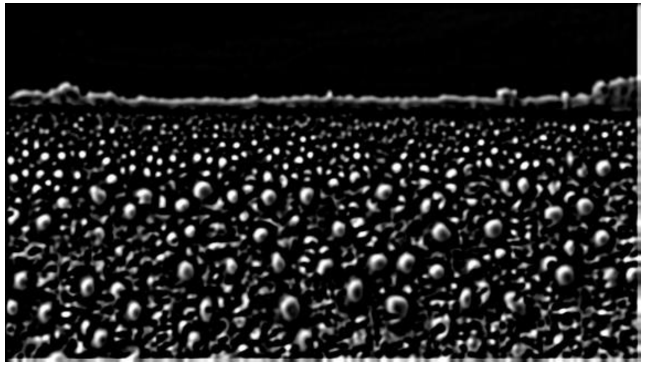
Sigma0*1.270



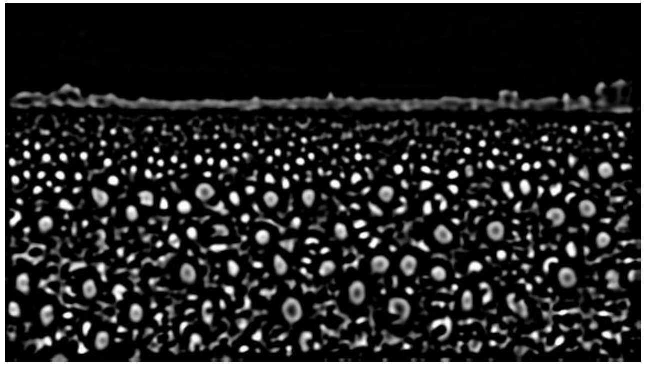
Sigma0*1.6



Sigma0*2.0



Sigma0*2.5



Detection of scale-space extrema

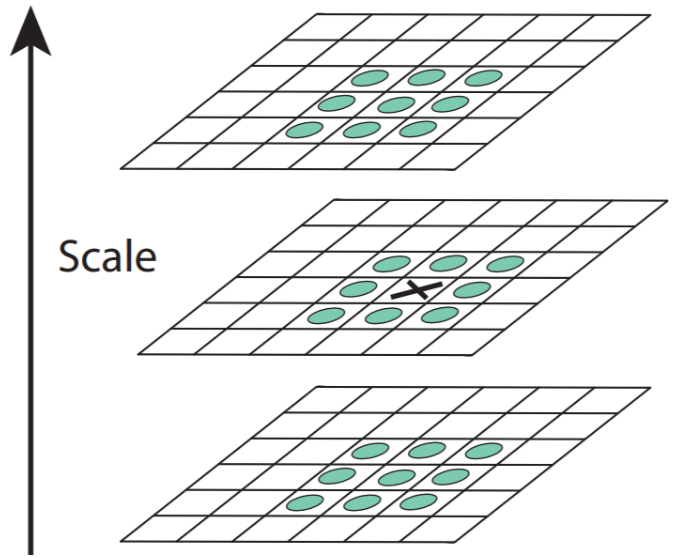
The scale space of an image is defined as a function, $L(x,y,\sigma)$, that is produced from the convolution of a variable-scale Gaussian, $G(x,y,\sigma)$, with an input image, $I(x,y)$:

where * is the convolution operation in *x* and *y*, and

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

Difference-of-Gaussian function:

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



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

Orientation Assignment

By assigning a consistent orientation to each keypoint based on local image properties, the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation.

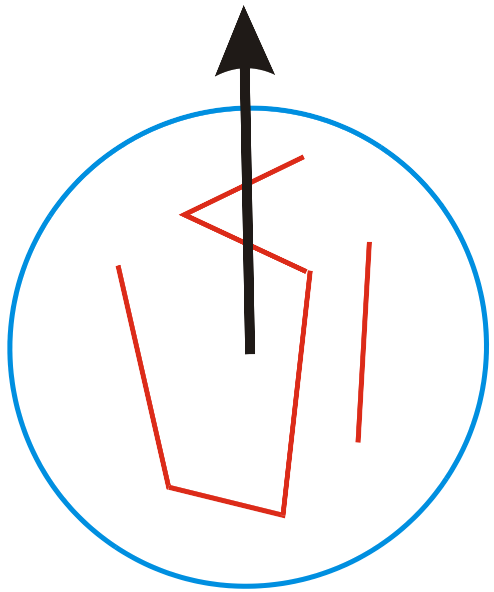
For each image sample, $L(x,y,\sigma)$, at this scale, the gradient magnitude, $m(x,y)$, and orientation, $\theta(x,y)$, is precomputed using pixel differences:

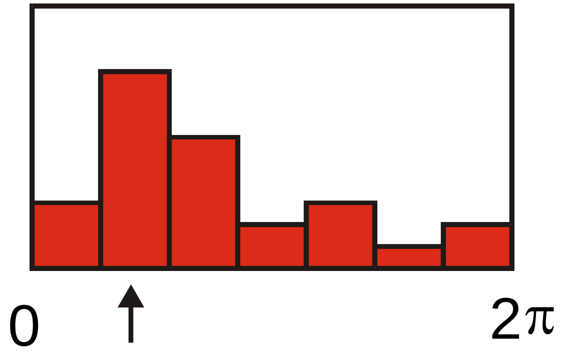
$$m(x,y) = \sqrt{(L(x+1,y)-L(x-1,y))^2 + (L(x,y+1)-L(x,y-1))^2}$$

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

Orientation Normalization

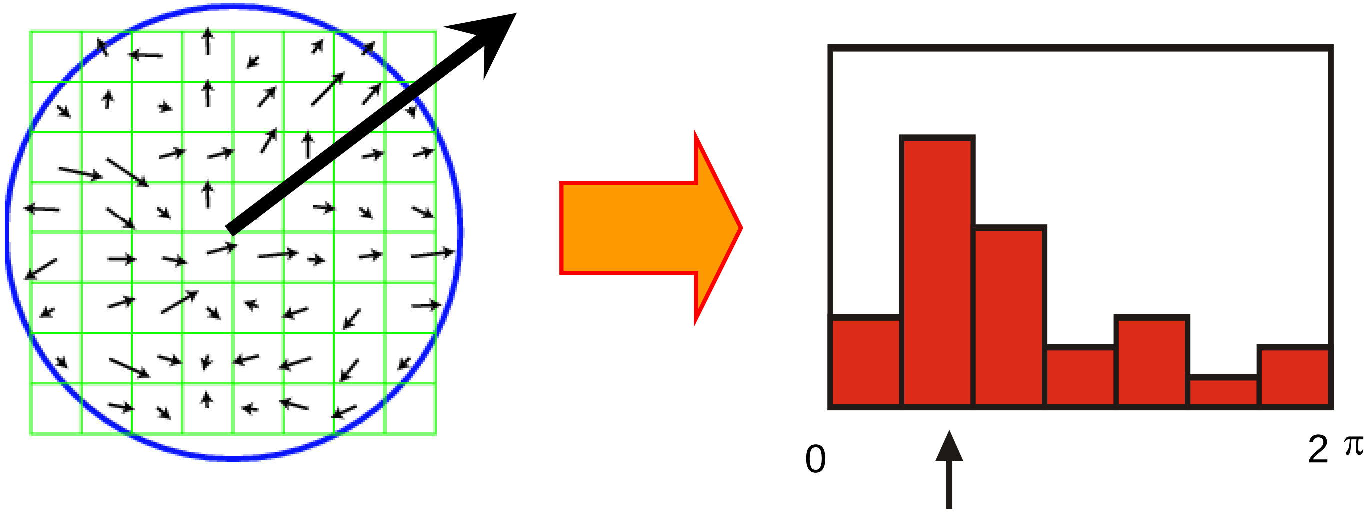
- Compute orientation histogram
- Select dominant orientation
- Normalize: rotate to fixed orientation





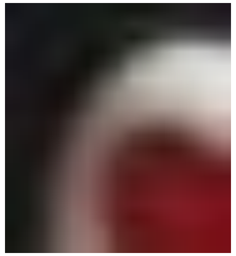
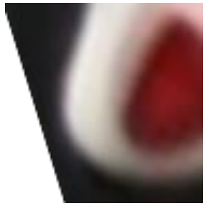
Eliminating Rotation Ambiguity

- 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

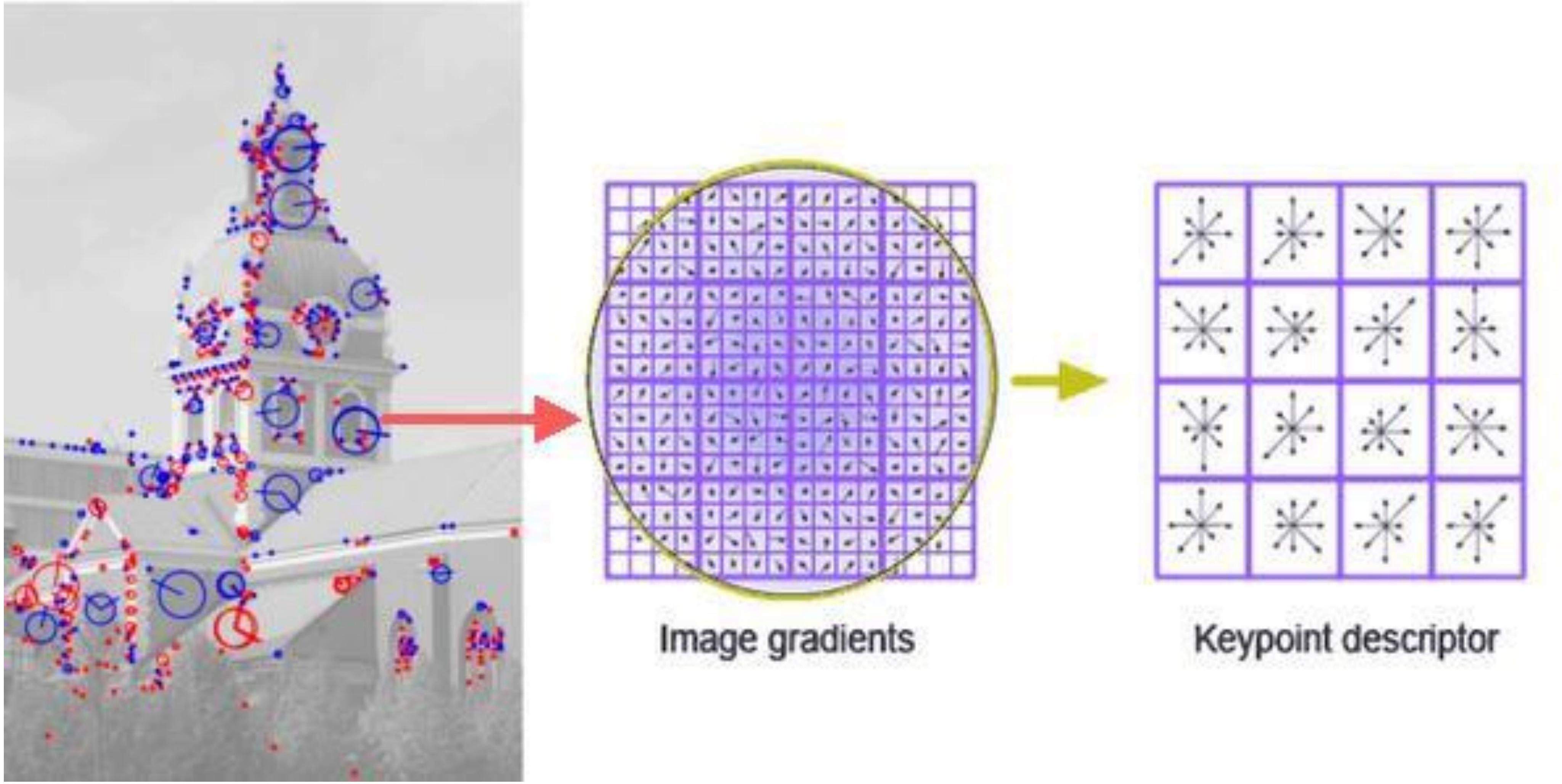


Example: Orientation

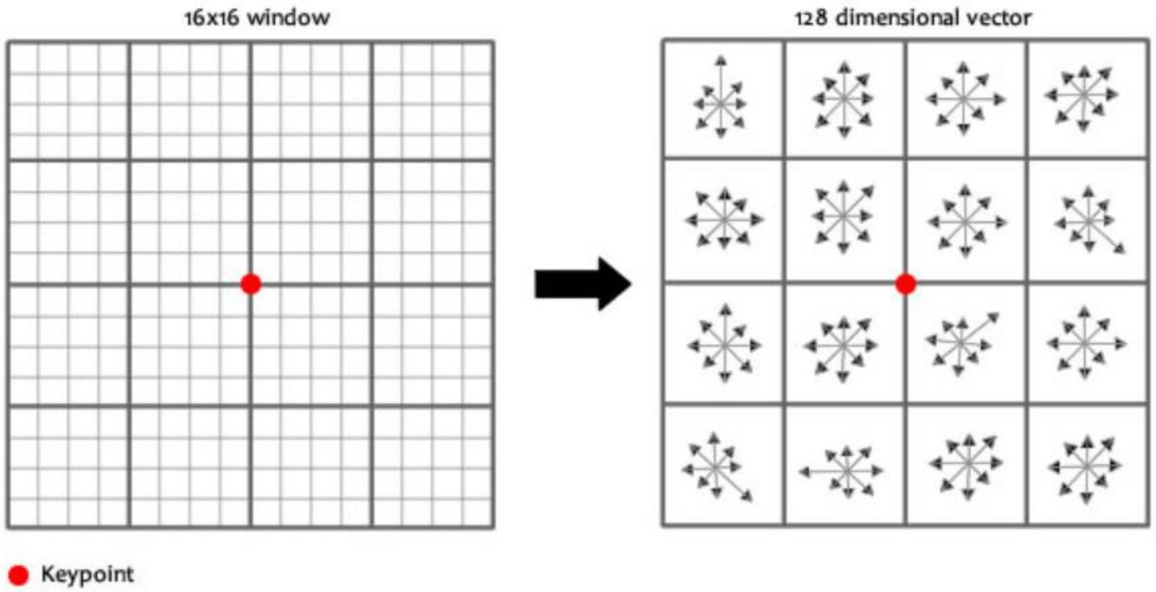


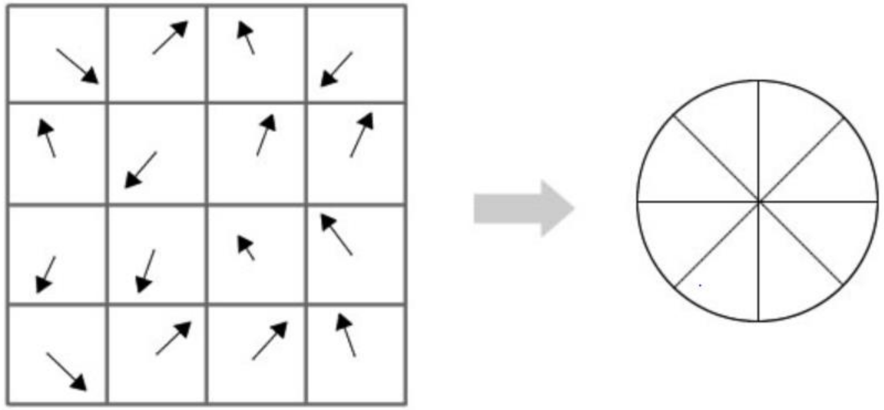



Example: Features and Descriptors



SIFT Descriptor





Any gradient

Feature Matching Procedure

- **Each extracted feature has a 128-element descriptor vector assigned to it.**
- **The Euclidean distances between each feature's descriptor vector in the *reference* image and any of the feature descriptor vectors in the *input* image are computed.**
- **If** $\frac{\text{distance between a feature in the reference image and its nearest feature in the input image}}{\text{distance between a feature in the reference image and its } 2^{nd} \text{ nearest feature in the input image}} < \tau,$ **the nearest feature is accepted as the matching feature otherwise the feature in the reference image does not have a match. Where τ is a threshold**

SIFT Library

F = VL_SIFT(I) computes the SIFT frames [1] (keypoints) F of the image I. I is a gray-scale image in single precision. Each column of F is a feature frame and has the format [X;Y;S;TH], where X,Y is the (fractional) center of the frame, S is the scale and TH is the orientation (in radians).

[F,D] = VL_SIFT(I) computes the SIFT descriptors [1] as well. Each column of D is the descriptor of the corresponding frame in F. A descriptor is a 128-dimensional vector of class UINT8.

VL_SIFT() accepts the following options:

http://www.vlfeat.org/overview/sift.html

Slide Credits and References

- Lecture notes: S. Narasimhan
- Lecture notes: Gordon Wetzstein
- Lecture notes: Mohammad Jahanshahi
- Lecture notes: Noah Snavely
- Lecture notes: L. Fei-Fei
- Lecture notes: D. Frosyth
- Lecture notes: James Hayes
- Lecture notes: Yacov Hel-Or
- Lecture notes: K. Grauman, B. Leibe