Last time..

DRL

Today:

Feature Engineering 1: Points: feature detection, description and matching

Harris, SIFT and RANSAC

Udacity: Introduction to Computer Vision

Agenda

- «Good» / Interest / Key **points** to find:
    - Flat, Edge and Corner
- **Harris** Corner Detection:
    - Taylor, Second moment matrix & Response function
- **SIFT**:
    - Taylor, Second moment matrix & Response function
- **RANSAC**:
    - Sample, model, inliers / outliers

Choosing interest points

Where would you tell your friend to meet you?

Feature point:
- detection
- description
- matching

Slide Credit: James Hays

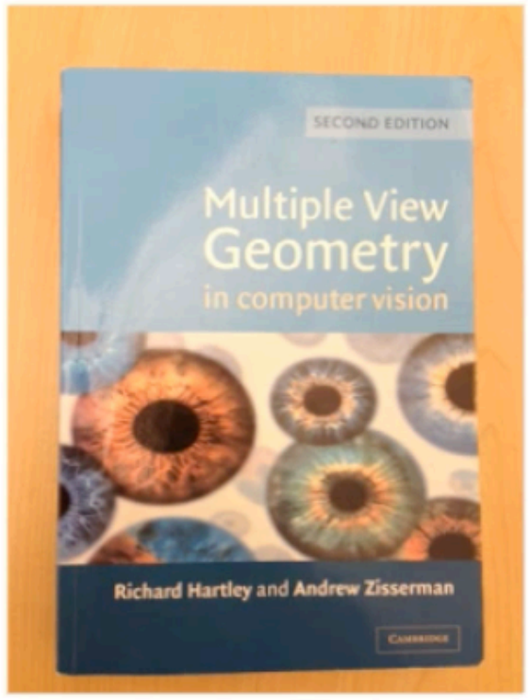
Features

- Automate object tracking
- Point matching for computing disparity
- Motion based segmentation
- Object Recognition
- 3D Object Reconstruction
- Robot Navigation
- Image Retrieval/Indexing
- ….

Feature point detection / description / matching use:
- Template matching / object retrieval (ex)
- Image stitching / Panoramas
- Stereo / Dept estimation / Correspondence
- Tracking / from frame to frame
- etc.

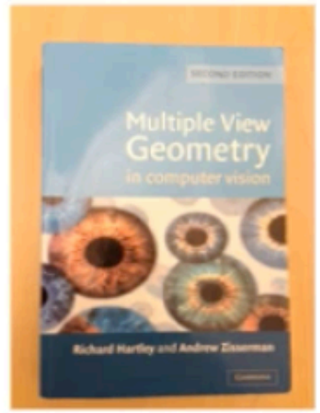
Object retrieval / template matching:
- Finding lost items (books)
- Finding criminals (faces)
- etc.

Demo: Locate an Object in an Image of a Cluttered Scene

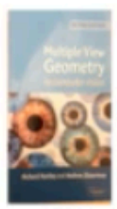


Challenges

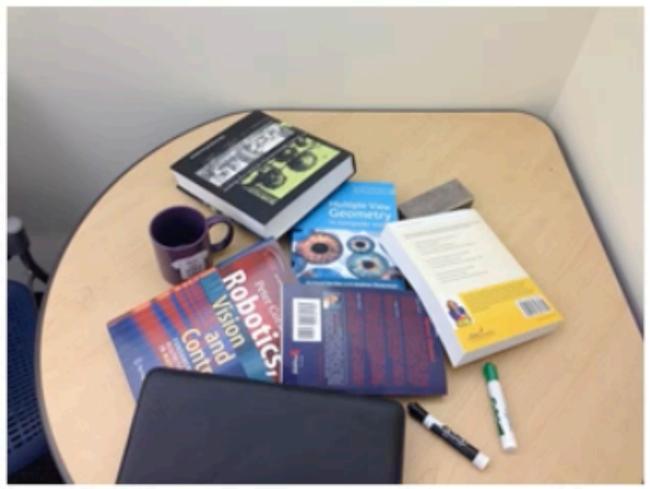
- E.g. Find a lost book on my desk



Rotation

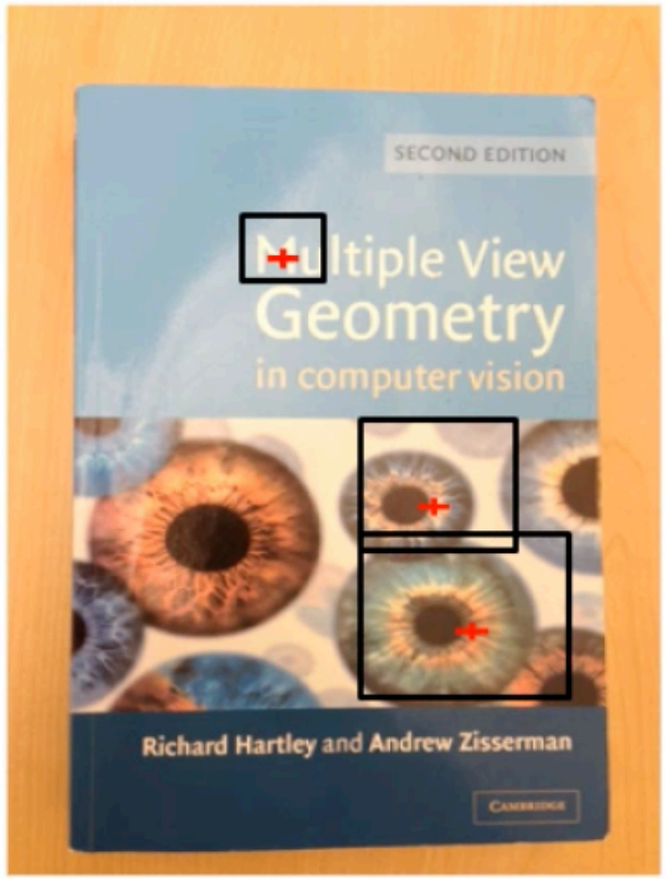


Scale

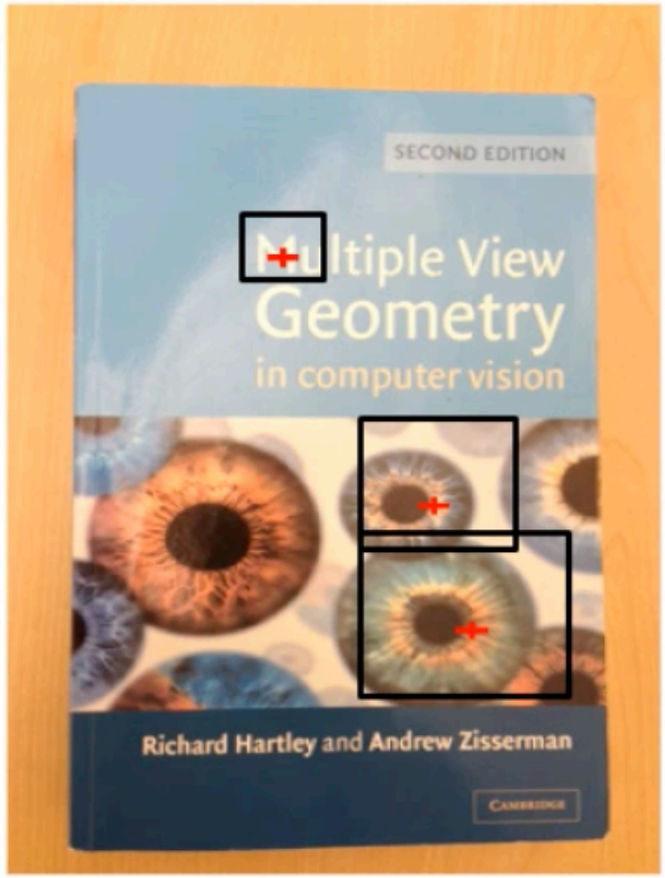


Clutter

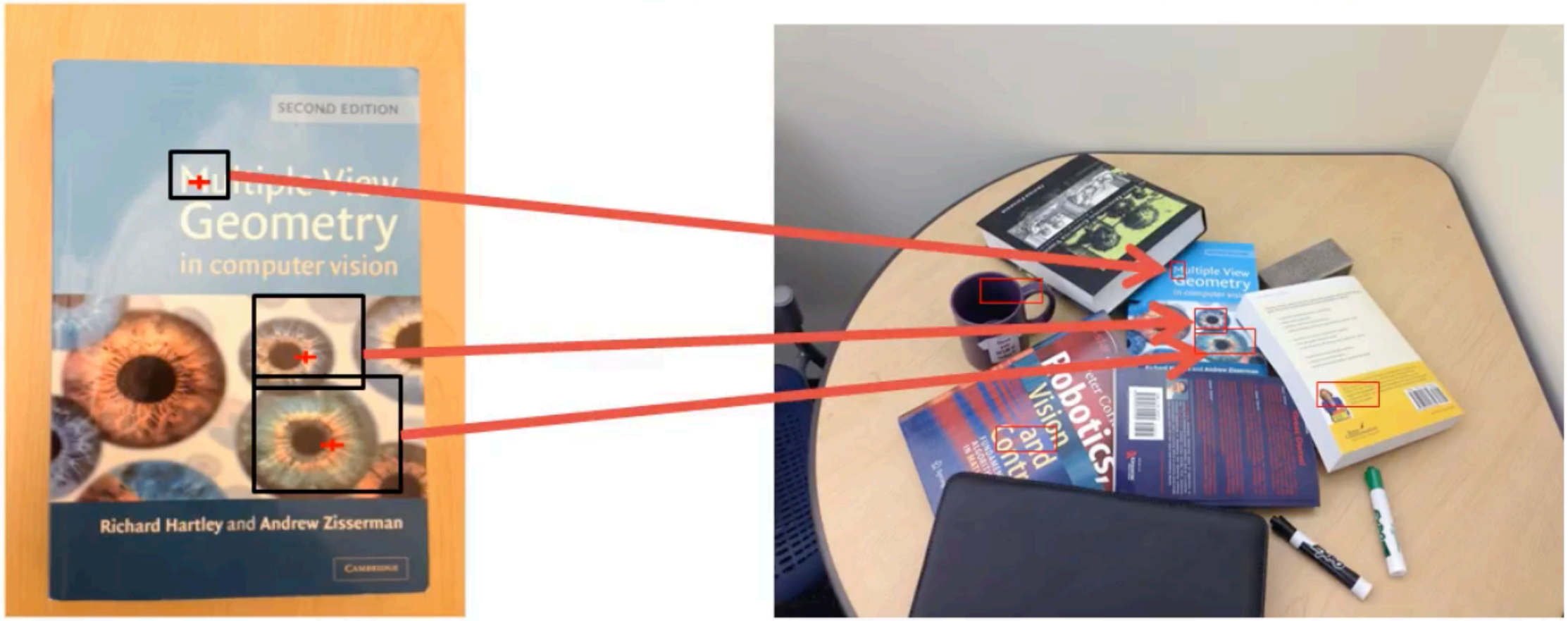
Solution: Feature Detection, Extraction and Matching



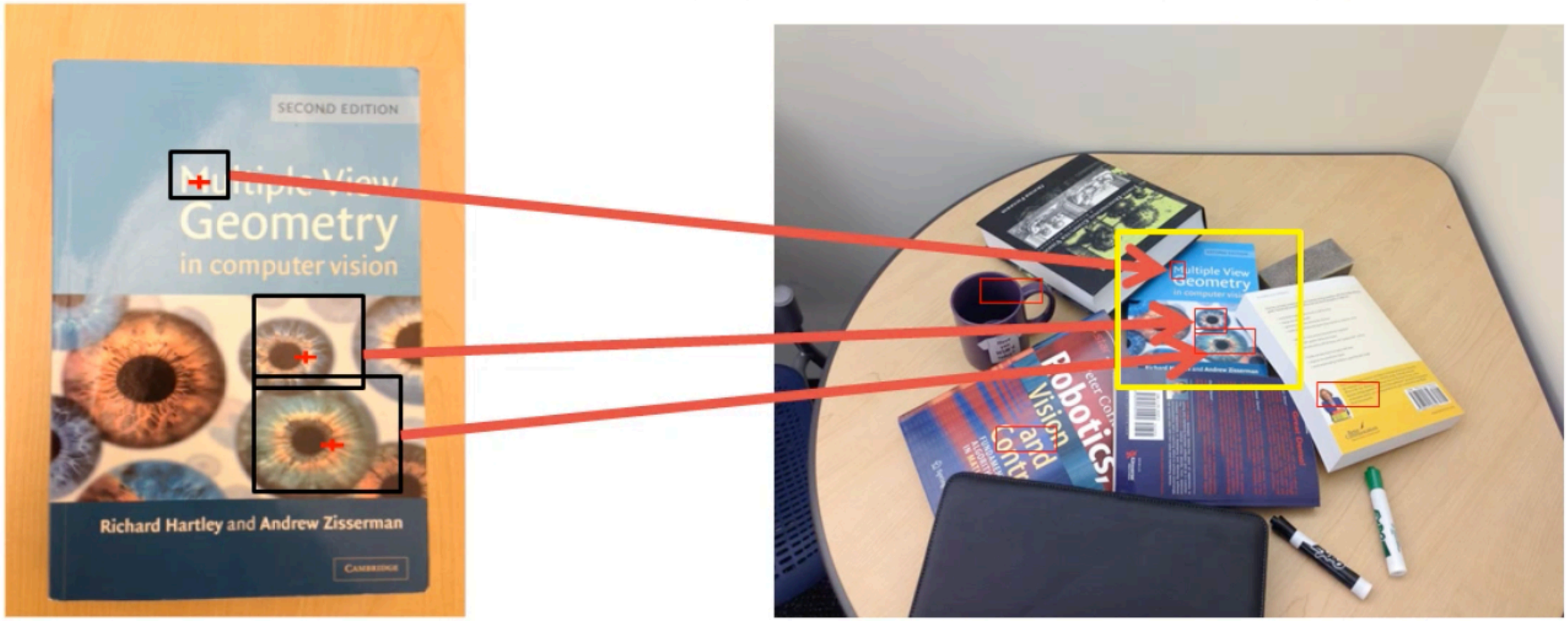
Solution: Feature Detection, Extraction and Matching



Solution: Feature Detection, Extraction and Matching



Solution: Feature Detection, Extraction and Matching

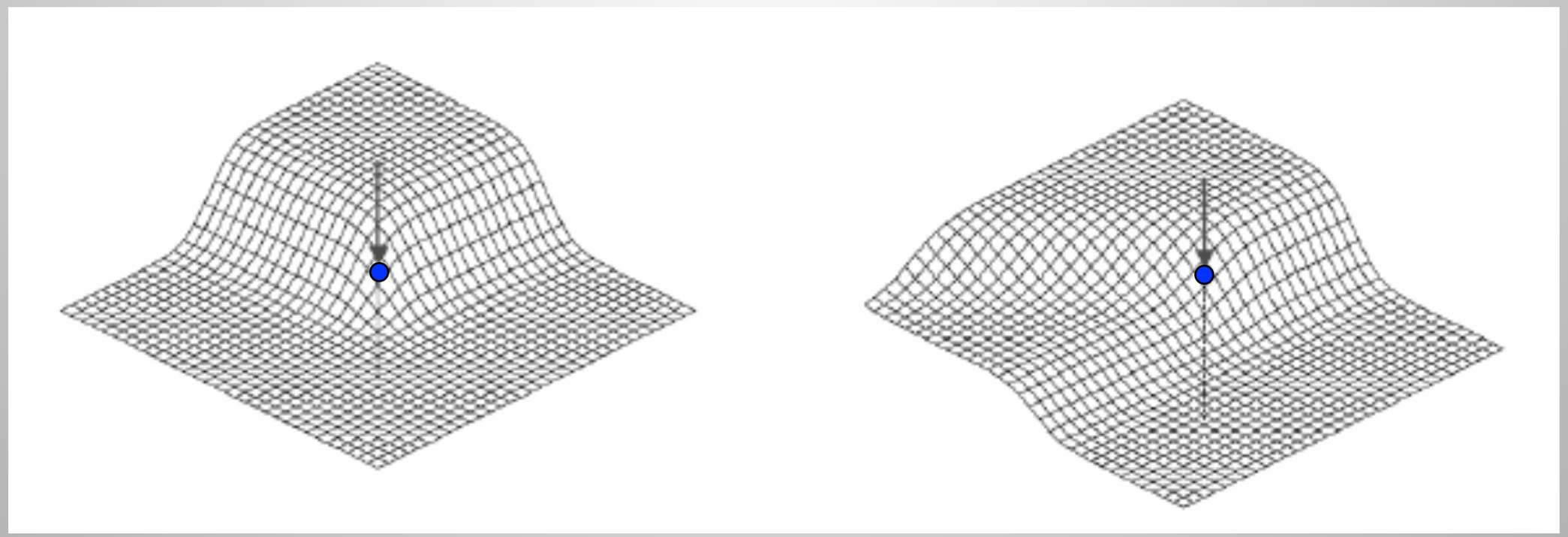


Solution

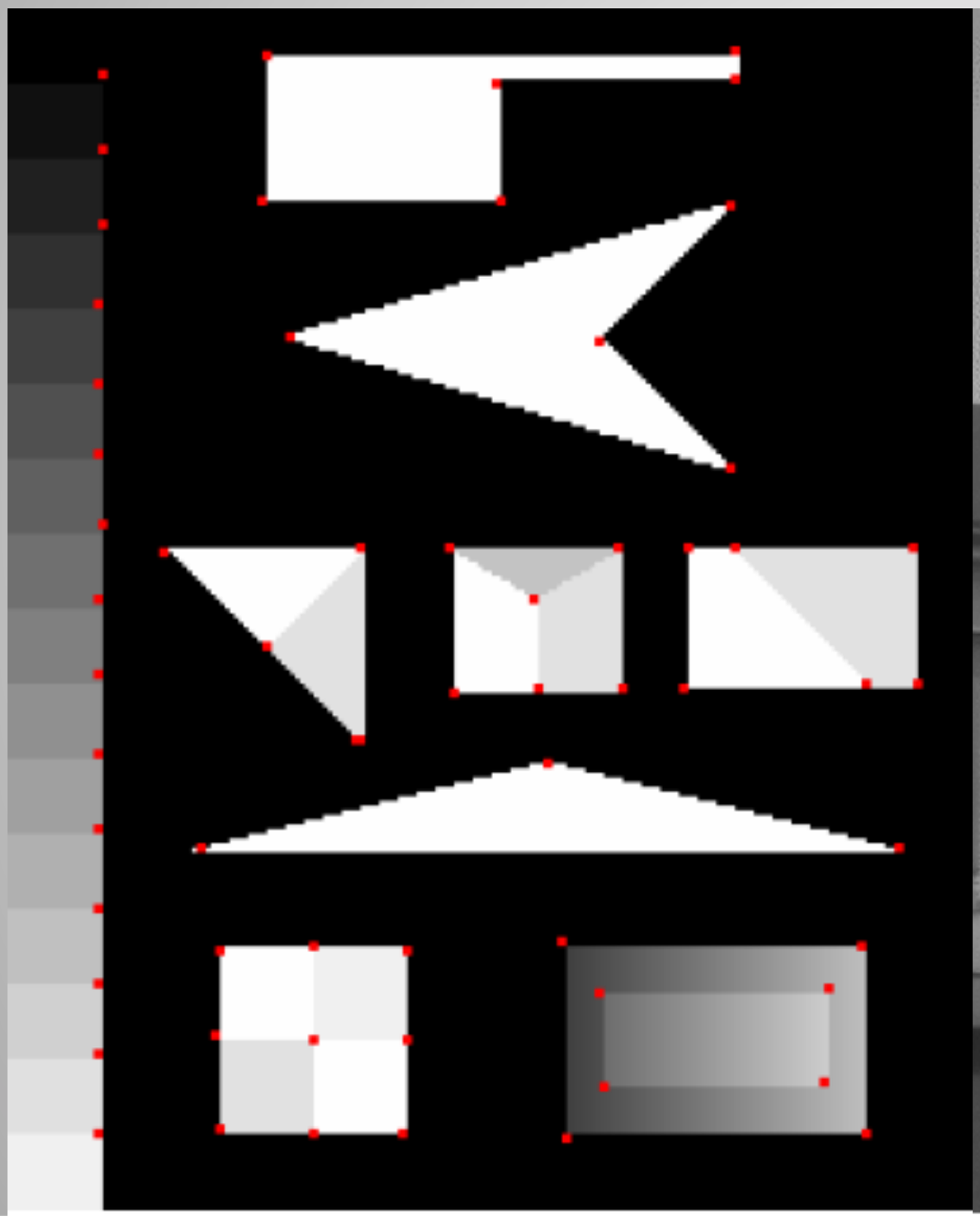
- Detect local features in both images
- Extract feature vectors
- Match features from two sources
- Estimate object location

What is an interest point?

- Expressive texture
  - The point at which the direction of the boundary of object changes abruptly
  - Intersection point between two or more edge segments



What is an interest point?



Properties of Interest Points

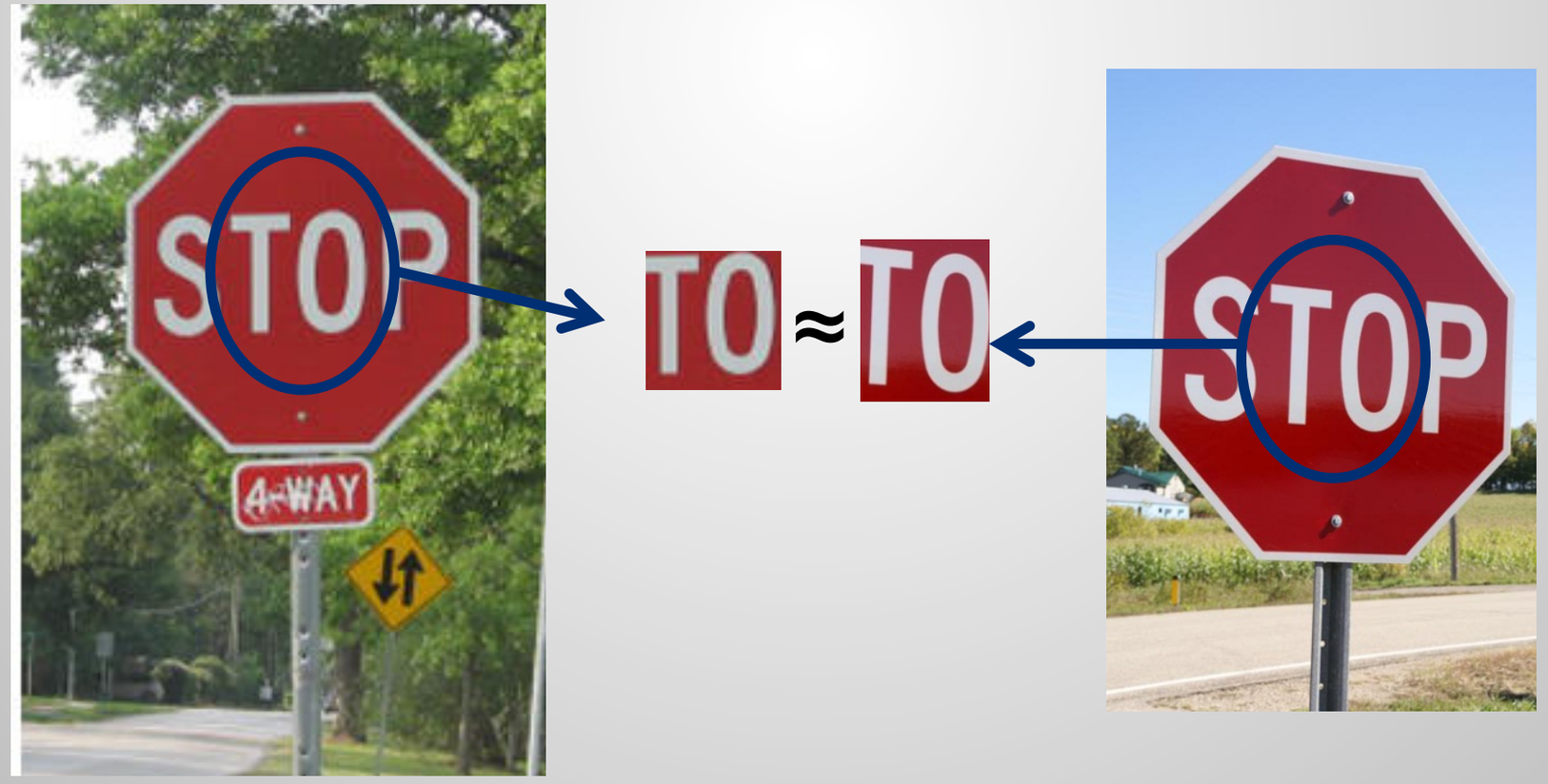
- Detect all (or most) true interest points
- No false interest points
- Well localized
- Robust with respect to noise
- Efficient detection

Possible Approaches for Corner Detection

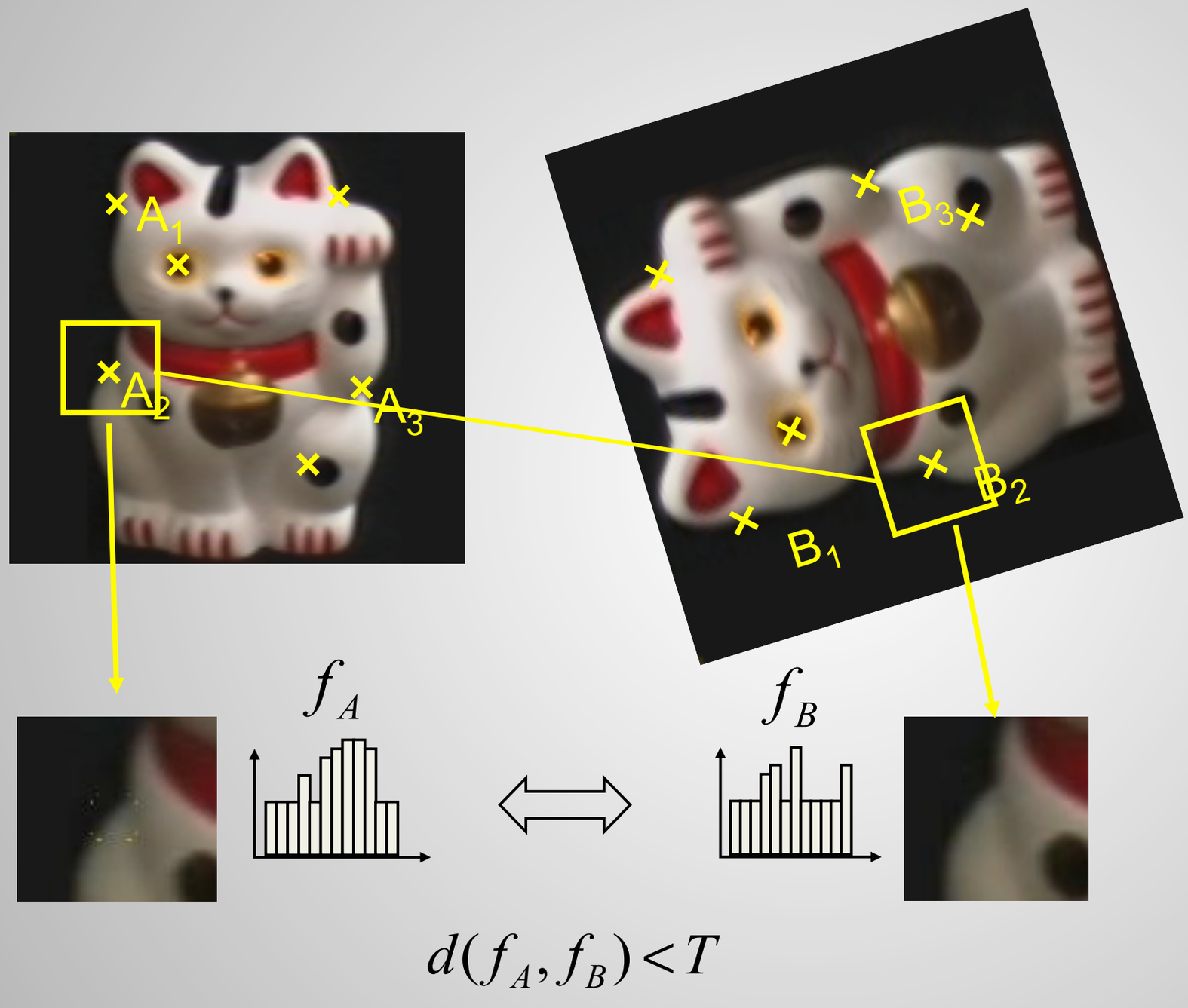
- Based on brightness of images
  - Usually image derivatives

Correspondence Across Views

- Correspondence: matching points, patches, edges, or regions across images



Overview of Keypoint Matching

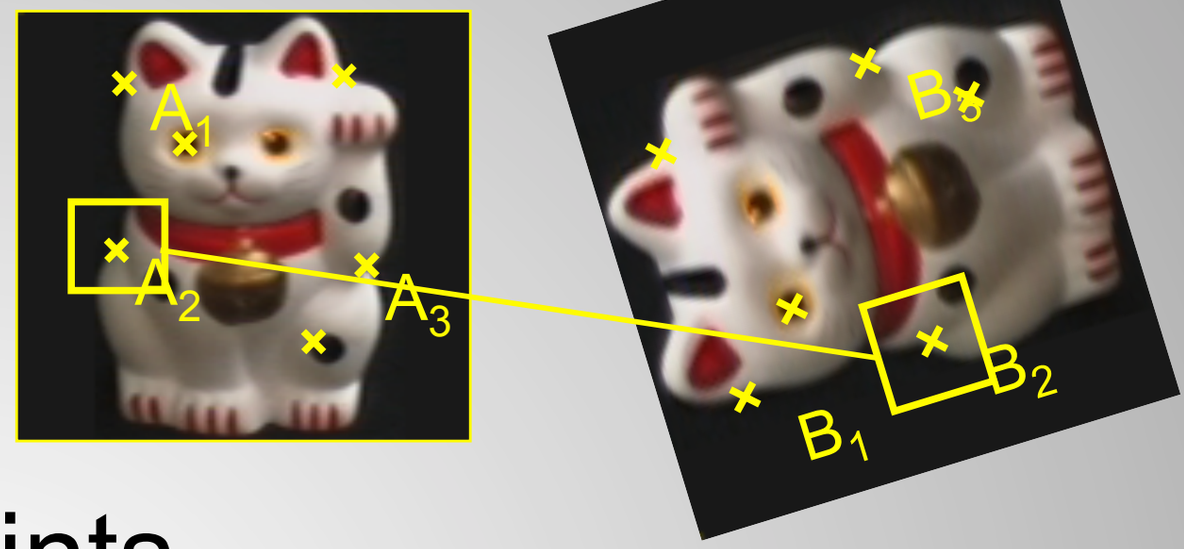


1. **Find a set of distinctive key-points**
2. **Define a region around each keypoint**
3. **Extract and normalize the region content**
4. **Compute a local descriptor from the normalized region**
5. **Match local descriptors**

Goals for Keypoints

Detect points that are *repeatable* and *distinctive*

Key Trade-offs



Detection of interest points

More Repeatable
- Robust detection
- Precise localization

More Interest Points
- Robust to occlusion
- Works with less texture

Description of patches

More Distinctive
- Minimize wrong matches

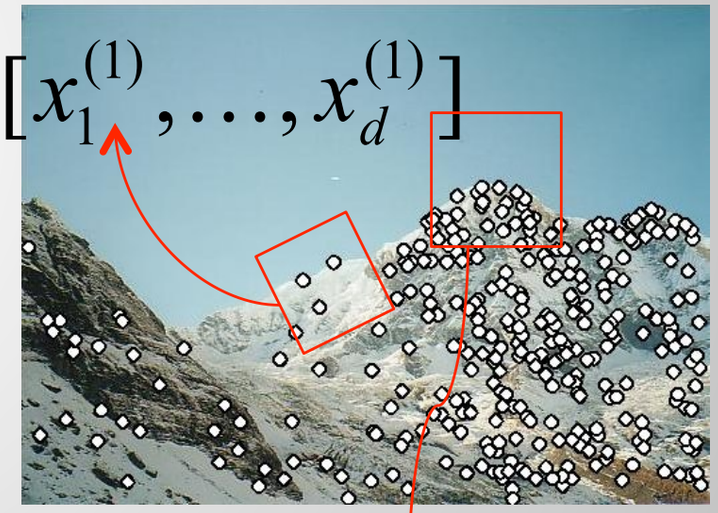
More Flexible

Local Features: Main Components

1) Detection: Identify the interest points

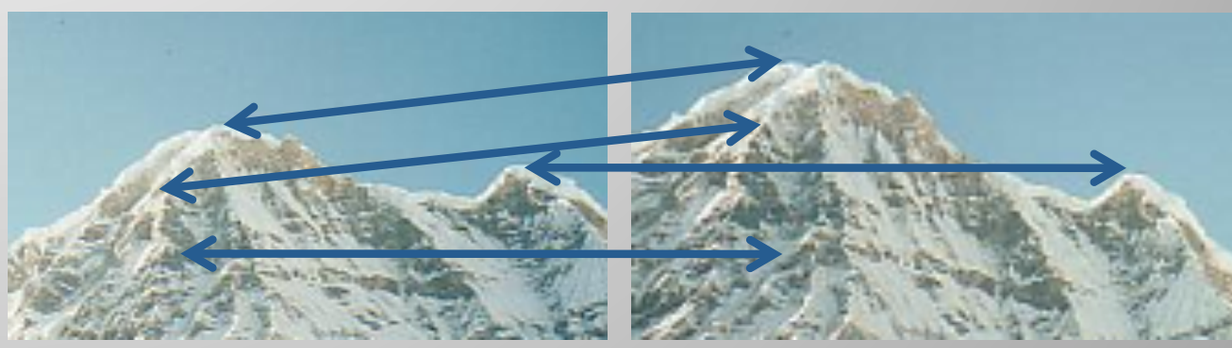
2) Description: Extract feature vector descriptor surrounding each interest point.

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



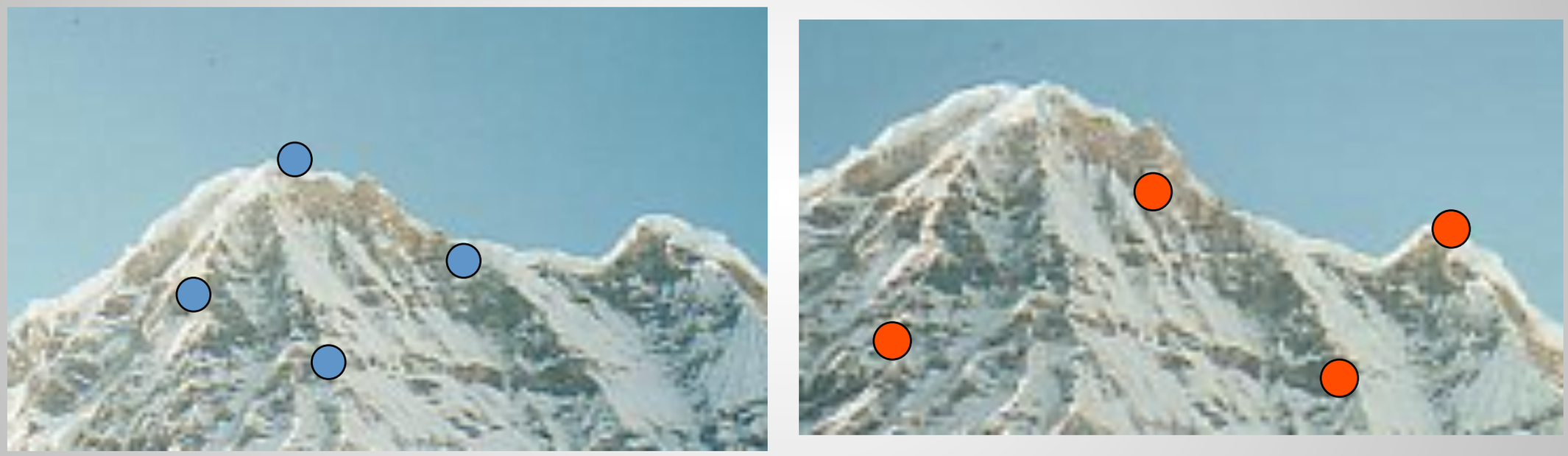
$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) Matching: Determine correspondence between descriptors in two views



Goal: interest operator repeatability

- We want to detect (at least some of) the same points in both images.

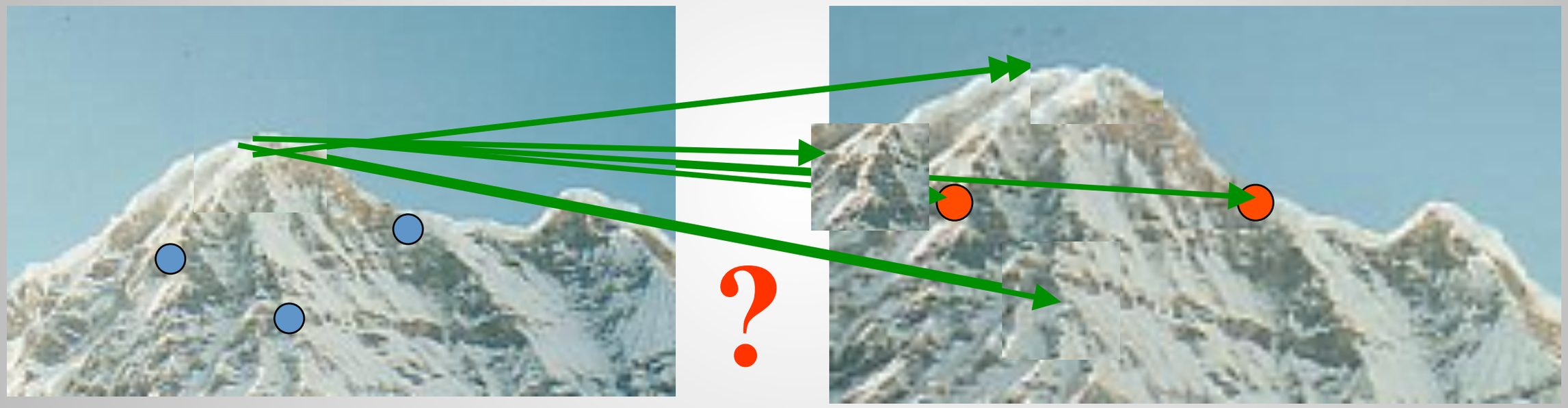


No chance to find true matches!

- Yet we have to be able to run the detection procedure *independently* per image.

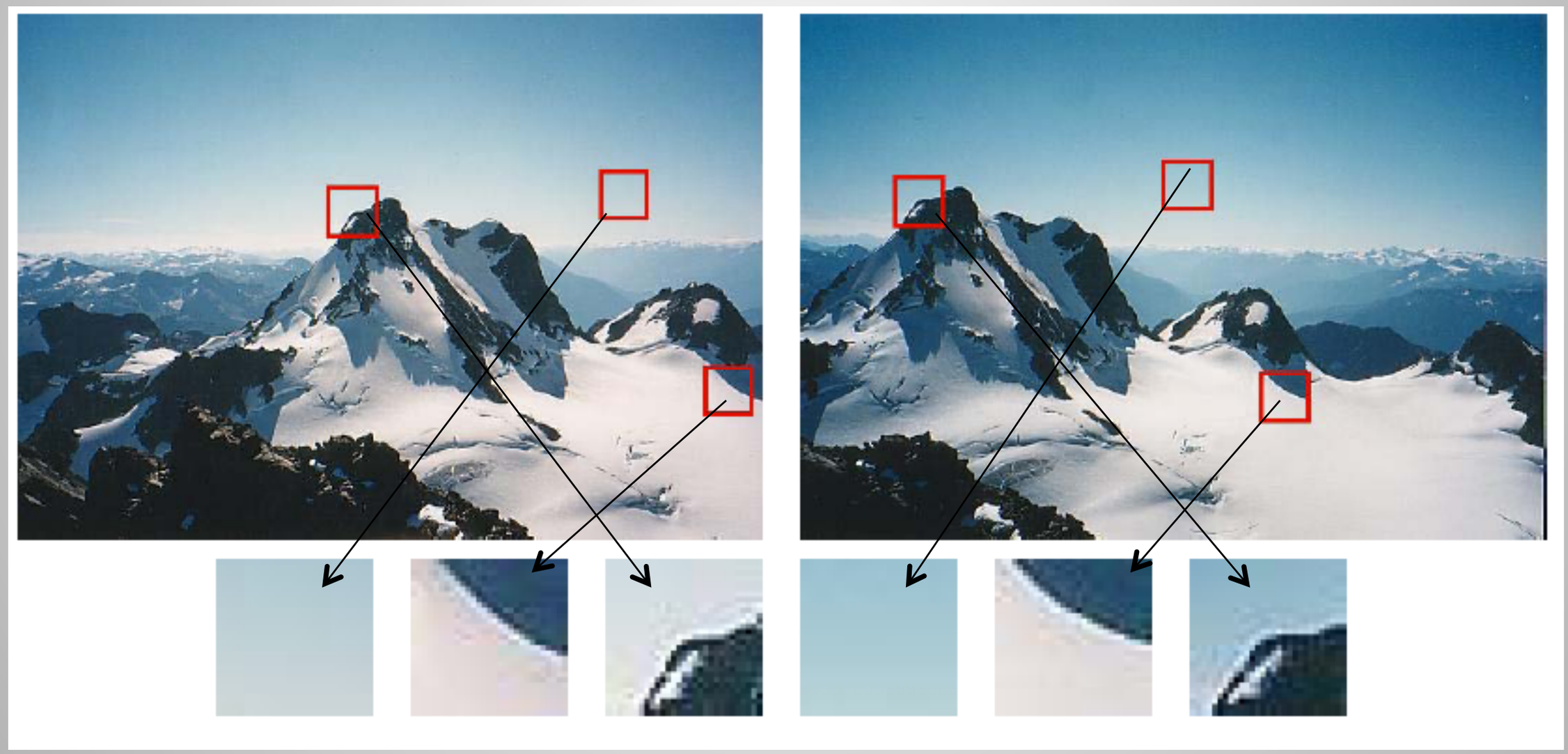
Goal: descriptor distinctiveness

- We want to be able to reliably determine which point goes with which.

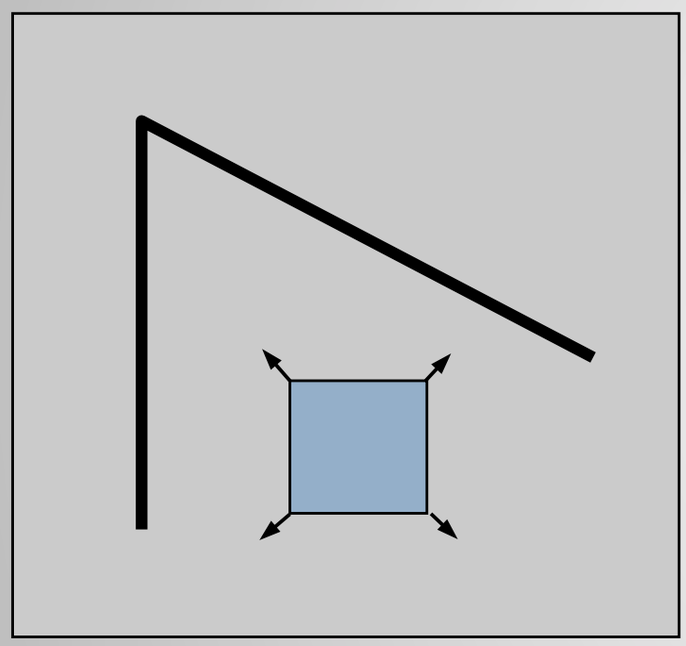


- Must provide some invariance to geometric and photometric differences between the two views.

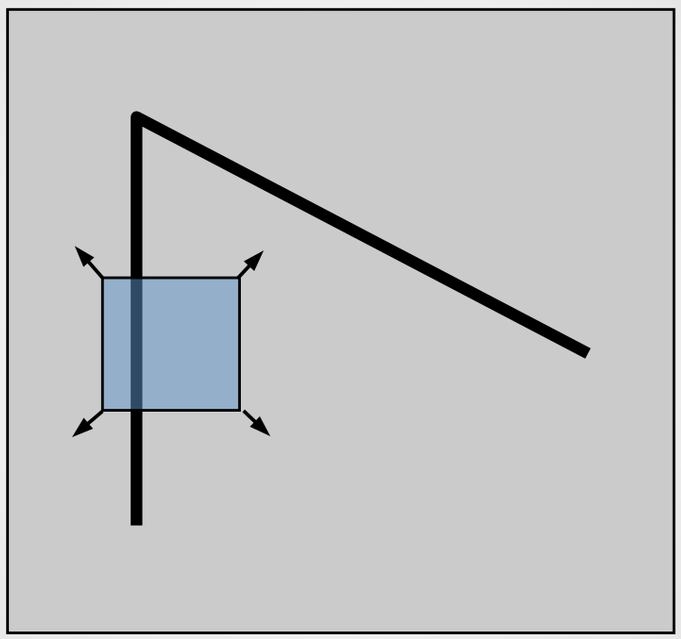
Some patches can be localized or matched with higher accuracy than others



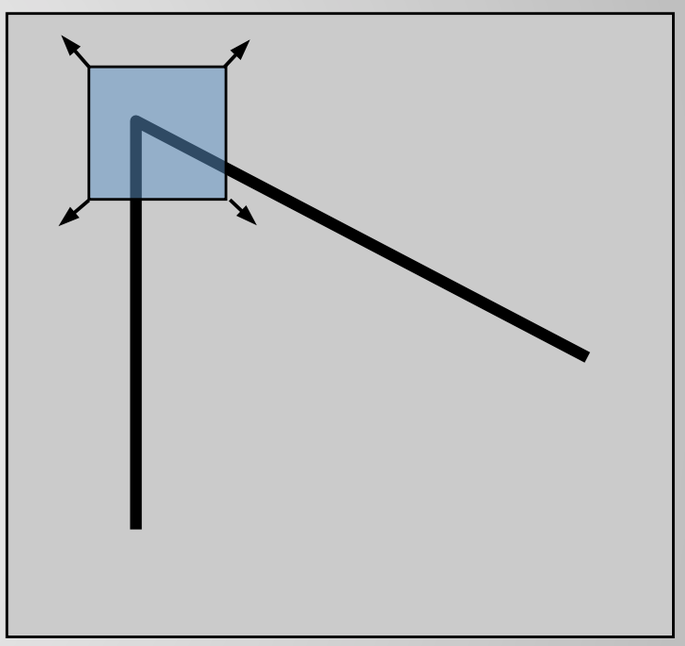
Basic Idea in Corner Detection



“flat” region:
no change in
all directions



“edge”:
no change along
the edge direction



“corner”:
significant change
in all directions

Corner Detection: Mathematics

Change in appearance for the shift [*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

Intensity

I, shifted I, subtract and square them, sum up over some window

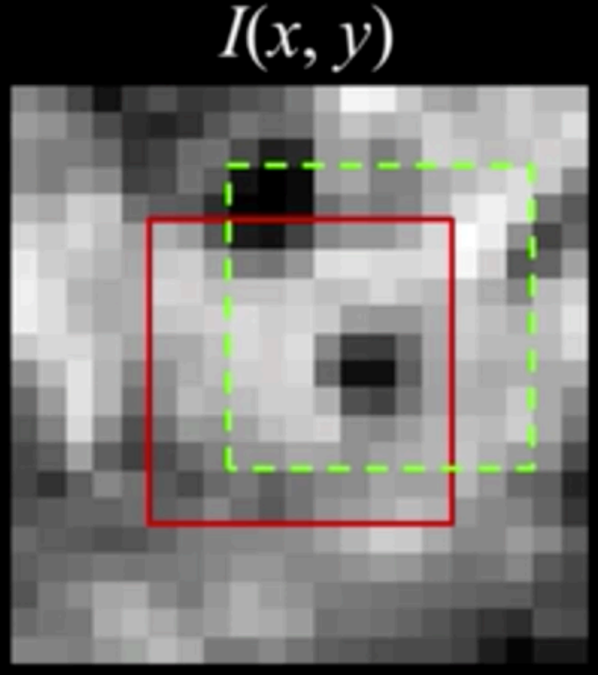
Source: R. Szeliski

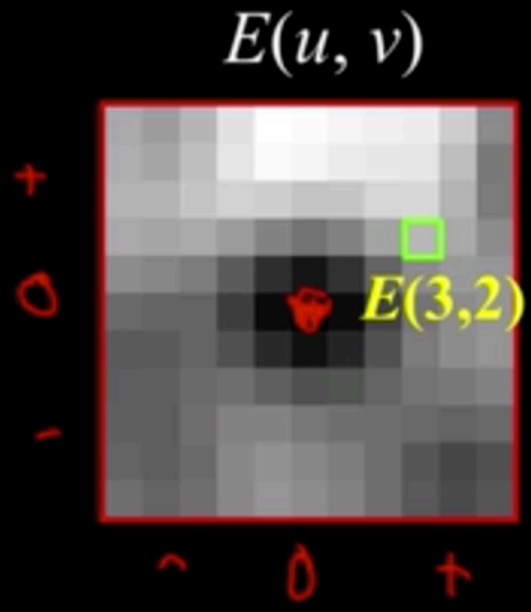
Based on an approximation
model and an error model

Corner Detection: Mathematics

Change in appearance for the shift [*u,v*]:

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





E(3,4) = gray > 0

What if the picture was completely gray?

Corner Detection: Mathematics

Change in appearance for the shift [*u,v*]:

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

Second-order Taylor expansion of $E(u,v)$ about (0,0) (local quadratic approximation for small *u,v*):

Notation:

C(x,y), Sum(s,t)

E(u,v), Sum(x,y)

Taylor:

f(s+x,t+y) =~ f(s,t) + xf_x(s,t) + yf_y(s,t)

I(x+u, y+v) =~ I(x,y) + uI_u(x,y) + vI_v(x,y)

Goal: Easy to compute whether the neighborhood around a point feels corner like

Corner Detection: Mathematics

The quadratic approximation simplifies to

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

where *M* is a *second moment matrix* computed from image derivatives:

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

Corner Detection using the Hessian Matrix

- Corners are characterized by high-curvature of intensity values.

$$H(p) = \begin{bmatrix} I_{xx}(p) & I_{xy}(p) \\ I_{xy}(p) & I_{yy}(p) \end{bmatrix}$$

Hessian Matrix
at voxel p.

Corner Detection using the Hessian Matrix

- Corners are characterized by high-curvature of intensity values.

$$H(p) = \begin{bmatrix} I_{xx}(p) & I_{xy}(p) \\ I_{xy}(p) & I_{yy}(p) \end{bmatrix}$$

Hessian matrix summarizes Distribution of gradients.

Hessian Matrix at voxel p. →

Eigenvalues of the Hessian indicate corner features if both eigenvalues are large!

One large, one small eigenvalues -> edge regions

Two small eigenvalues indicate -> flat regions

Reminder: Eigenvalues/Eigenvectors

$$H(p) = \begin{bmatrix} I_{xx}(p) & I_{xy}(p) \\ I_{xy}(p) & I_{yy}(p) \end{bmatrix}$$

$$det(H) = I_{xx}I_{yy} - I_{xy}^2$$

$$Tr(H) = I_{xx} + I_{yy}$$

Reminder: Eigenvalues/Eigenvectors

$$H(p) = \begin{bmatrix} I_{xx}(p) & I_{xy}(p) \\ I_{xy}(p) & I_{yy}(p) \end{bmatrix}$$

$$det(H) = I_{xx}I_{yy} - I_{xy}^2$$

$$Tr(H) = I_{xx} + I_{yy}$$

The eigenvalues of the matrix H are solutions of its characteristics polynomial

$$det(\mathbf{H} - \lambda \mathbf{1}_2) = 0$$

$$det(\begin{bmatrix} I_{xx}(p) - \lambda & I_{xy}(p) \\ I_{xy}(p) & I_{yy}(p) - \lambda \end{bmatrix}) = 0$$

$$(I_{xx}(p) - \lambda)(I_{yy}(p) - \lambda) - I_{xy}(p)^2 = 0$$

Harris and Stephens Corner Detector

- Instead of using *Hessian* of image I, use first derivative of smoothed I (i.e., L)

$$G(p,\sigma) = \begin{bmatrix} L_x^2(p,\sigma) & L_x(p,\sigma)L_y(p,\sigma) \\ L_x(p,\sigma)L_y(p,\sigma) & L_y^2(p,\sigma) \end{bmatrix}$$

Harris and Stephens Corner Detector

- Instead of using *Hessian* of image I, use first derivative of smoothed I (i.e., L)

$$G(p,\sigma) = \begin{bmatrix} L_x^2(p,\sigma) & L_x(p,\sigma)L_y(p,\sigma) \\ L_x(p,\sigma)L_y(p,\sigma) & L_y^2(p,\sigma) \end{bmatrix}$$

- Instead of calculating eigenvalues, we will consider ***cornerness*** feature:

$$\mathcal{H}(p,\sigma,\alpha) = det(G) - \alpha.Tr(G)$$

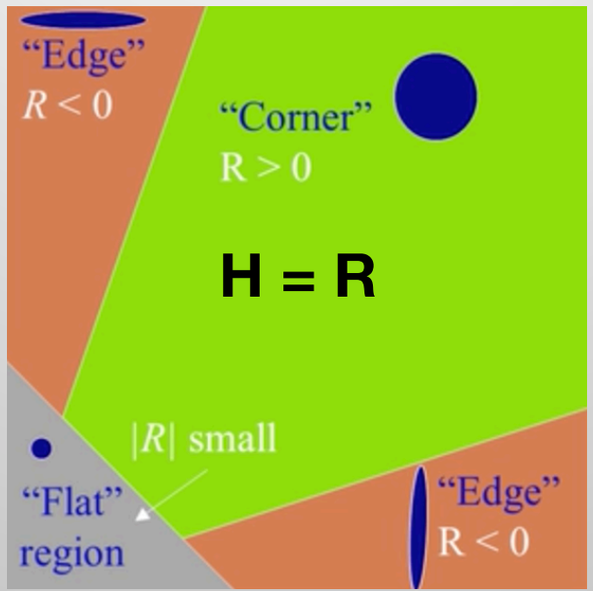
for small alpha (=1/25)

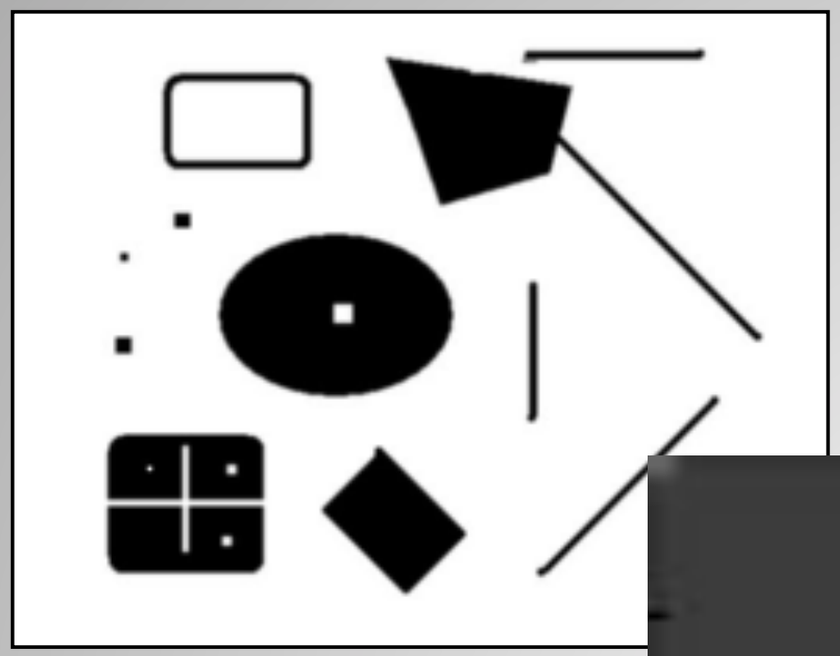
Harris and Stephens Corner Detector

$$\mathcal{H}(p, \sigma, \alpha) = \lambda_1 \lambda_2 - \alpha.(\lambda_1 + \lambda_2)$$

- The same behavior as Hessian based detector, but now we directly use simple determinant and trace functions!

$$\mathcal{H}(p, \sigma, \alpha) = det(G) - \alpha.Tr(G)$$





I_x

I_y

Square of derivatives

Gaussian Smoothing

Feature Extraction: Corners

Summary

Corner Detection: Basic Idea

"flat" region: no change in all directions

"edge": no change along the edge direction

"corner": significant change in all directions with small shift

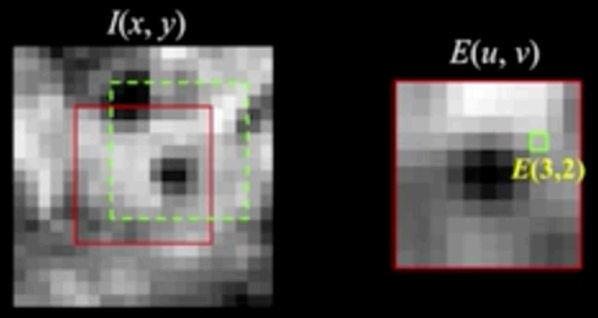
Source: A. Efros

Move region - see change

Corner Detection: Mathematics

Change in appearance for the shift [u,v]:

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



Change / error equation, viz

Corner Detection: Mathematics

The quadratic approximation simplifies to

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

where *M* is a *second moment matrix* computed from image derivatives:

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

Approximation, 2nd moment matrix

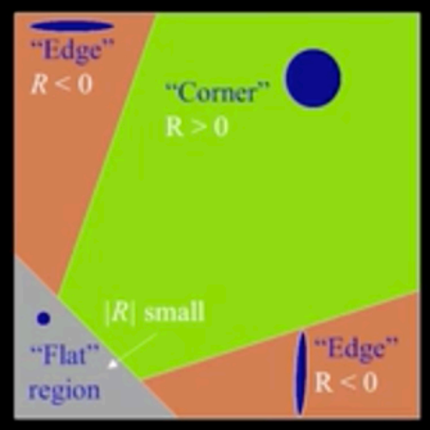
Harris corner response function

$$R = \det(M) - \alpha\, \mathrm{trace}(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

R is large for a corner

R is negative with large magnitude for an edge

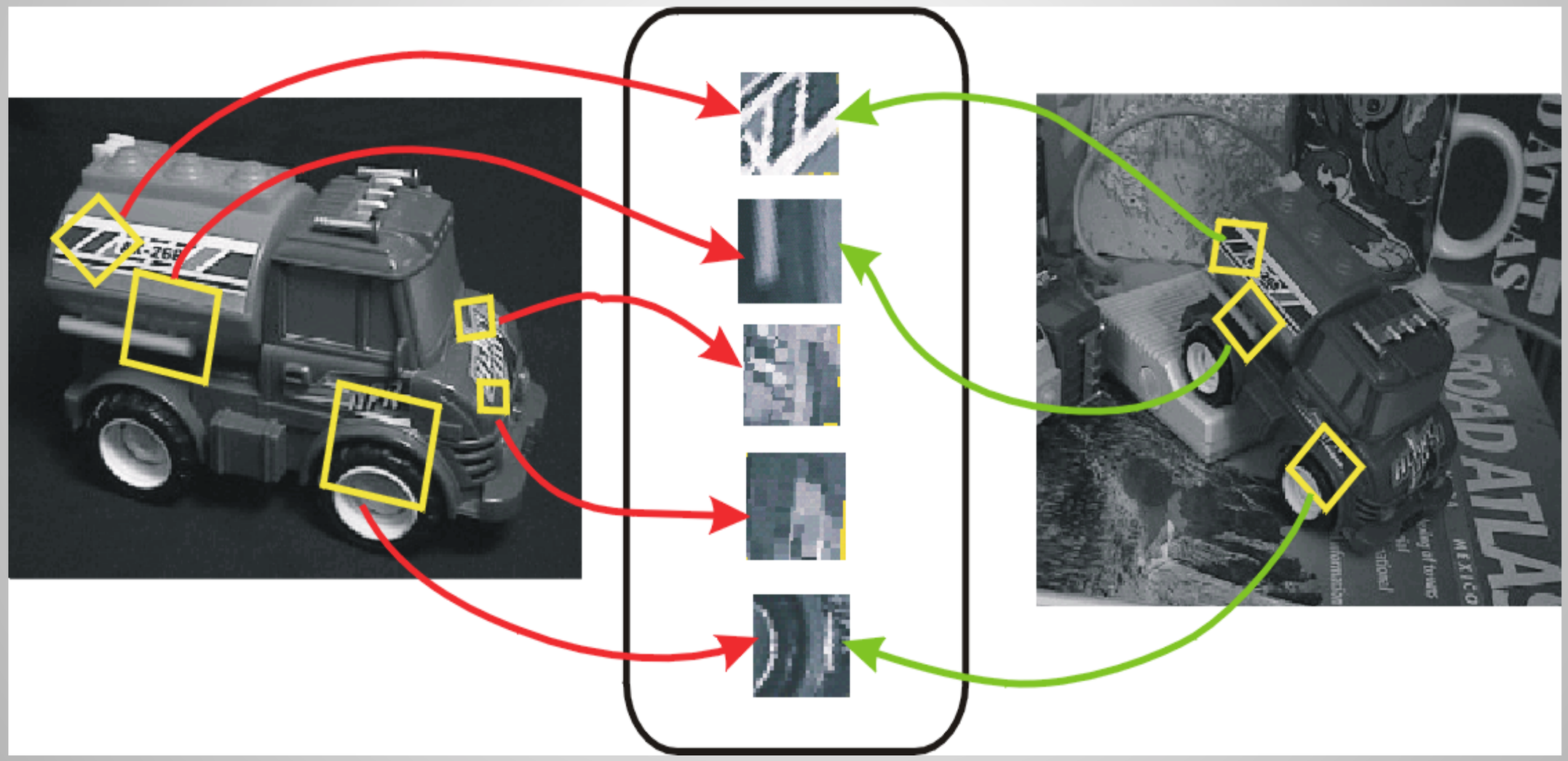
|*R*| is small for a flat region



R operator, det() & trace()

Invariant Local Features

•Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



Features Descriptors

Invariance and Covariance

- We want corner locations to be ***invariant*** to photometric transformations and ***covariant*** to geometric transformations
  - **Invariance:** image is transformed and corner locations do not change
  - **Covariance:** if we have two transformed versions of the same image, features should be detected in corresponding locations

Outline

- Concept of Scale
  - Pyramids
  - Scale-space approaches briefly
- Scale invariant region selection
- SIFT: an image region descriptor

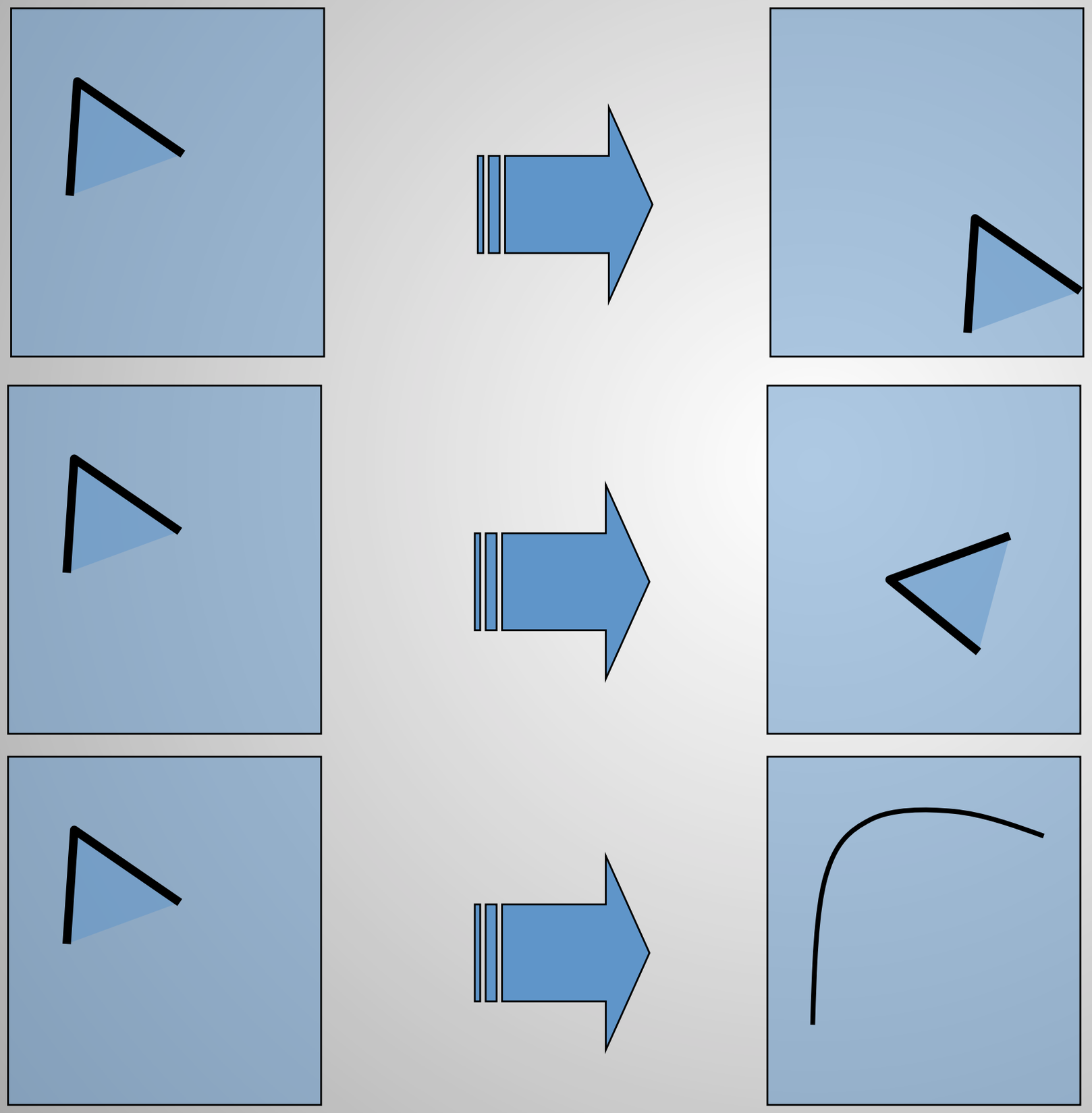
- *David Lowe, IJCV 2004 paper [Please read it!]*

Reminder: Motivation

- Image Matching
  - Fundamental aspect of many problems
    - Object Recognition
    - 3D Structures
    - Stereo Correspondence
    - Motion Tracking
    - ….

What are the desired features to conduct these tasks?

Review of the Corner Detection



Corners are invariant to

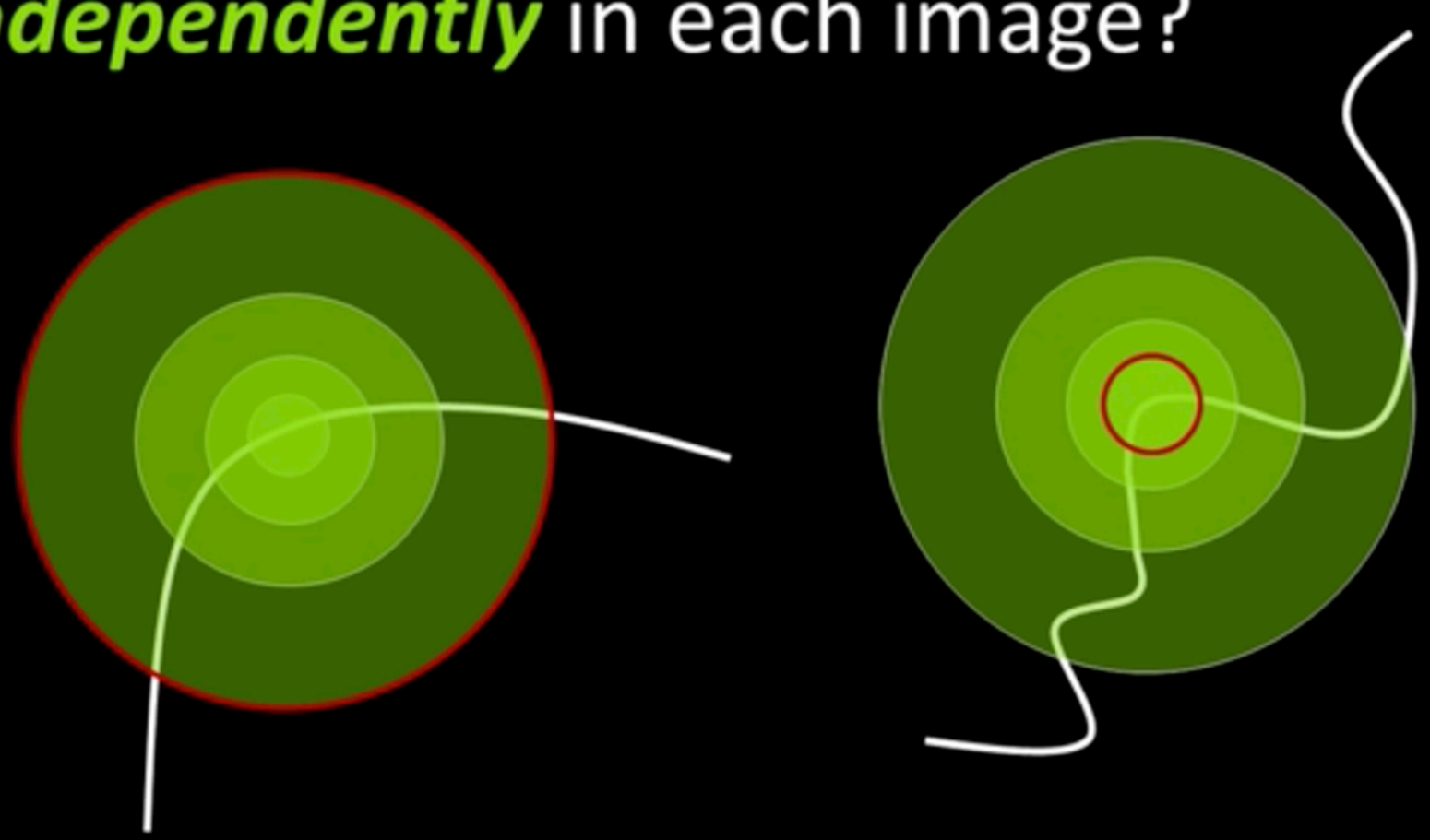
translation ✔

rotation ✔

scaling ✖

Scale Invariant Detection

- The problem: how do we choose corresponding circles ***independently*** in each image?



Multi-scale Representation

Image

Small Scale Features → Smaller Filter Masks

Large Scale Features → Larger Filter Masks

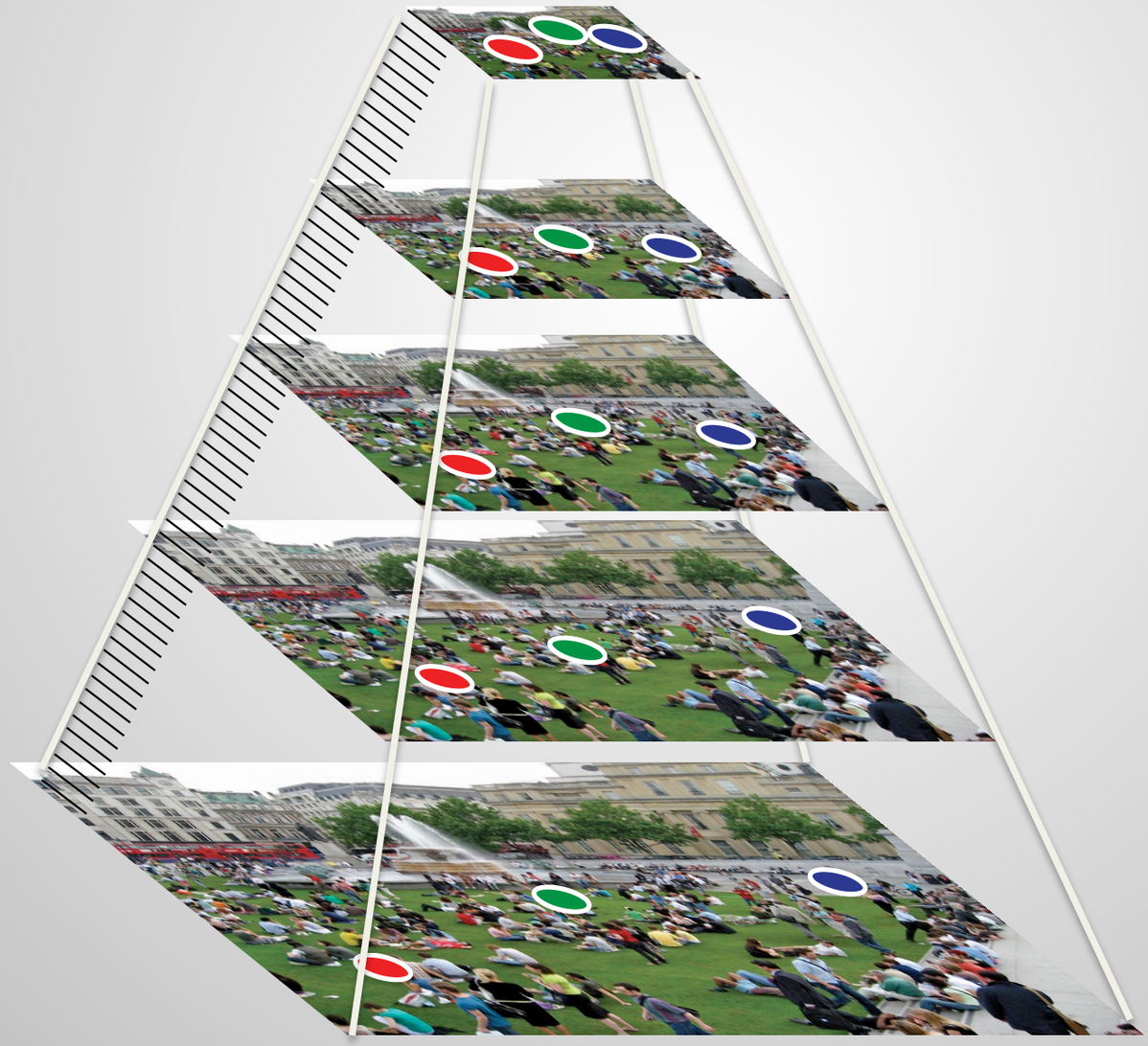
Computational Cost increases!

Doubling the scale leads to four-fold increase in the number of operations in 2D.

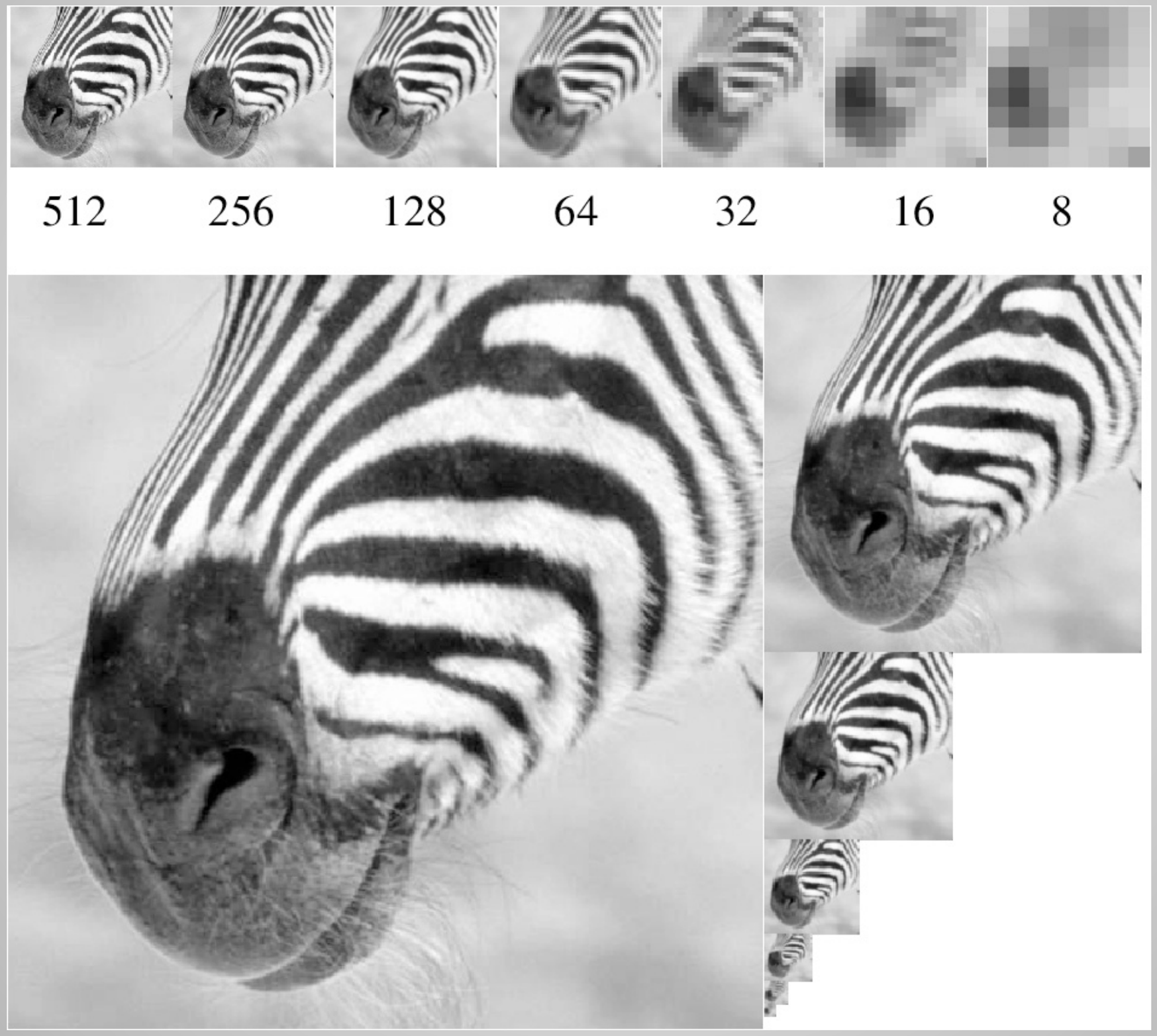
Multi-Scale Representation: Pyramid

- Pyramid is one way to represent images in multi-scale
  - Pyramid is built by using multiple copies of image.
  - Each level in the pyramid is 1/4 of the size of previous level.
  - The lowest level is of the highest resolution.
  - The highest level is of the lowest resolution.

Pyramid can capture global and local features

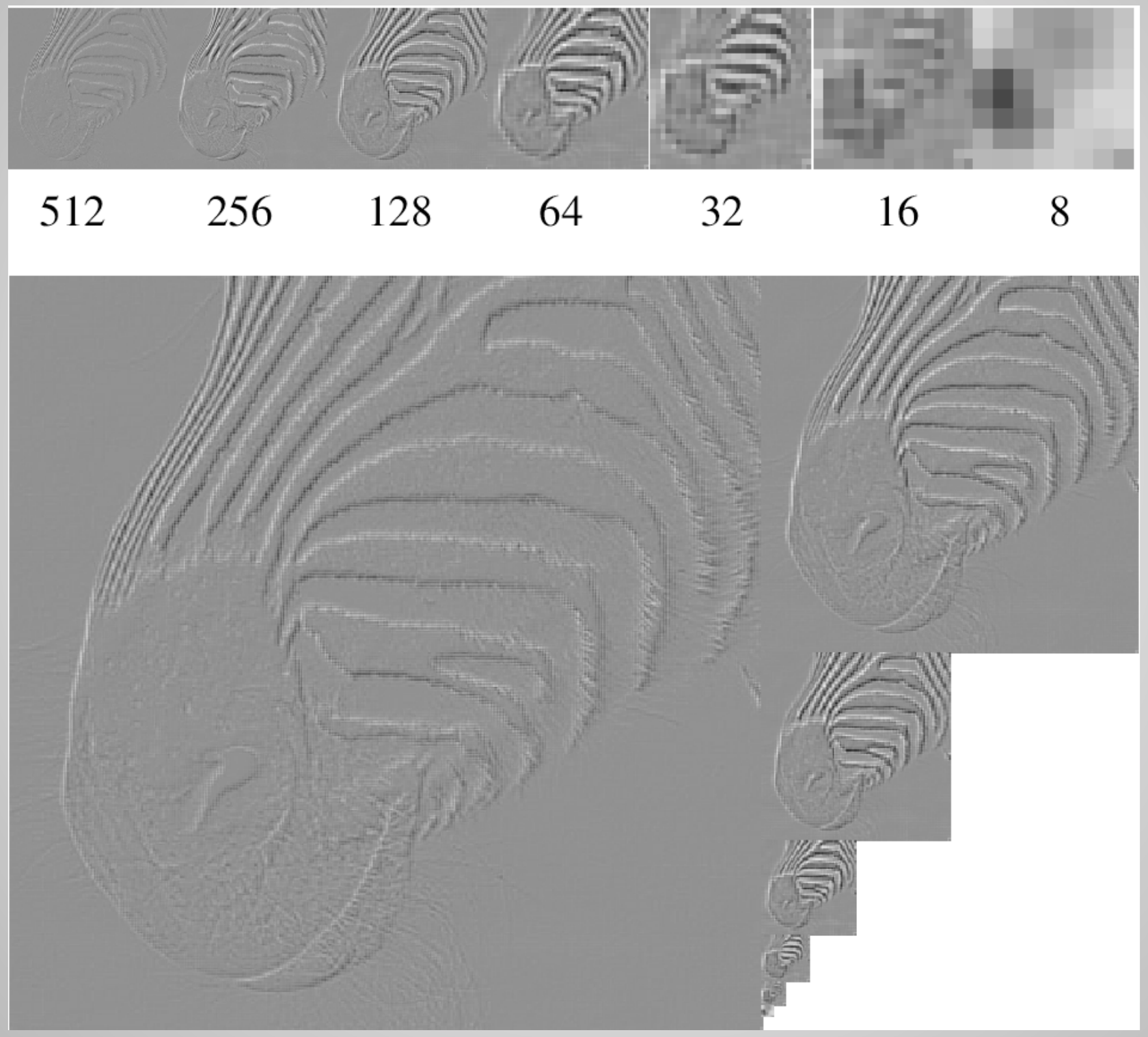


Gaussian Pyramids



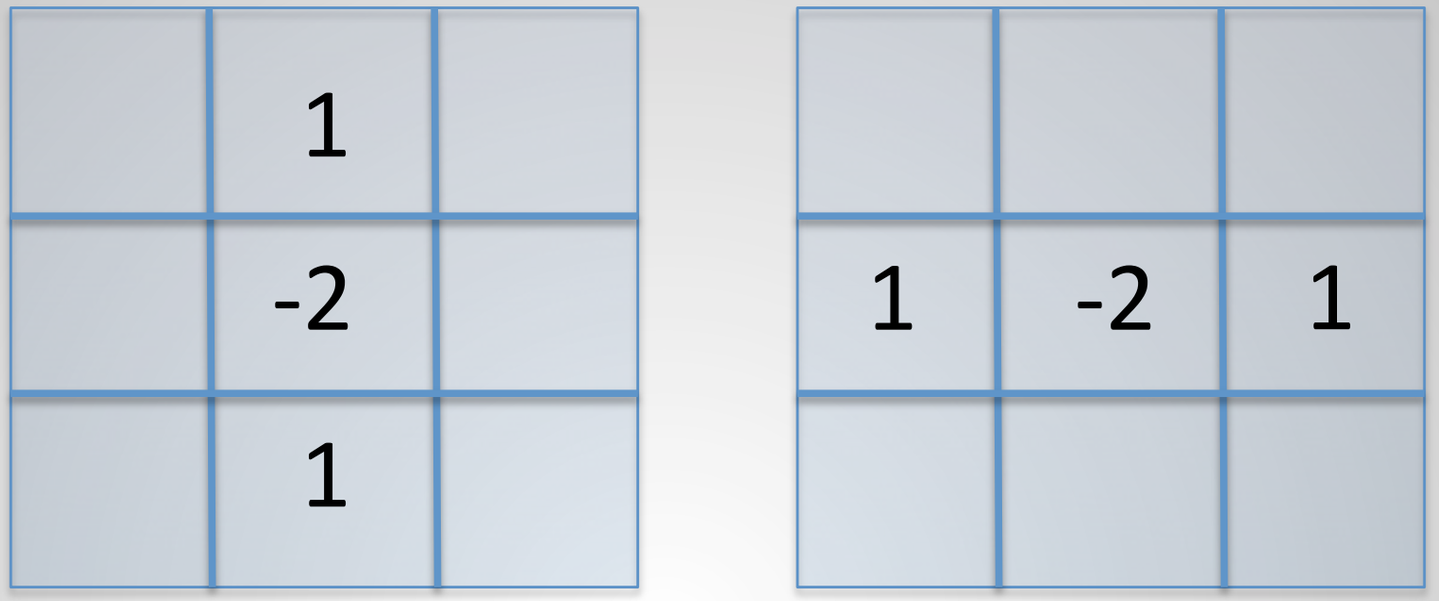
Source: Forsyth

Laplacian Pyramids



Source: Forsyth

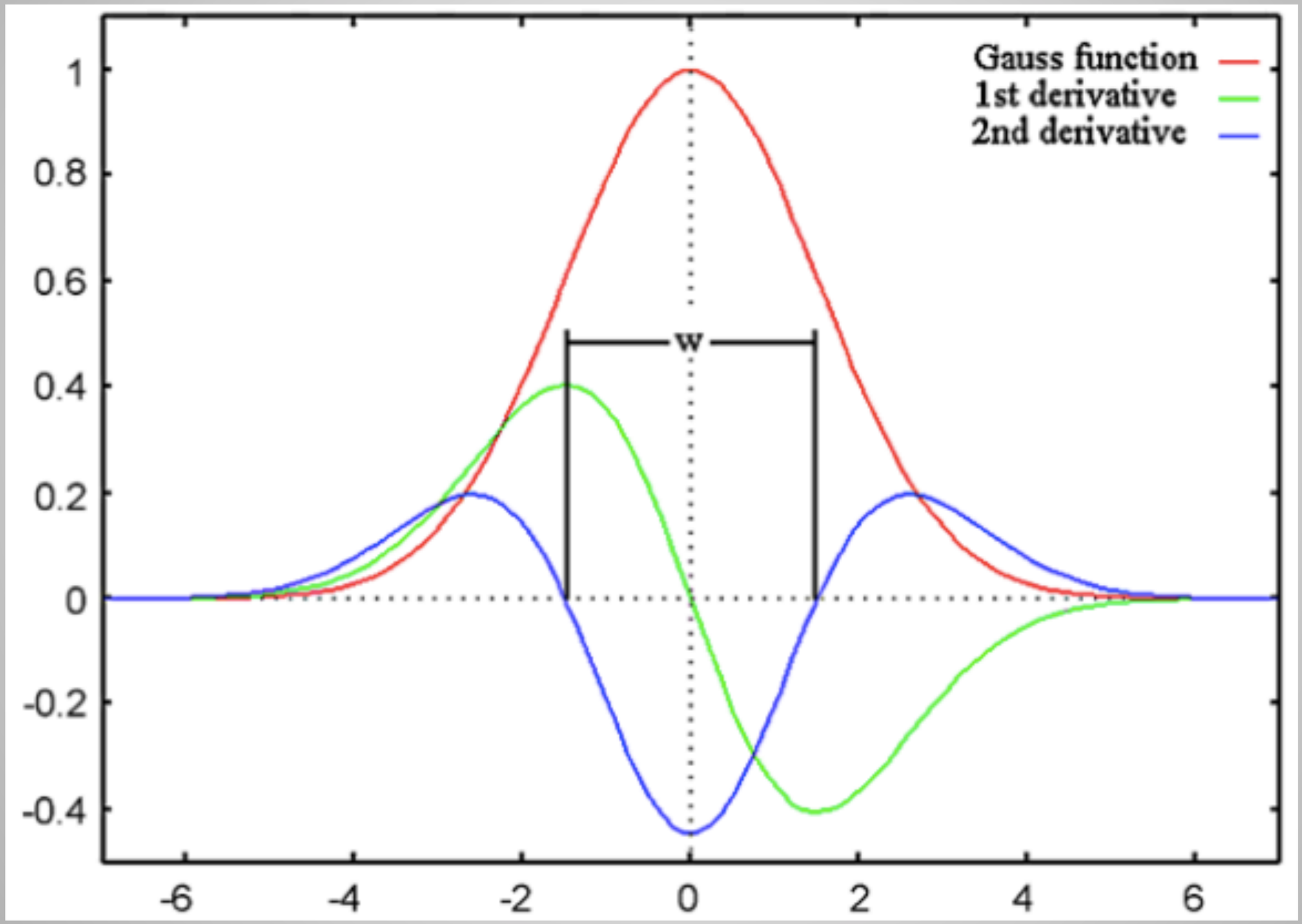
Laplacian Pyramids



Laplacian of Gaussian (LoG): Gaussian first to smooth images then perform Laplacian operation

$$\bigtriangledown^2(G_\sigma * I) = I * \bigtriangledown^2 G_\sigma$$

Laplacian Pyramids



Laplacian Pyramids

$$\frac{\delta G_\sigma}{\delta x}(x,y) = -\frac{x}{2\pi\sigma^4} e^{-(x^2+y^2)/(2\sigma^2)}$$

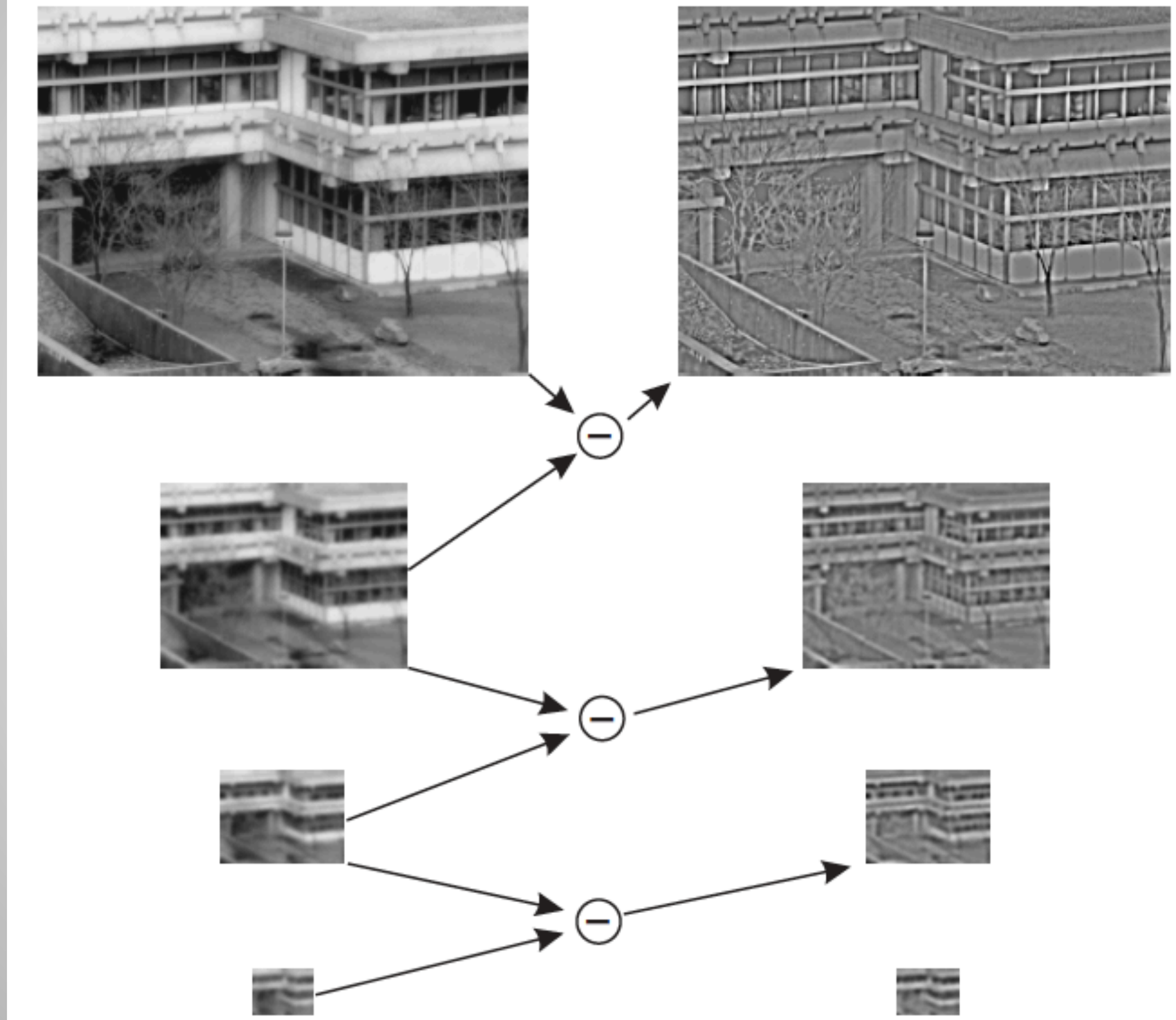
Laplacian Pyramids

$$\frac{\delta G_\sigma}{\delta x}(x,y) = -\frac{x}{2\pi\sigma^4} e^{-(x^2+y^2)/(2\sigma^2)}$$

Repeat same derivative for y component, and then take second derivatives.

$$\bigtriangledown^2 G_\sigma(x,y) = \frac{1}{2\pi\sigma^4} \frac{(x^2+y^2-2\sigma^2)}{\sigma^2} e^{-(x^2+y^2)/(2\sigma^2)}$$

Laplacian Pyramid Construction



Difference of Gaussian (DoG)

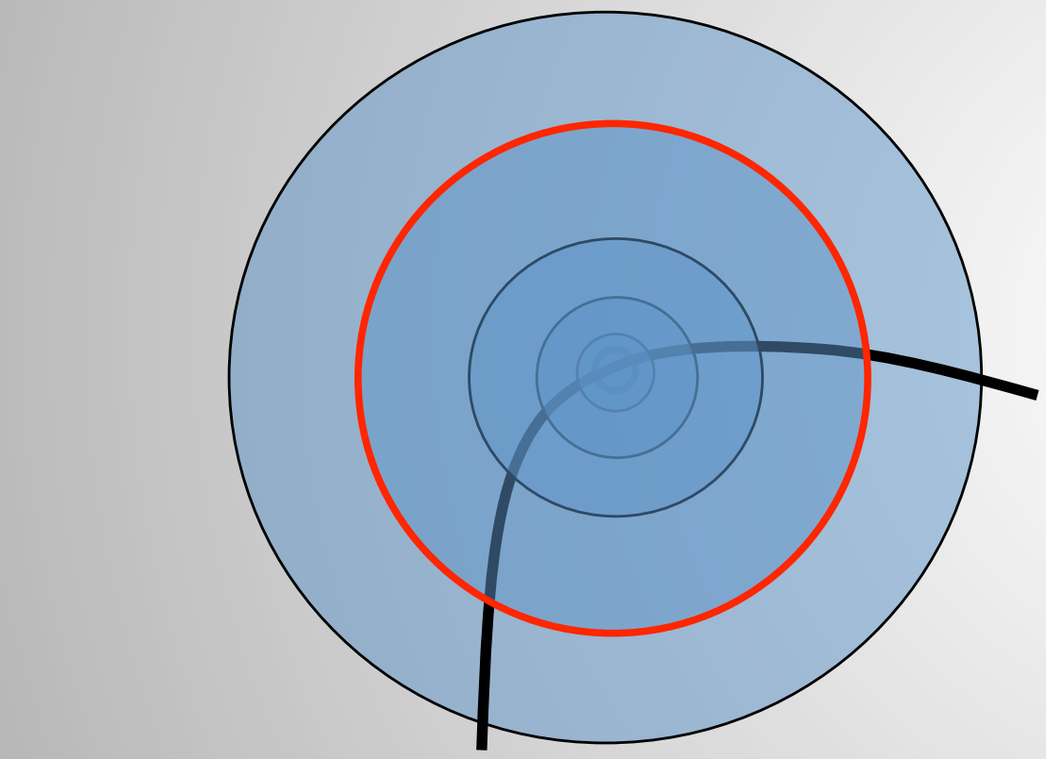
- It is a common approximation of LoG (better run time).

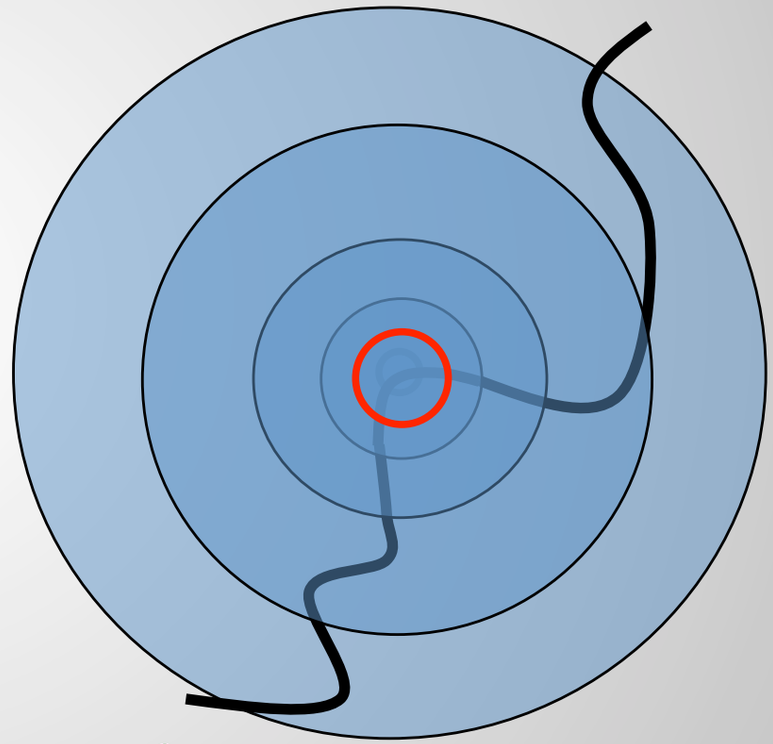
$$D_{\sigma,\alpha}(x,y) = L(x,y,\alpha) - L(x,y,\alpha\sigma)$$

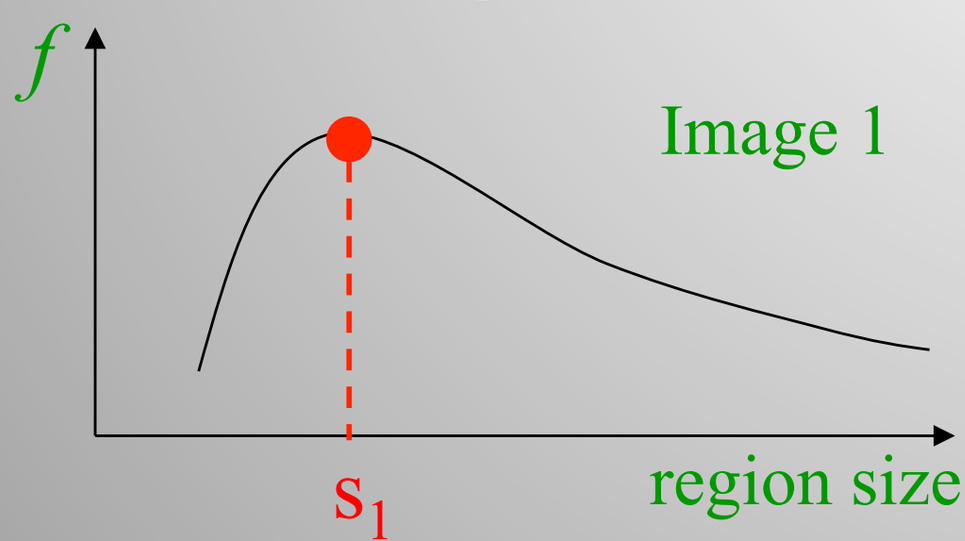
- It is the difference between a blurred copy of image I and an even more blurred copy of I.

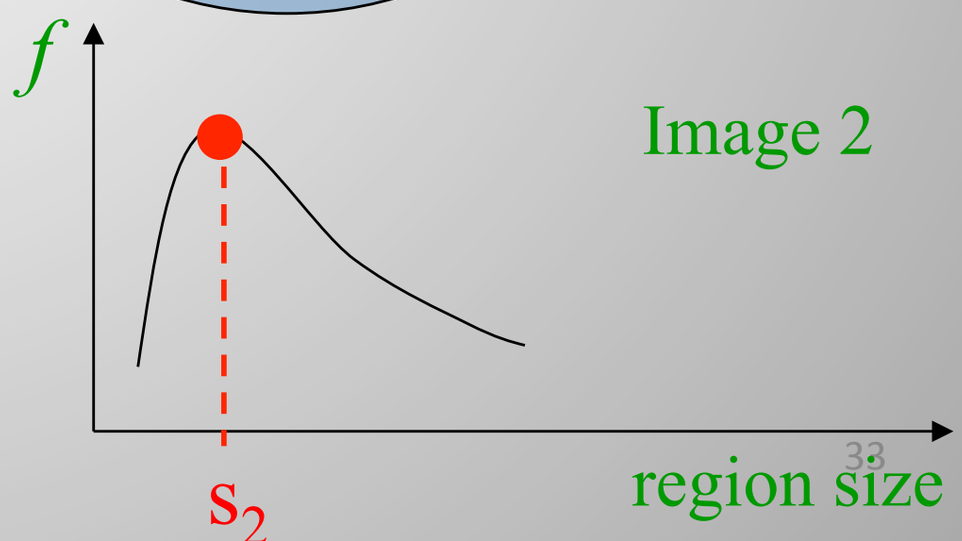
Scale Selection-Automated

- Find scale that gives local maxima of some function *f* in both position and scale.



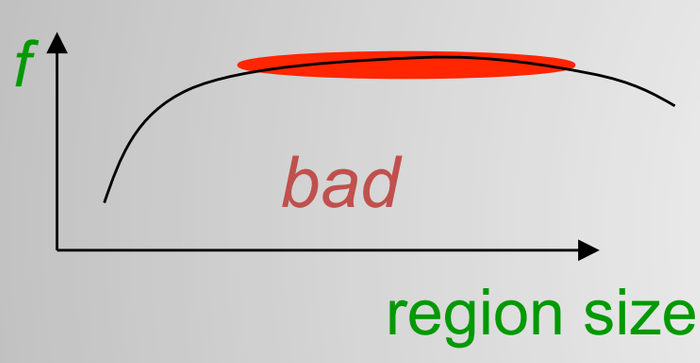


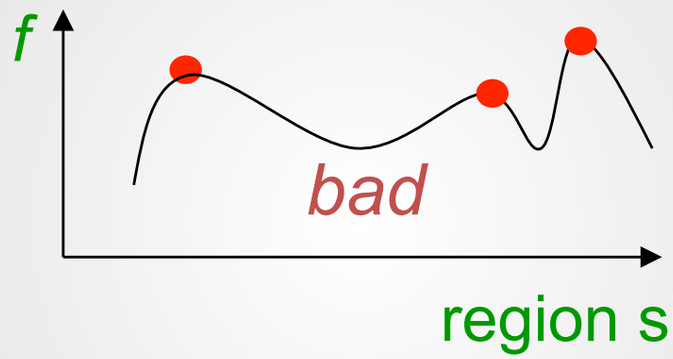


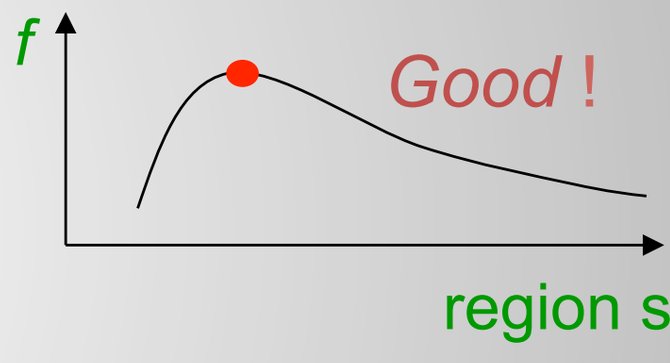


Good Function?

- A "good" function for scale detection: has one stable sharp peak







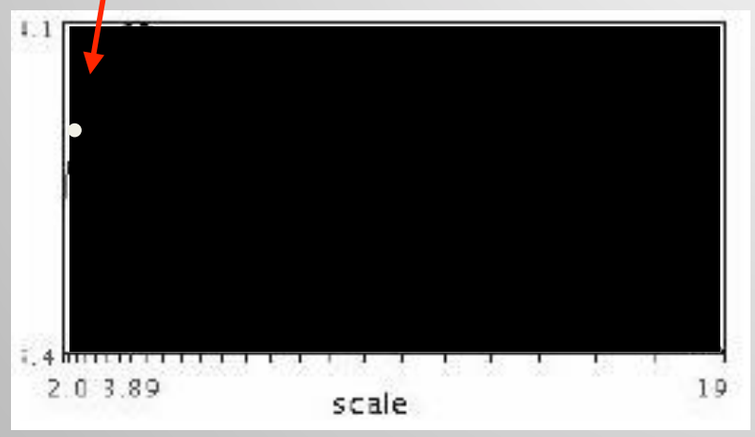
- For usual images: a good function would be a one which responds to contrast **(sharp local intensity change**)

Patch Size Corresponding to Scale

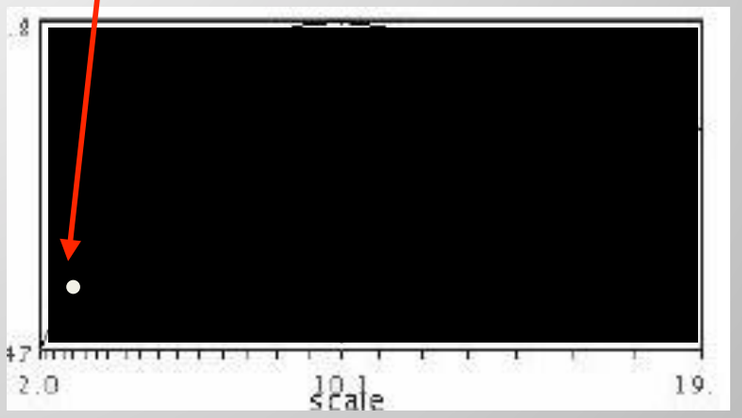
$$f(I_{i_1 \ldots i_m}(x,\sigma)) \;=\; f(I_{i_1 \ldots i_m}(x',\sigma'))$$

How to find corresponding patch sizes?

Patch Size Corresponding to Scale

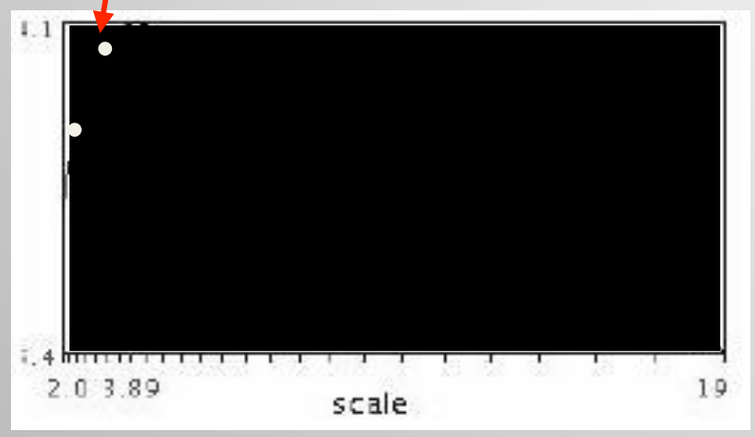


$f(I_{i_1 \ldots i_m}(x,\sigma))$

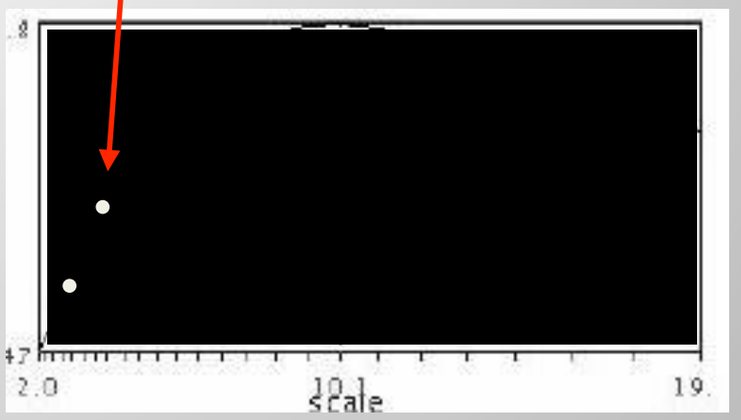


$f(I_{i_1 \ldots i_m}(x',\sigma))$

Patch Size Corresponding to Scale

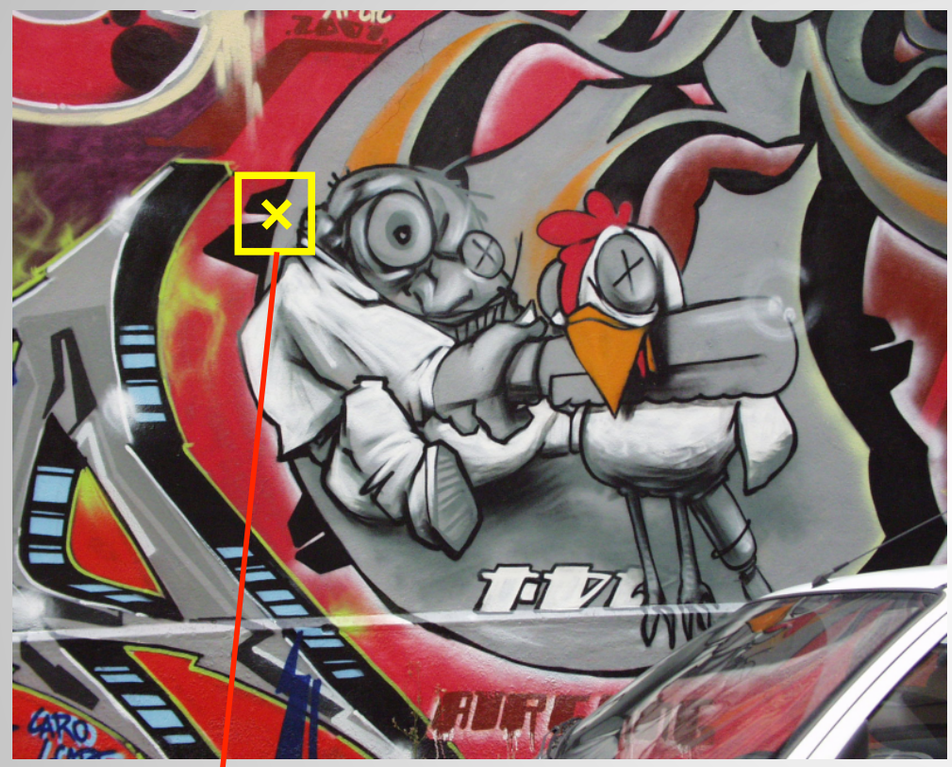


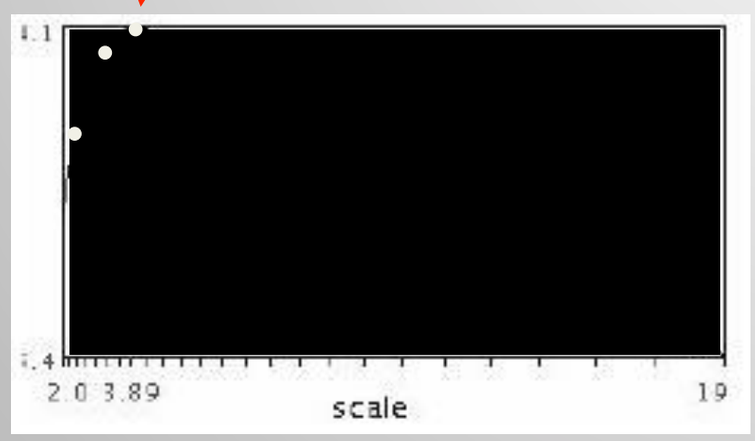
$f(I_{i_1 \ldots i_m}(x,\sigma))$



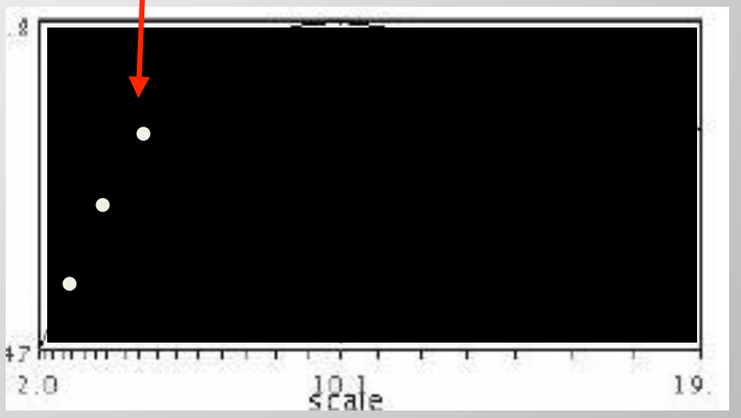
$f(I_{i_1 \ldots i_m}(x',\sigma))$

Patch Size Corresponding to Scale



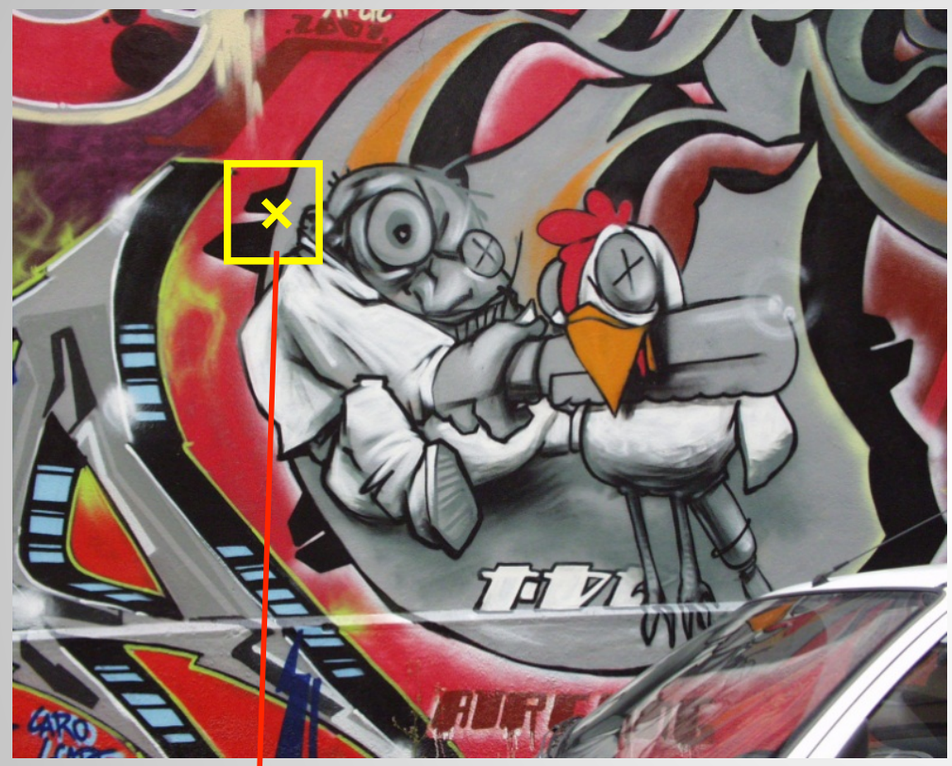


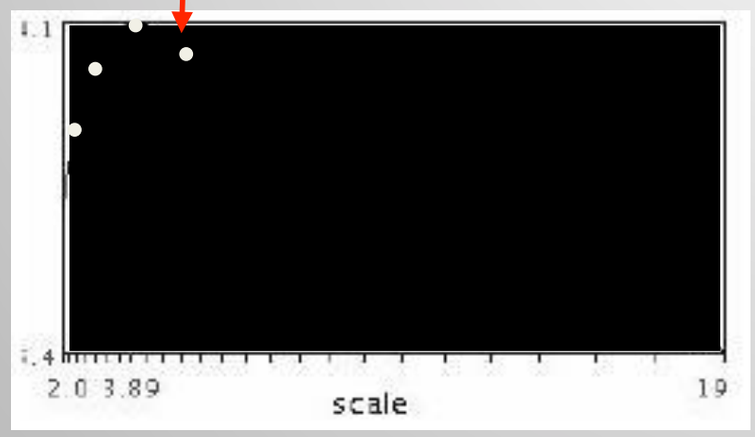
$f(I_{i_1 \ldots i_m}(x, \sigma))$



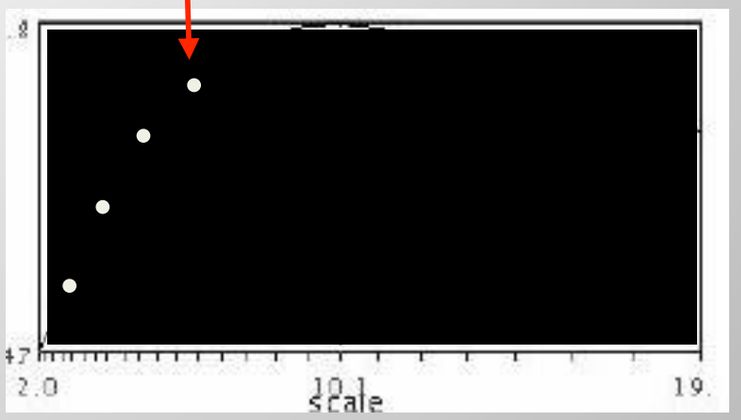
$f(I_{i_1 \ldots i_m}(x', \sigma))$

Patch Size Corresponding to Scale



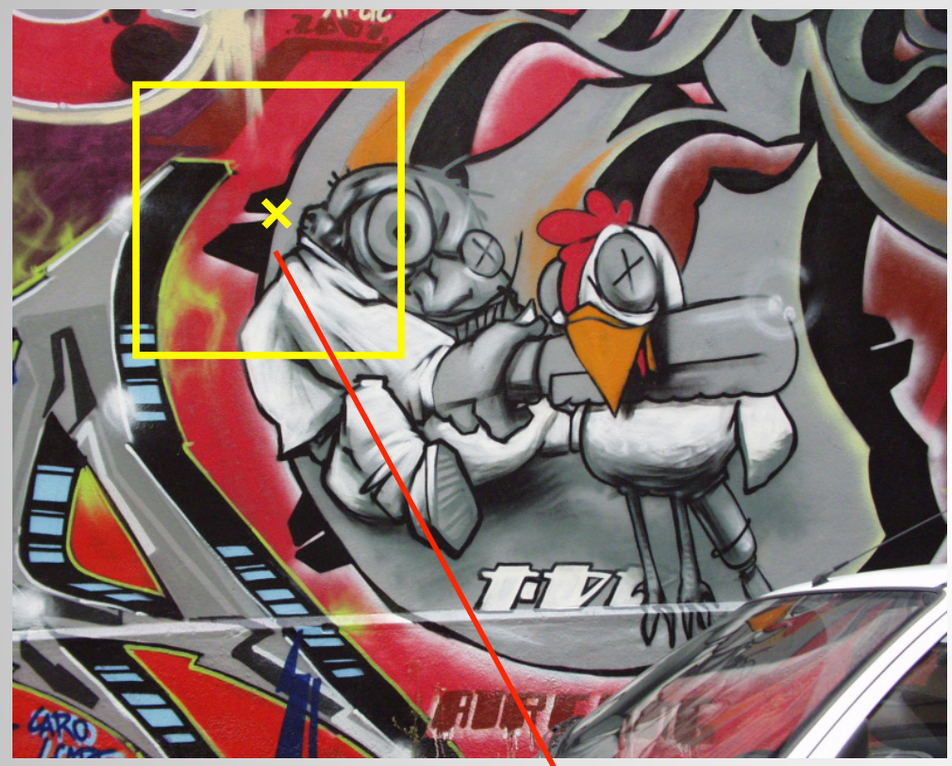


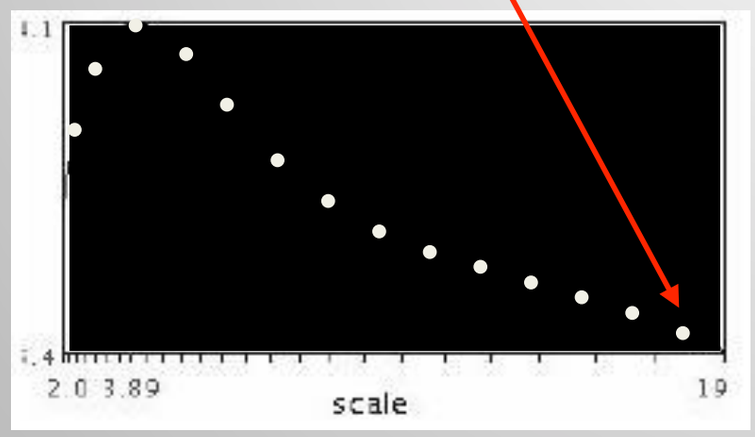
$f(I_{i_1 \ldots i_m}(x, \sigma))$



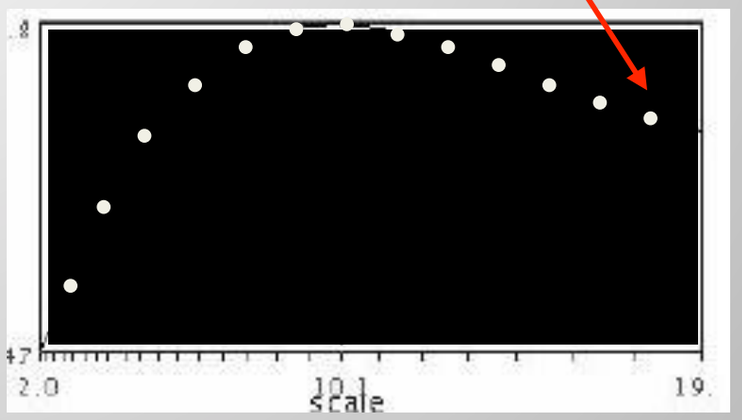
$f(I_{i_1 \ldots i_m}(x', \sigma))$

Patch Size Corresponding to Scale





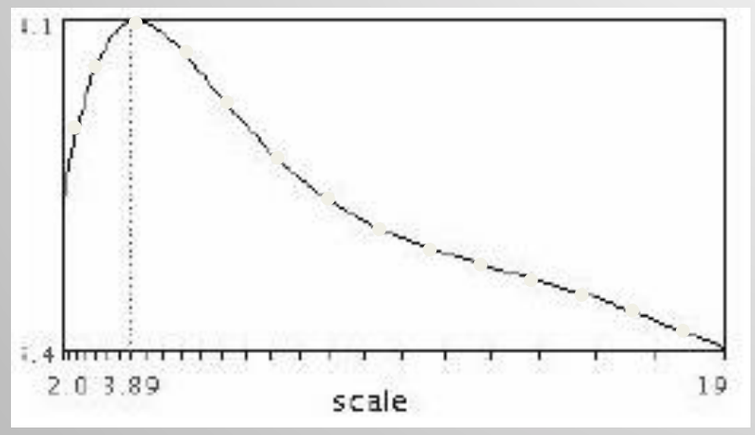
$f(I_{i_1 \ldots i_m}(x, \sigma))$



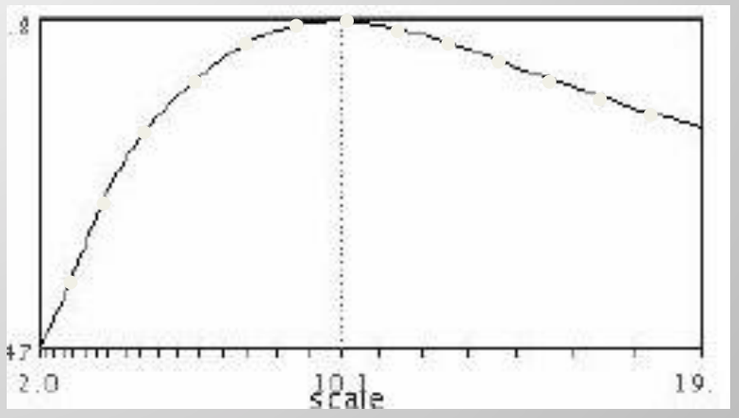
$f(I_{i_1 \ldots i_m}(x', \sigma))$

Patch Size Corresponding to Scale

- Function responses for increasing scale (scale signature)

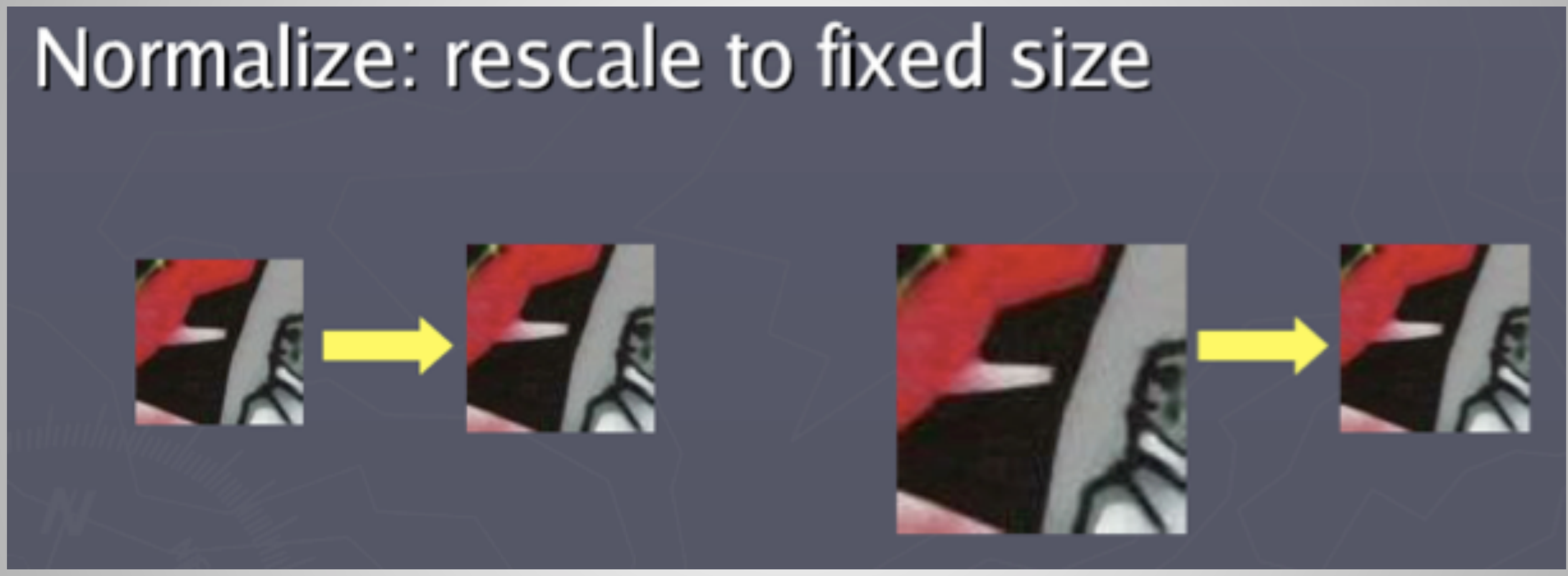


$$f(I_{i_1 \ldots i_m}(x,\sigma))$$



$$f(I_{i_1 \ldots i_m}(x',\sigma'))$$

Normalization



Scale Invariant Feature Transform (SIFT)

- Lowe., D. 2004, IJCV

cited > 25K

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

Received January 10, 2003; Revised January 7, 2004; Accepted January 22, 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 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.

Keywords: invariant features, object recognition, scale invariance, image matching

1. Introduction

Image matching is a fundamental aspect of many problems in computer vision, including object or scene recognition, solving for 3D structure from multiple images, stereo correspondence, and motion tracking. This paper describes image features that have many properties that make them suitable for matching differing images of an object or scene. The features are invariant to image scaling and rotation, and partially invariant to change in illumination and 3D camera viewpoint. They are well localized in both the spatial and frequency domains, reducing the probability of disruption by occlusion, clutter, or noise. Large numbers of features can be extracted from typical images with efficient algorithms. In addition, the features are highly distinctive, which allows a single feature to be correctly matched with high probability against a large database of features, providing a basis for object and scene recognition.

The cost of extracting these features is minimized by taking a cascade filtering approach, in which the more expensive operations are applied only at locations that pass an initial test. Following are the major stages of computation used to generate the set of image features:

1. *Scale-space extrema detection*: The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation.
2. *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.
3. *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.

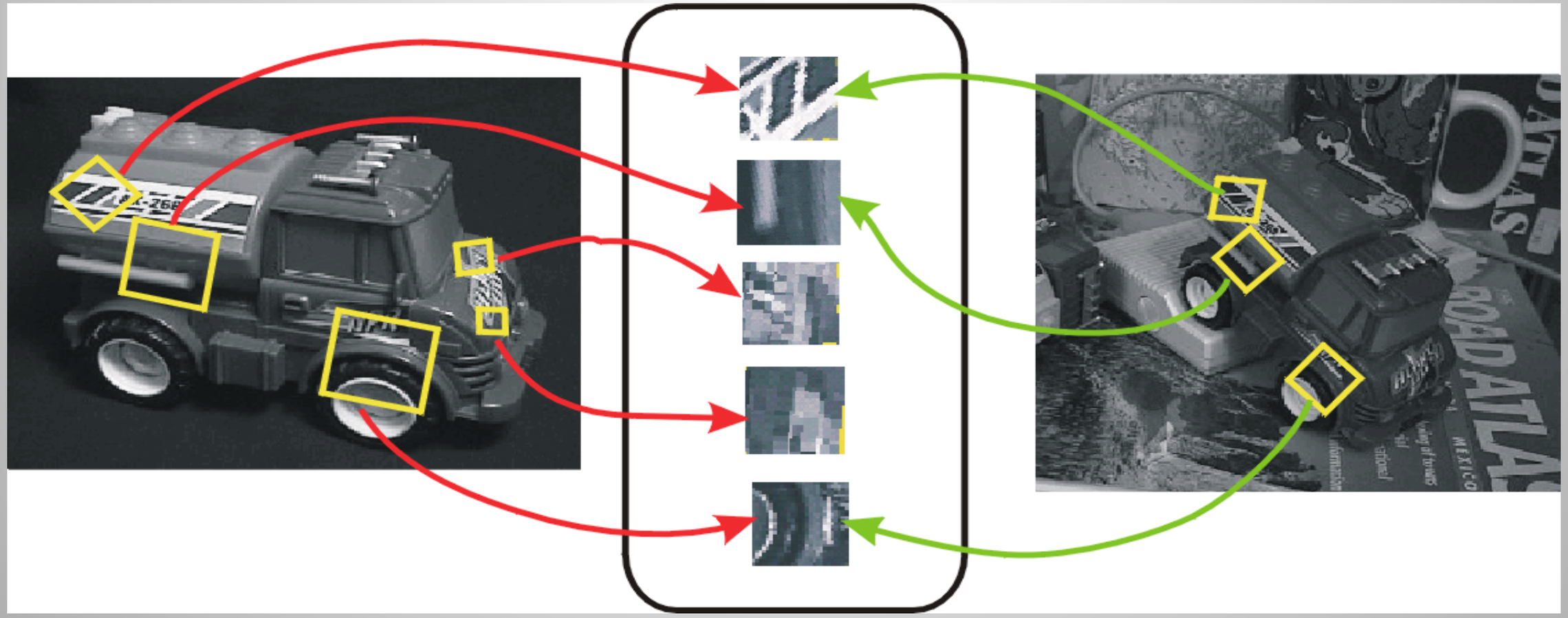
Scale Invariant Feature Transform (SIFT)

- Shows existence, importance, and value of invariant types of detector
- Demonstrates the richness that feature descriptors can bring to feature matching

Instead of using LoG (Laplacian of Gaussian) like in Harris and Hessian-based operators, SIFT uses DoG (Difference of Gaussian).

Scale Invariant Feature Transform (SIFT)

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



Overall Procedure at a High Level

Scale-Space Extrema Detection — Search over multiple scales and image locations

Keypoint Localization — Fit a model to determine location and scale. Select Keypoints based on a measure of stability.

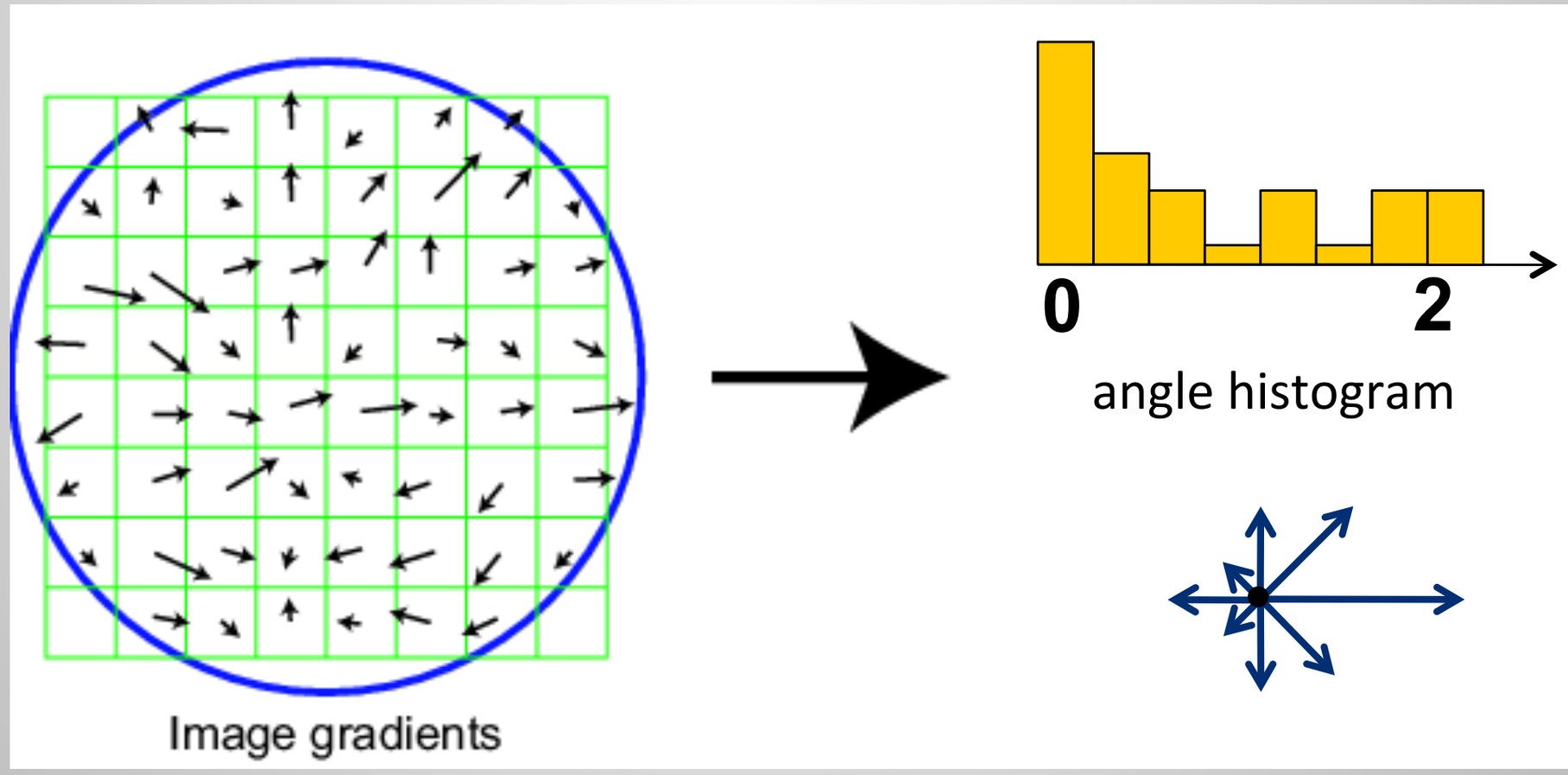
Orientation Assignment — Compute best orientation(s) for each keypoint region.

Keypoint Description — Use local image gradients at selected scale and rotation to describe each keypoint region.

SIFT Descriptor

Basic idea:

- Take n x n (i.e., n=16) square window (around a *feature/interest point*)
- Divide them into m x m (i.e, m=4) cells
- Compute gradient orientation for each cell
- Create histogram over edge orientations weighted by magnitude



SIFT Descriptor

16 histograms x 8 orientations
= 128 features

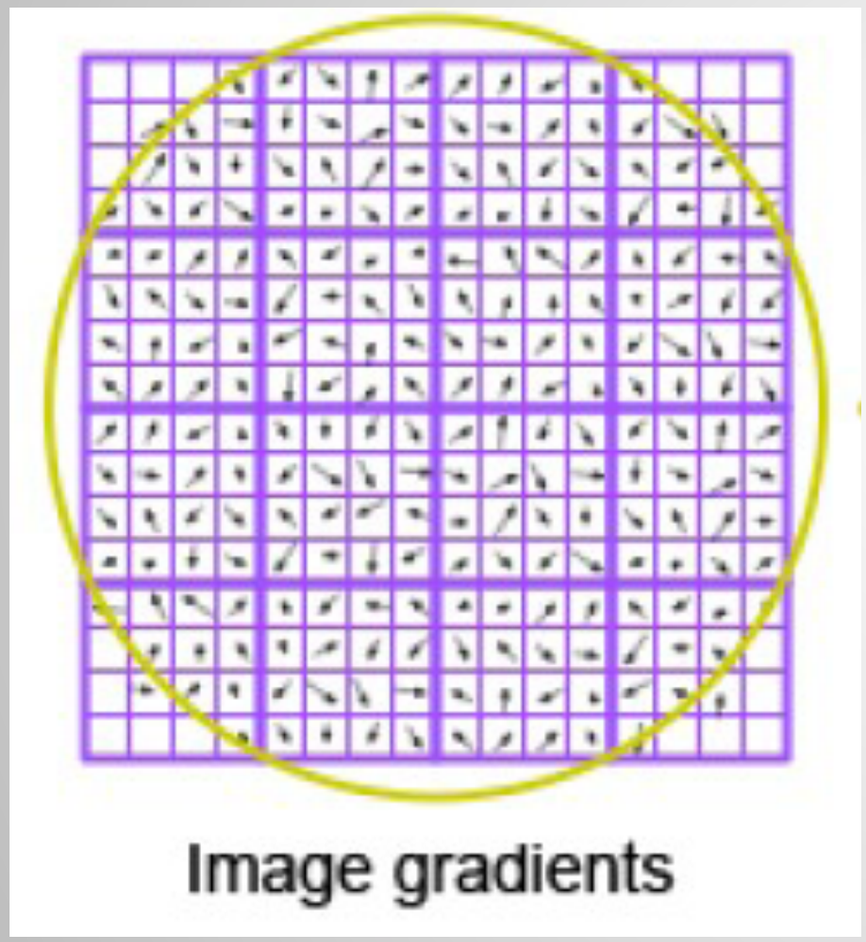
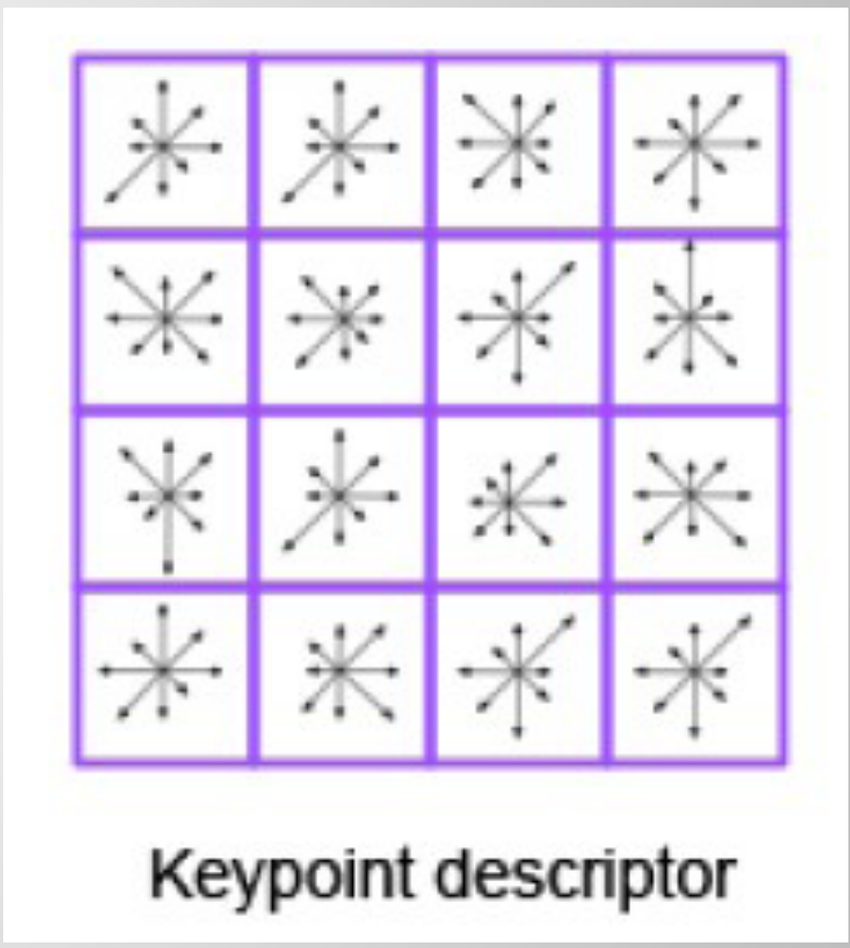


Image gradients



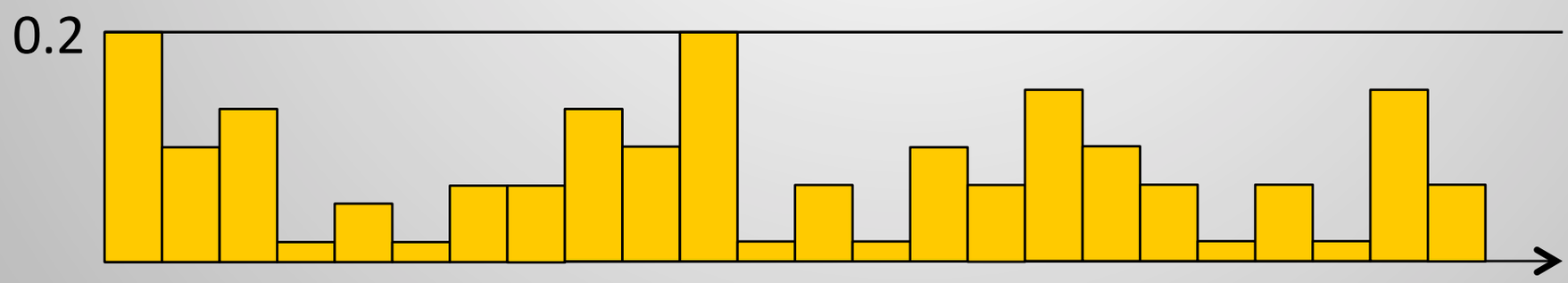
Keypoint descriptor

SIFT Descriptor

Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor
- Threshold normalize the descriptor:

$$\sum_i d_i^2 = 1$$ such that: $$d_i < 0.2$$



Revisit SIFT Steps

(1) Scale-space extrema detection

– Extract scale and rotation invariant interest points (i.e., keypoints).

(2) Keypoint localization

– Determine location and scale for each interest point.

– *Eliminate "weak" keypoints*

(3) Orientation assignment

– Assign one or more orientations to each keypoint.

(4) Keypoint descriptor

– Use local image gradients at the selected scale.

(1) Scale-Space Extrema Detection

- We want to find points that give us information about the objects in the image.

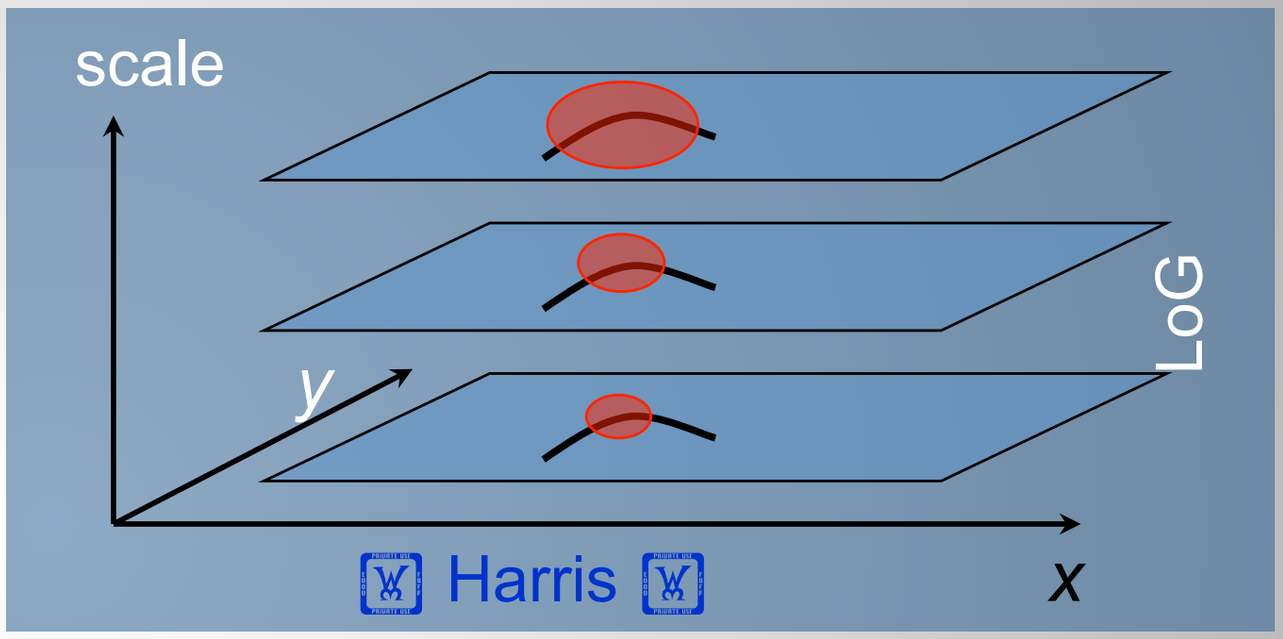
- The information about the objects is in the object's **edges**.

- We will represent the image in a way that gives us these edges as this representations extrema points.

(1) Scale-Space Extrema Detection

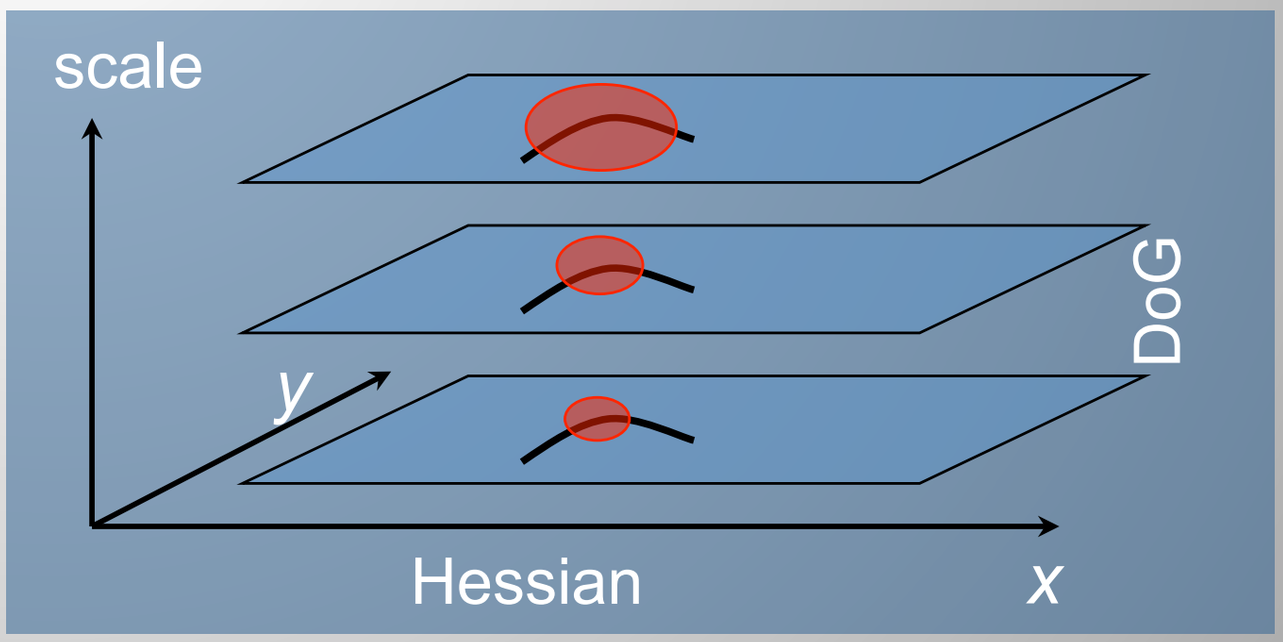
- Harris-Laplace

Find local maxima of:
- Harris detector in space
- LoG in scale

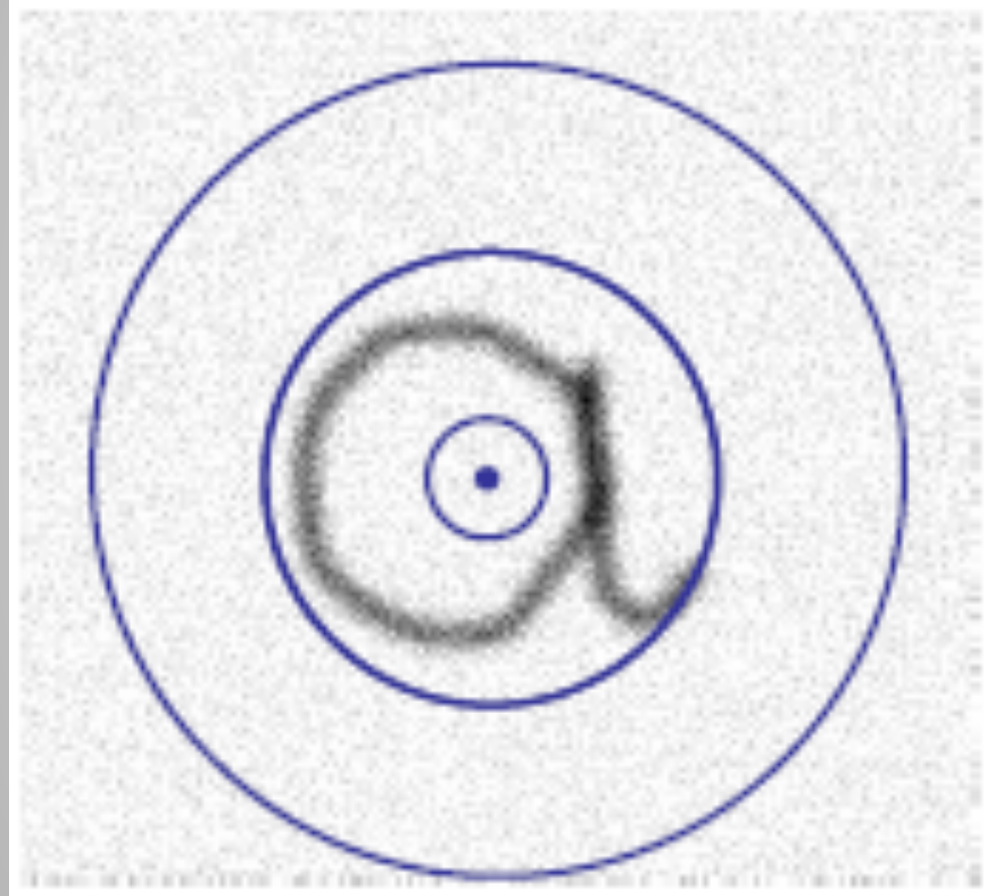


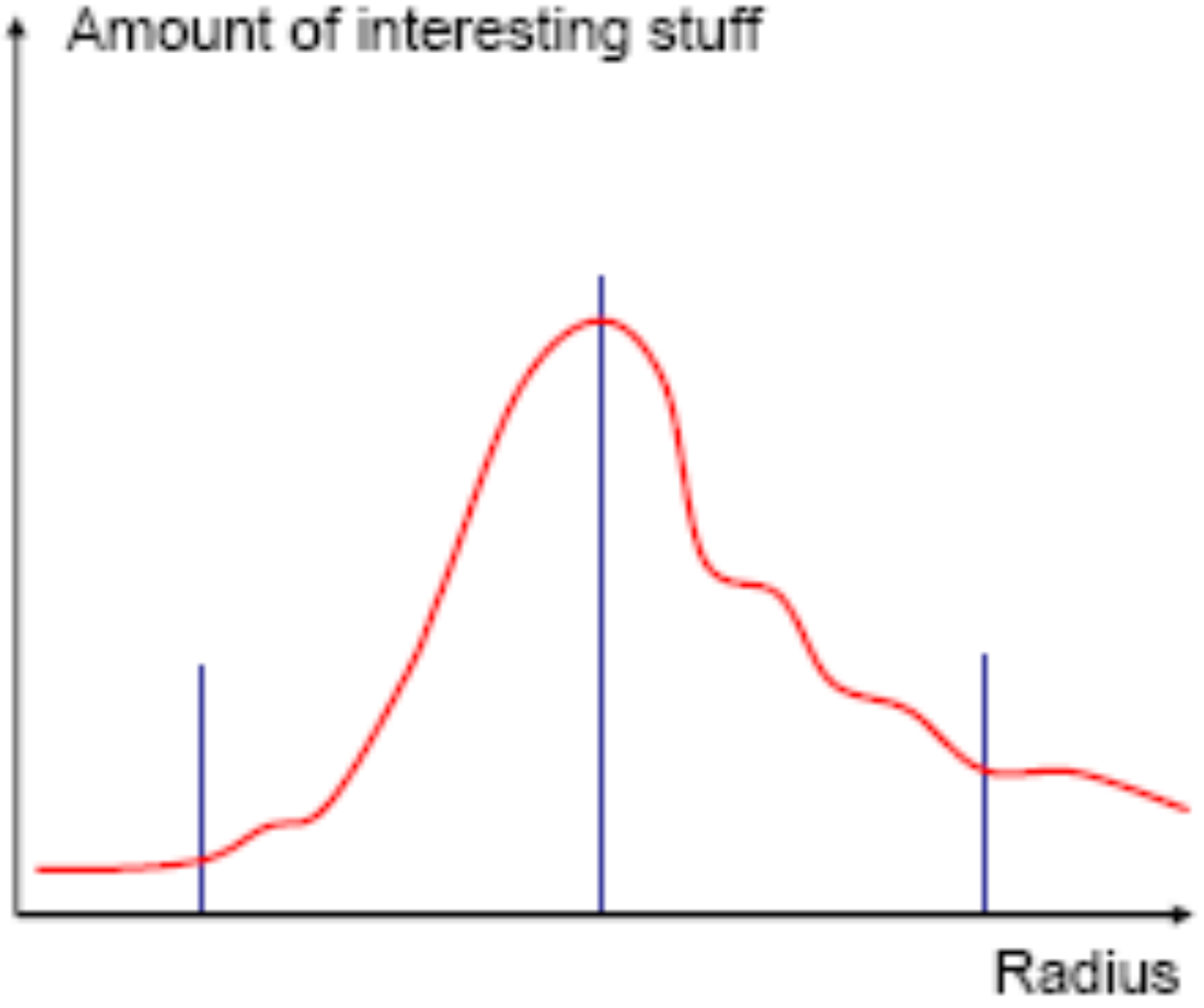
- SIFT

Find local maxima of:
- Hessian in space
- DoG in scale

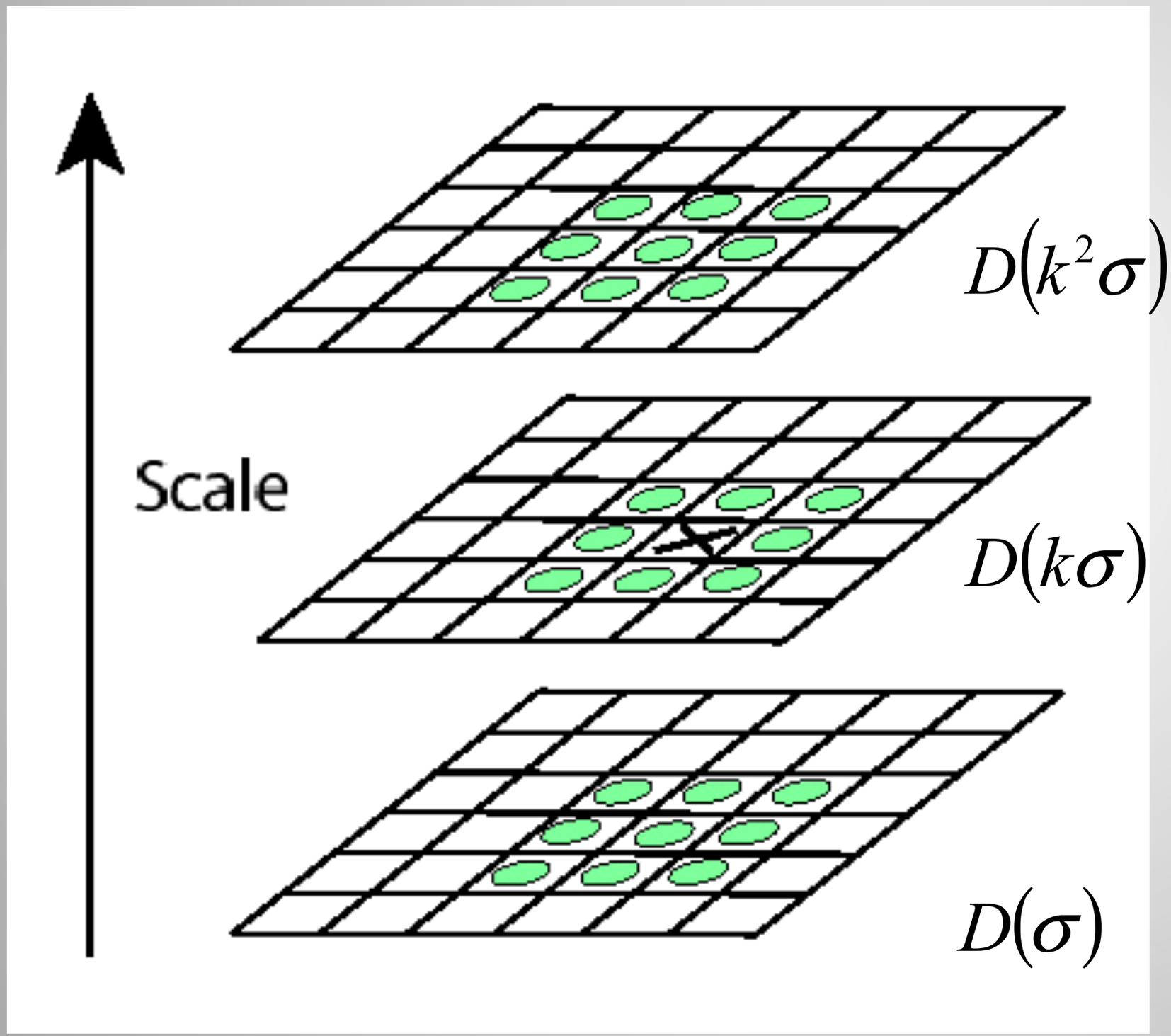


(1) Scale-Space Extrema Detection





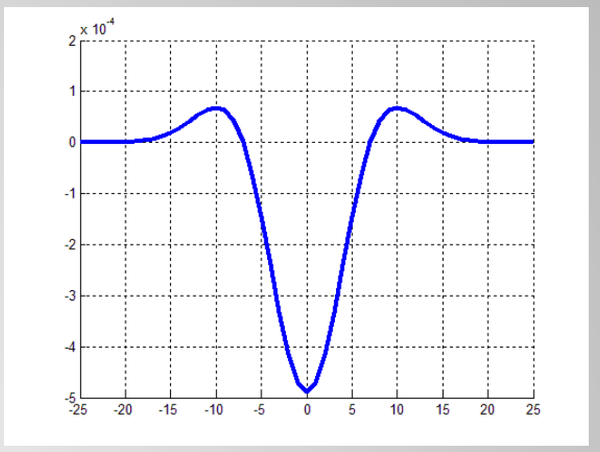
(1) Scale-Space Extrema Detection

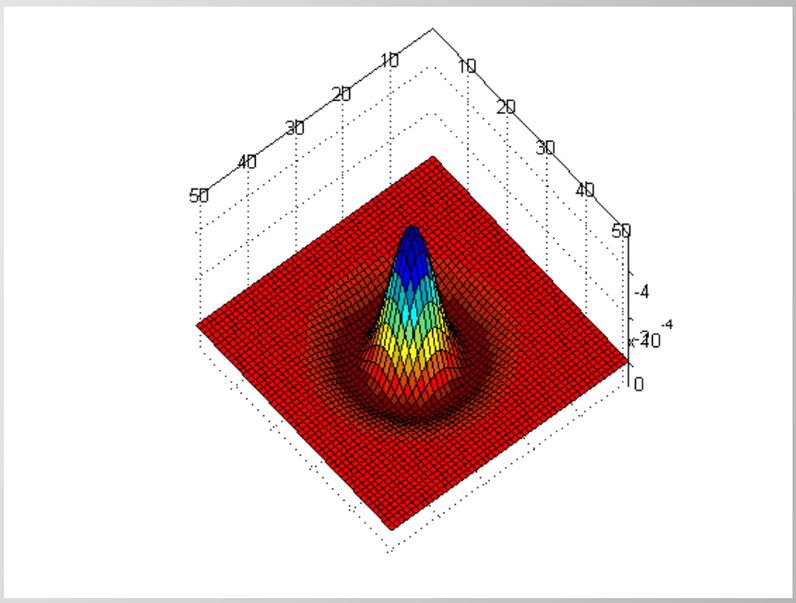


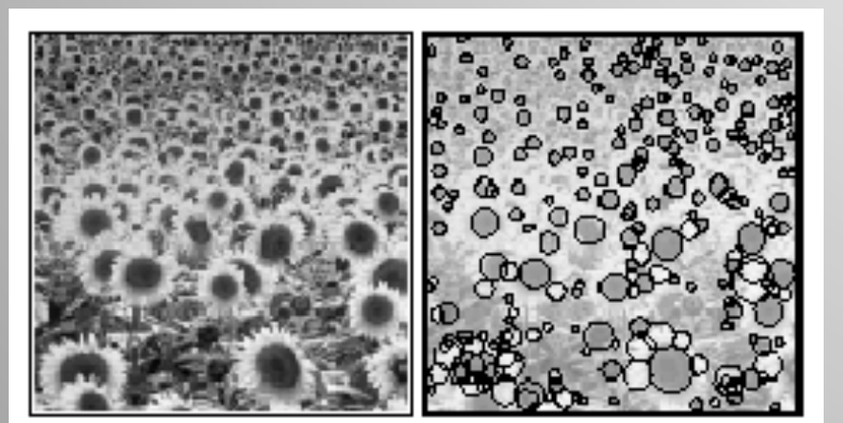
- Extract local extrema (i.e., minima or maxima) in DoG pyramid.
  - -Compare each point to its 8 neighbors at the same level, 9 neighbors in the level above, and 9 neighbors in the level below (i.e., 26 total).

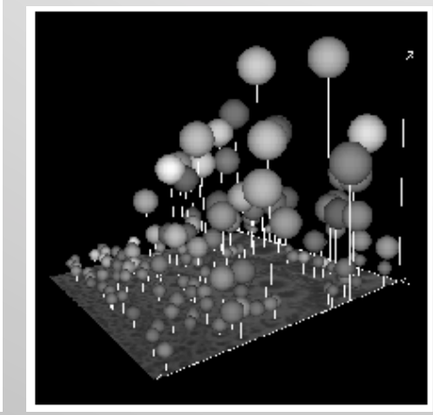
(1) Scale-Space Extrema Detection

- **Laplacian of Gaussian** kernel
  - Scale normalized (x by scale^2)
  - Proposed by Lindeberg
- Scale-space detection
  - Find local maxima across scale/space
  - A good "blob" detector









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

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

(2) Keypoint Localization

- There are still a lot of points, some of them are not good enough.
- The locations of keypoints may be not accurate.
- Eliminating edge points.

(2) Keypoint Localization

- Reject (1) points with low contrast (flat)
  (2) poorly localized along an edge (edge)

(2) Keypoint Localization

- Reject (1) points with low contrast (flat)
  (2) poorly localized along an edge (edge)

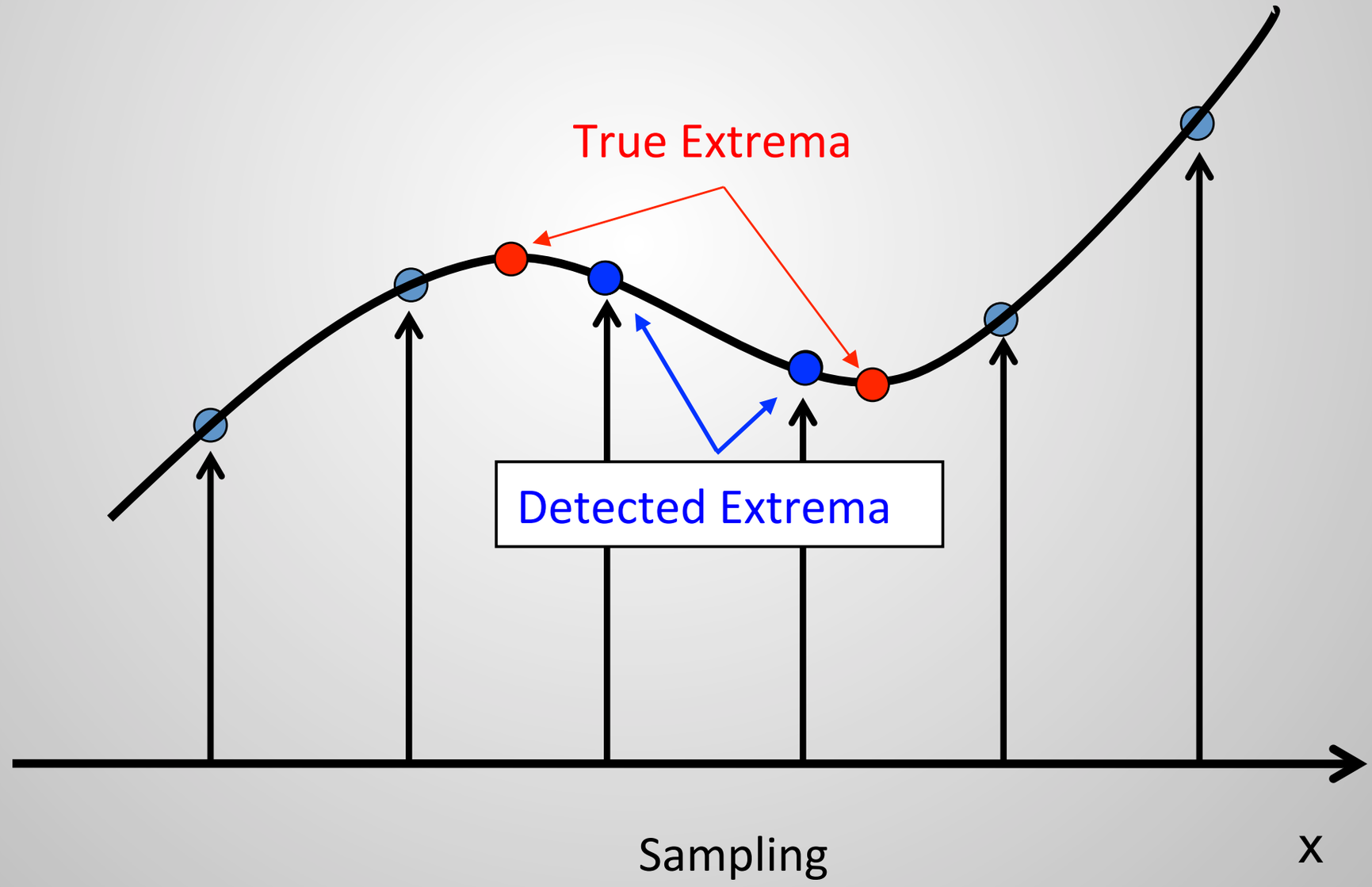
if $$\frac{Tr(H)}{|H|} < \frac{(r+1)^2}{r}$$

Reject

Where $$r = \frac{\lambda_1}{\lambda_2}$$ and practically SIFT uses r=10.

Inaccurate Keypoint Localization

- Poor contrast



Inaccurate Keypoint Localization

- The Solution:
  - Taylor expansion:

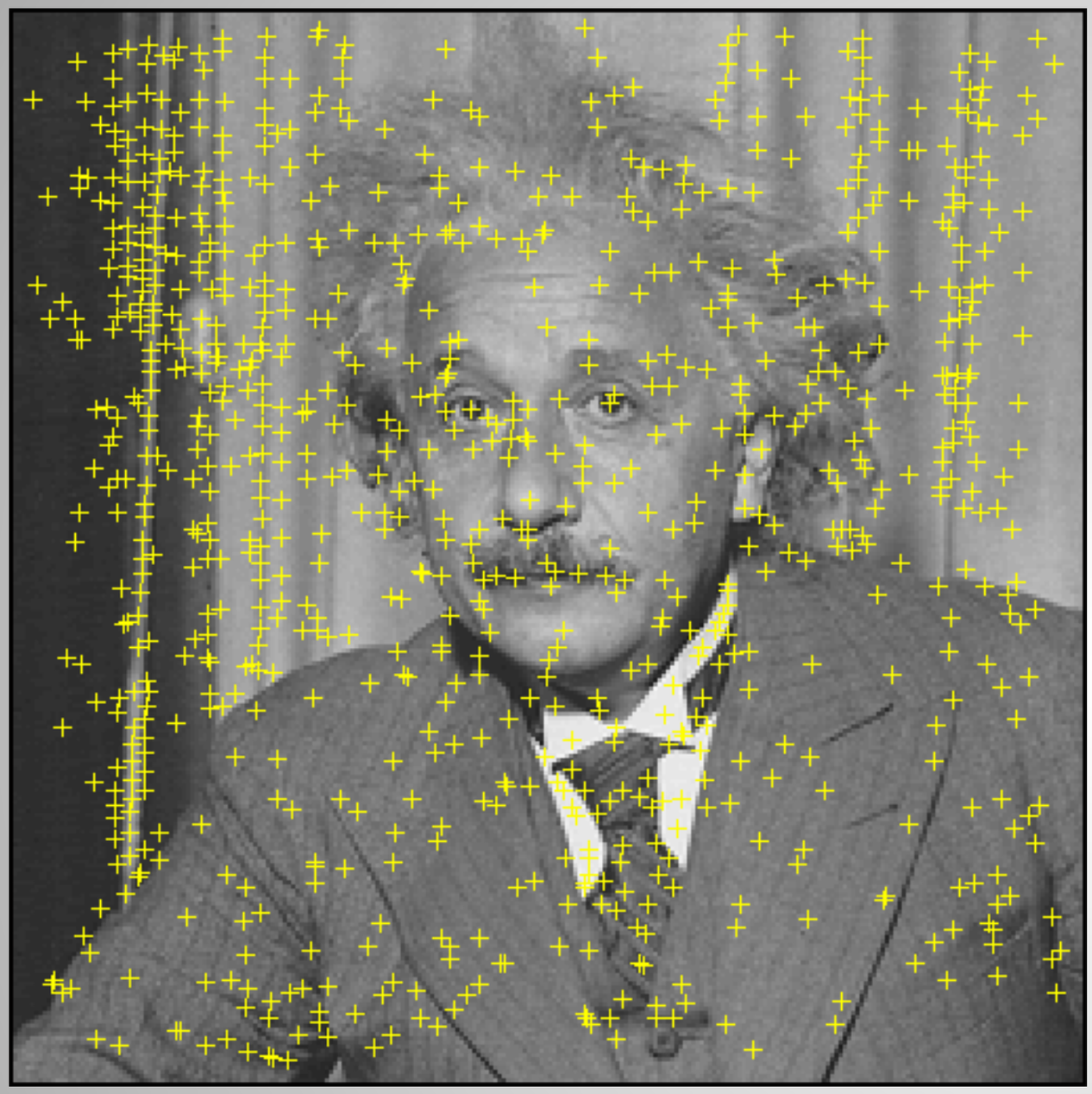
$$D(\vec{x}) = D + \frac{\partial D^T}{\partial \vec{x}} \vec{x} + \frac{1}{2} \vec{x}^T \frac{\partial^2 D^T}{\partial \vec{x}^2} \vec{x}$$

- Minimize to find accurate extrema:

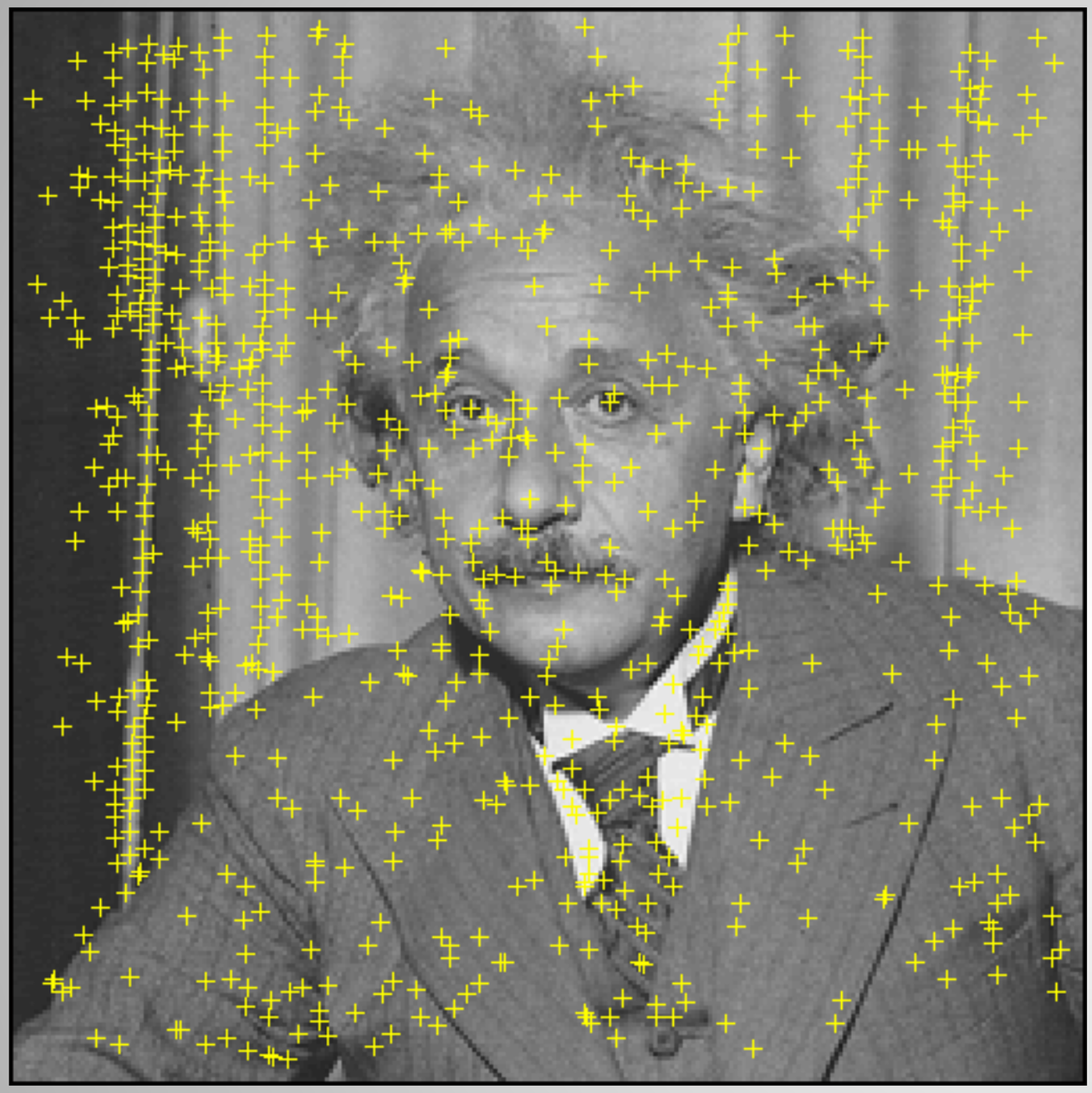
$$\hat{x} = -\frac{\partial^2 D}{\partial \vec{x}^2}^{-1} \frac{\partial D}{\partial \vec{x}}$$

- If offset from sampling point is larger than 0.5 - Keypoint should be in a different sampling point.

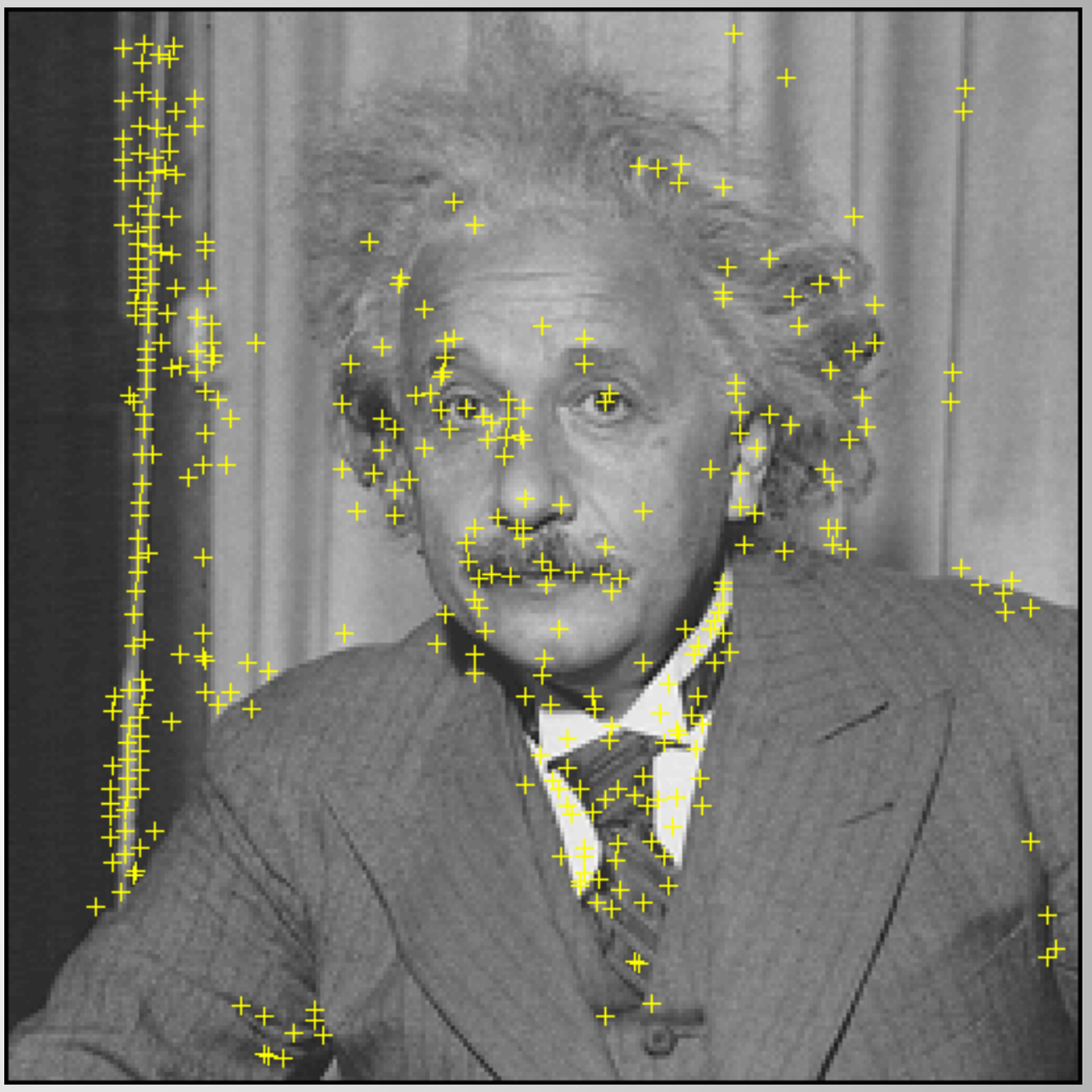
Brown & Lowe 2002



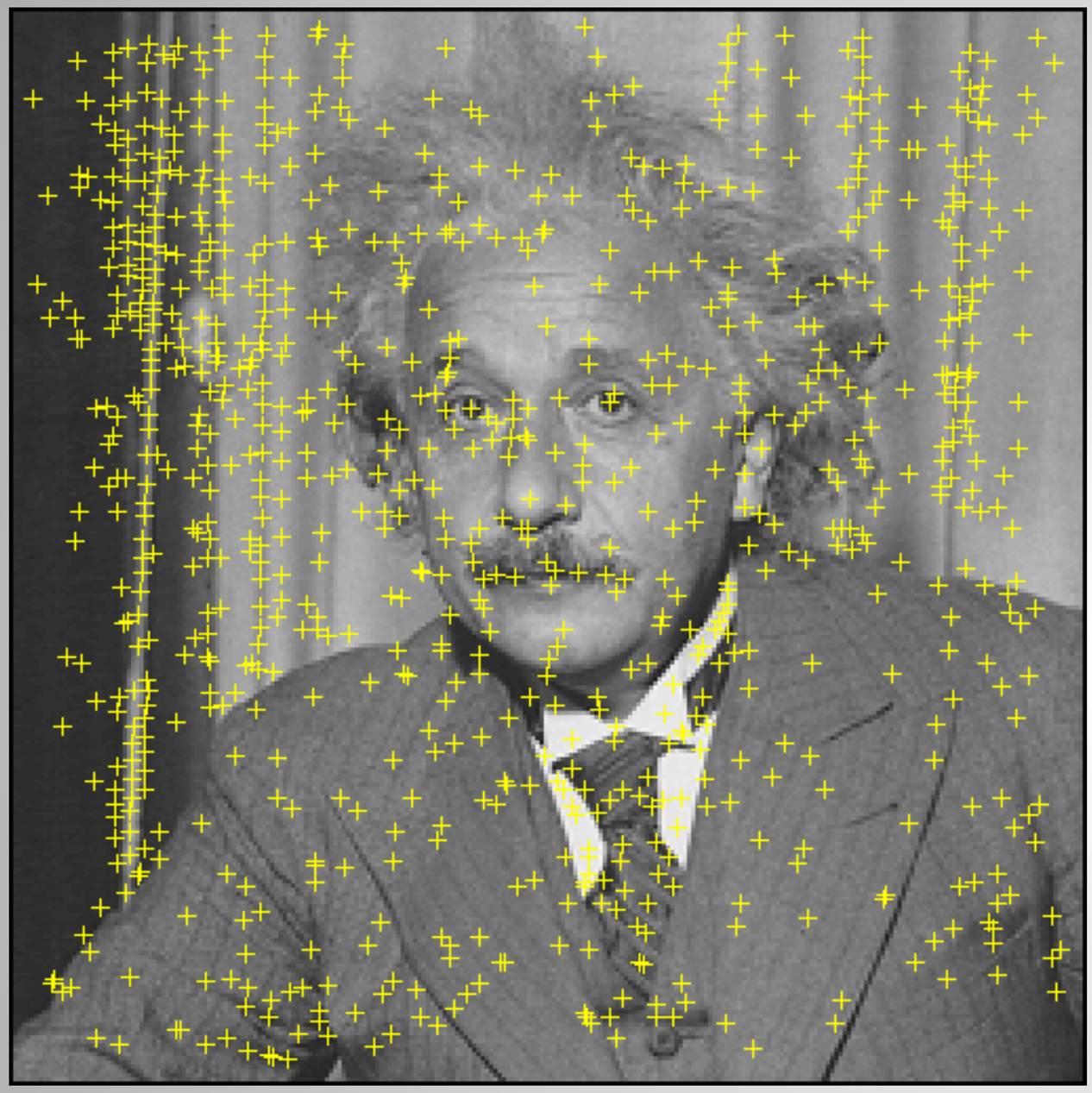
Local extremas



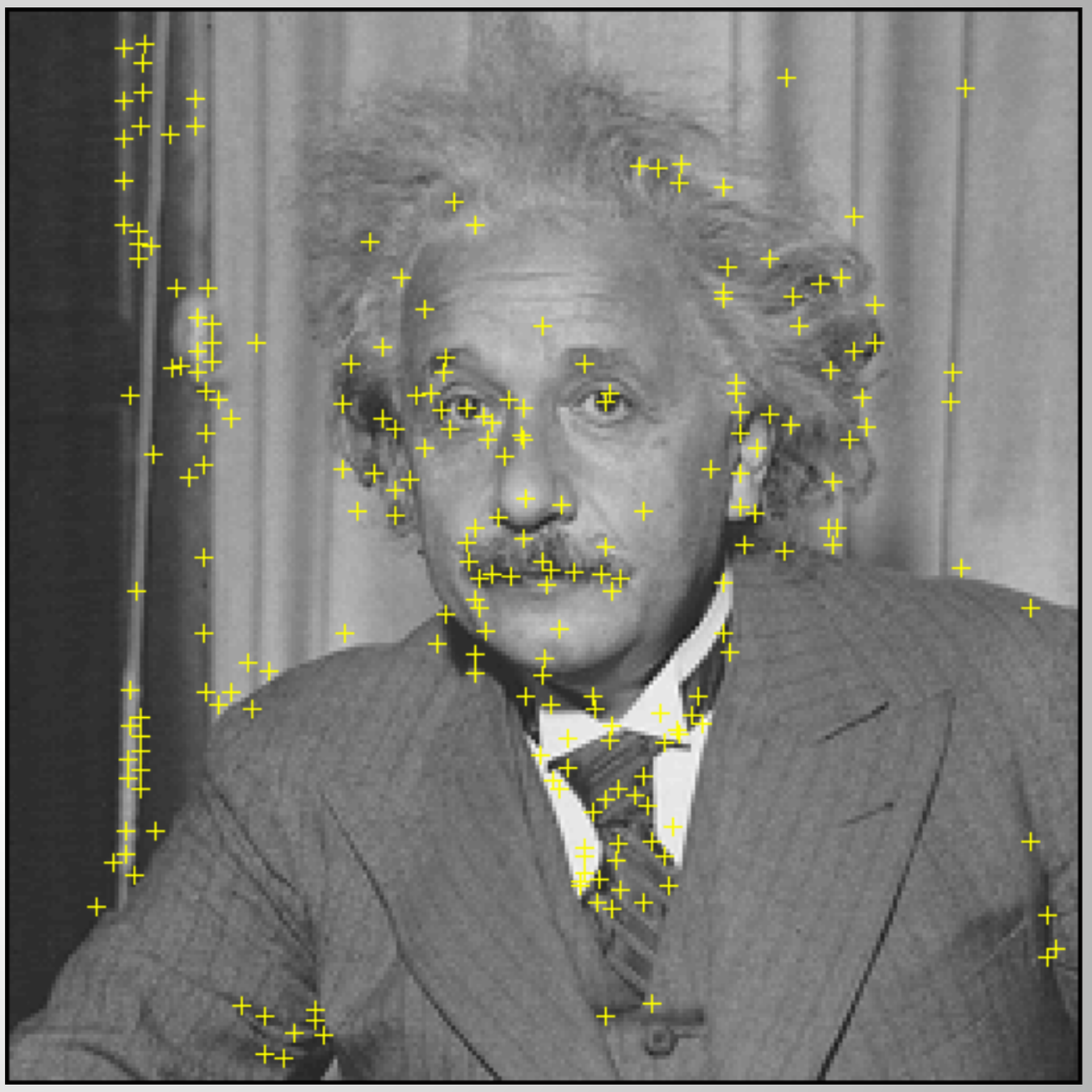
Local extremas



Remove low contrast features

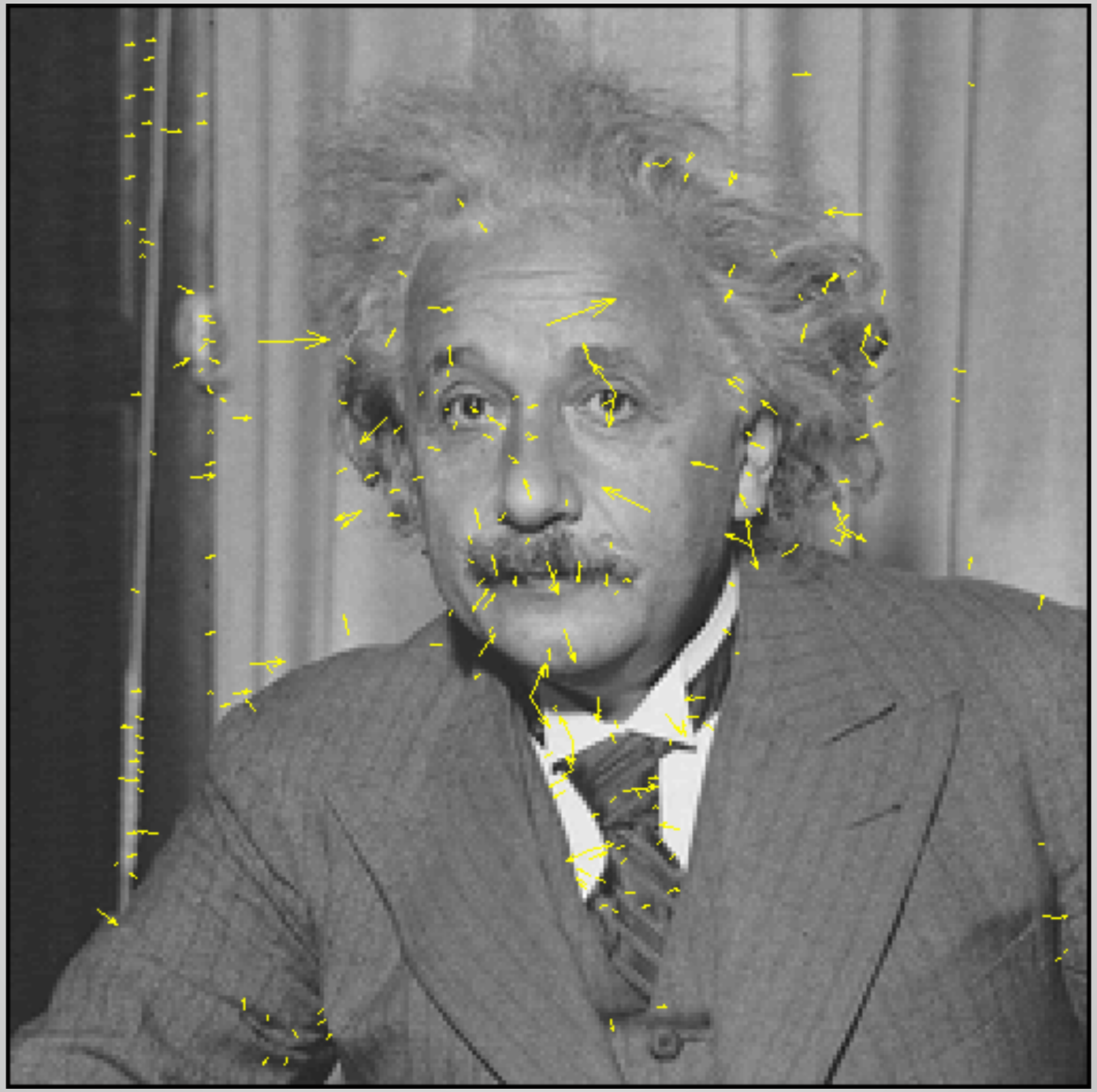


Local extremas



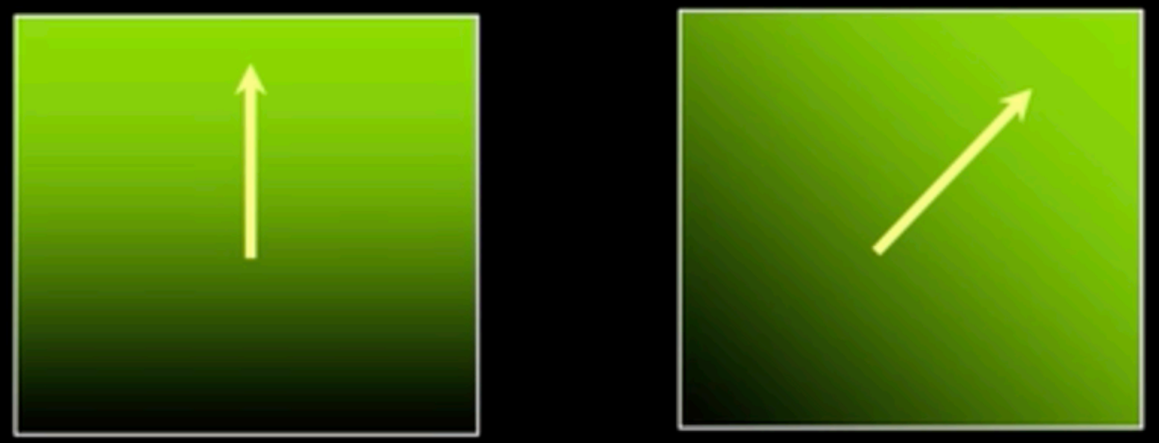
Remove low edges

SIFT Descriptor



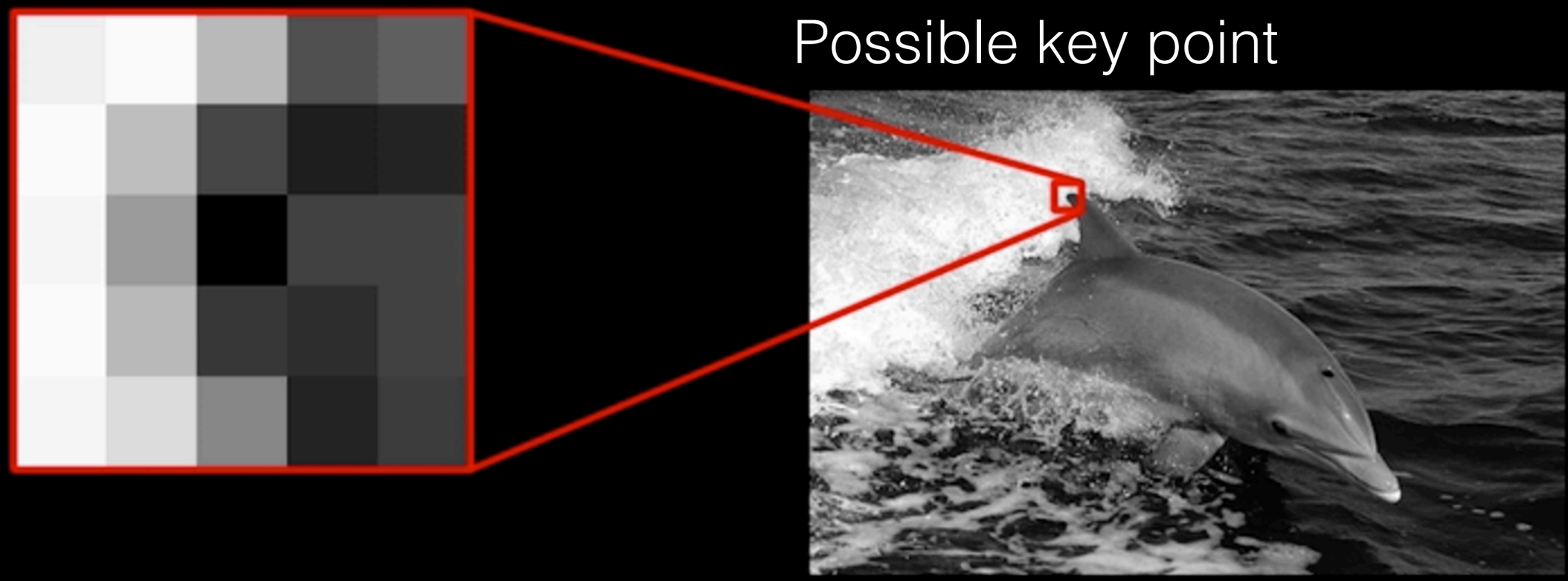
Descriptors Invariant to Rotation

- Find the dominant direction of gradient – that is the base orientation.

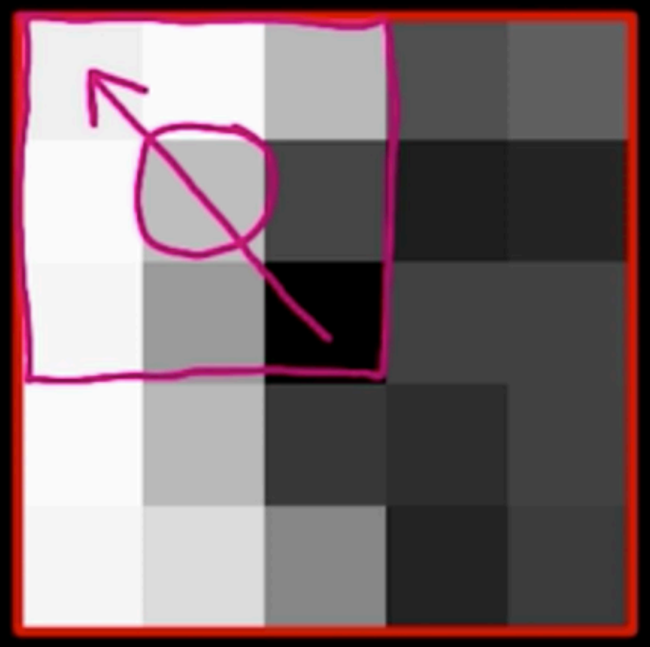


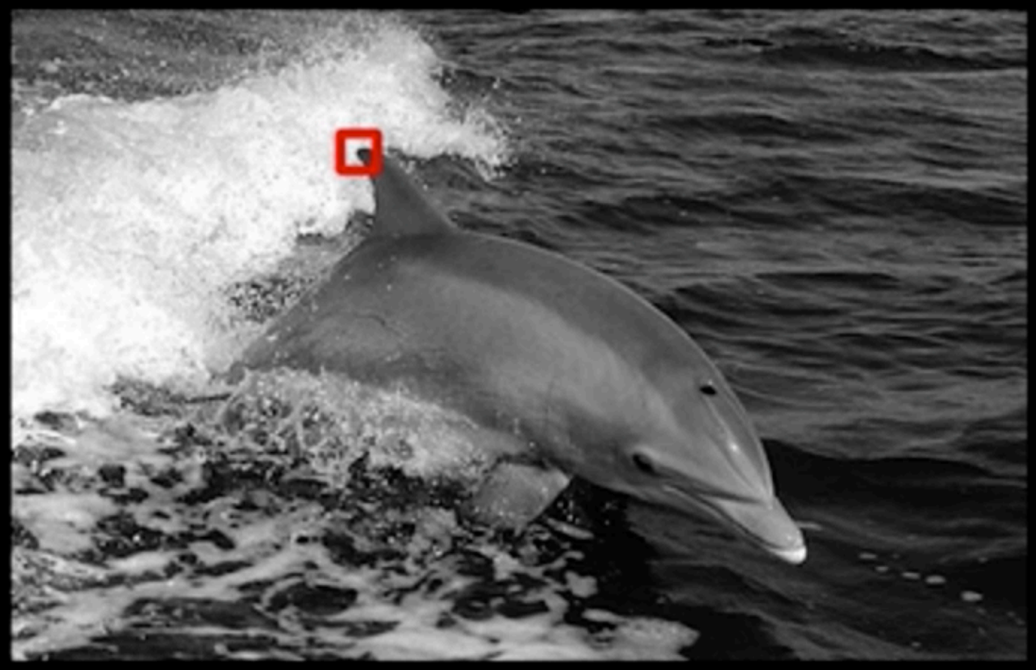
Compute image derivatives relative to this orientation

Dominant Orientation

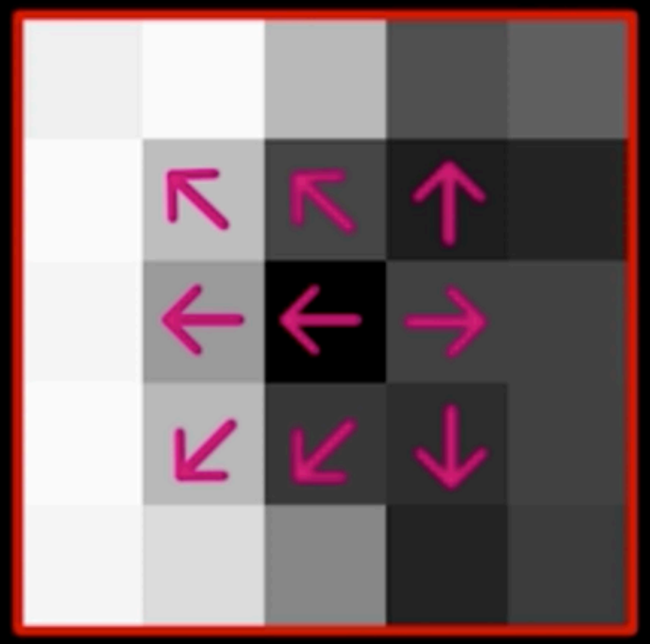


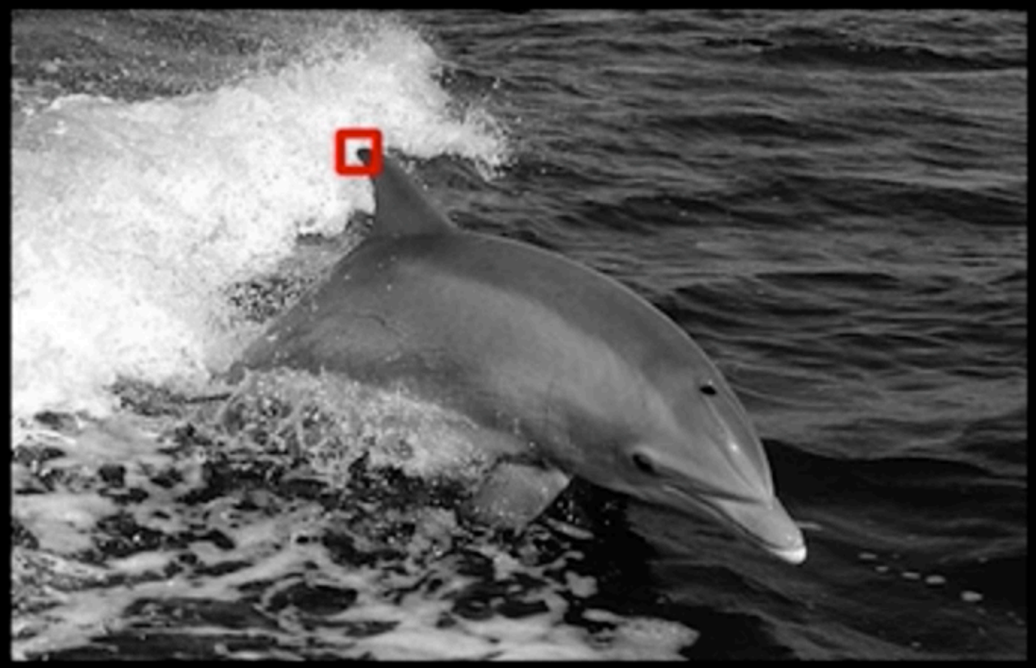
Dominant Orientation



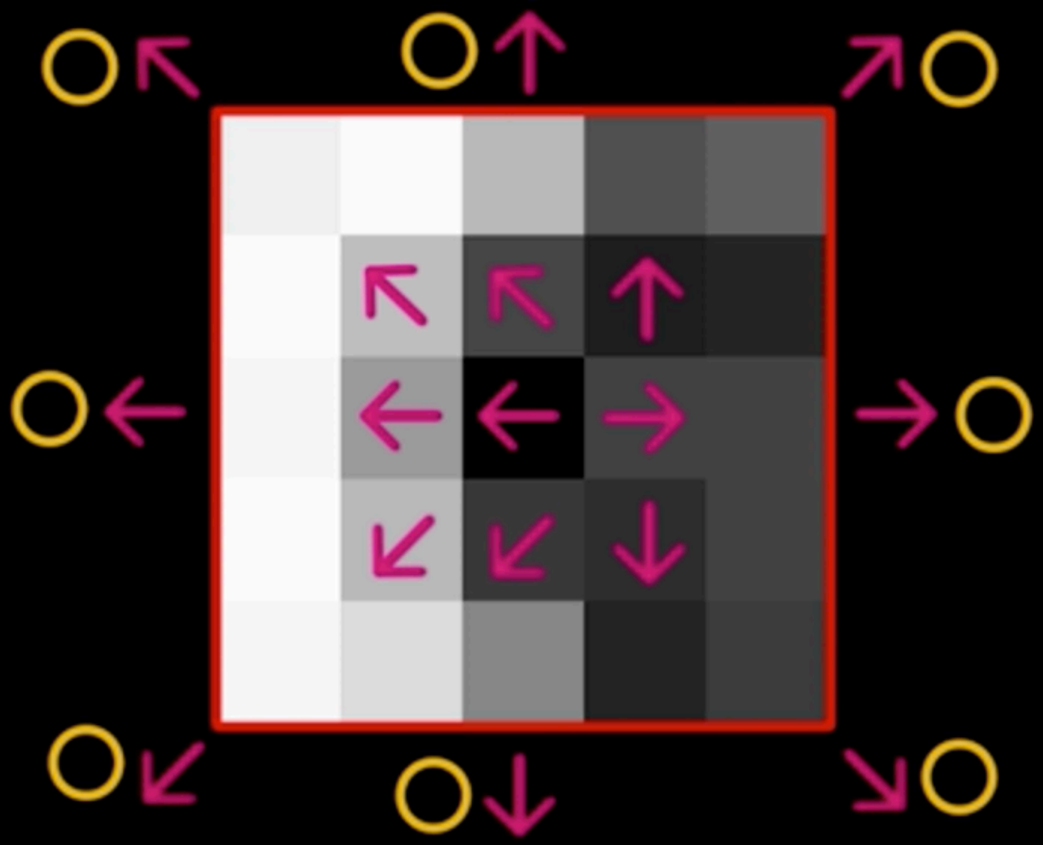


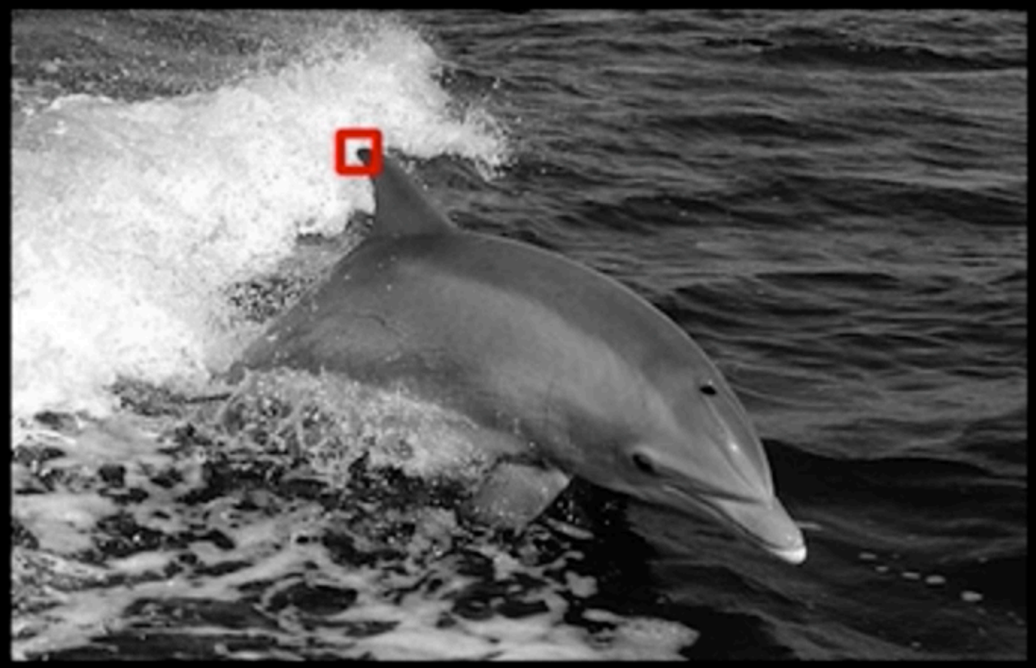
Dominant Orientation



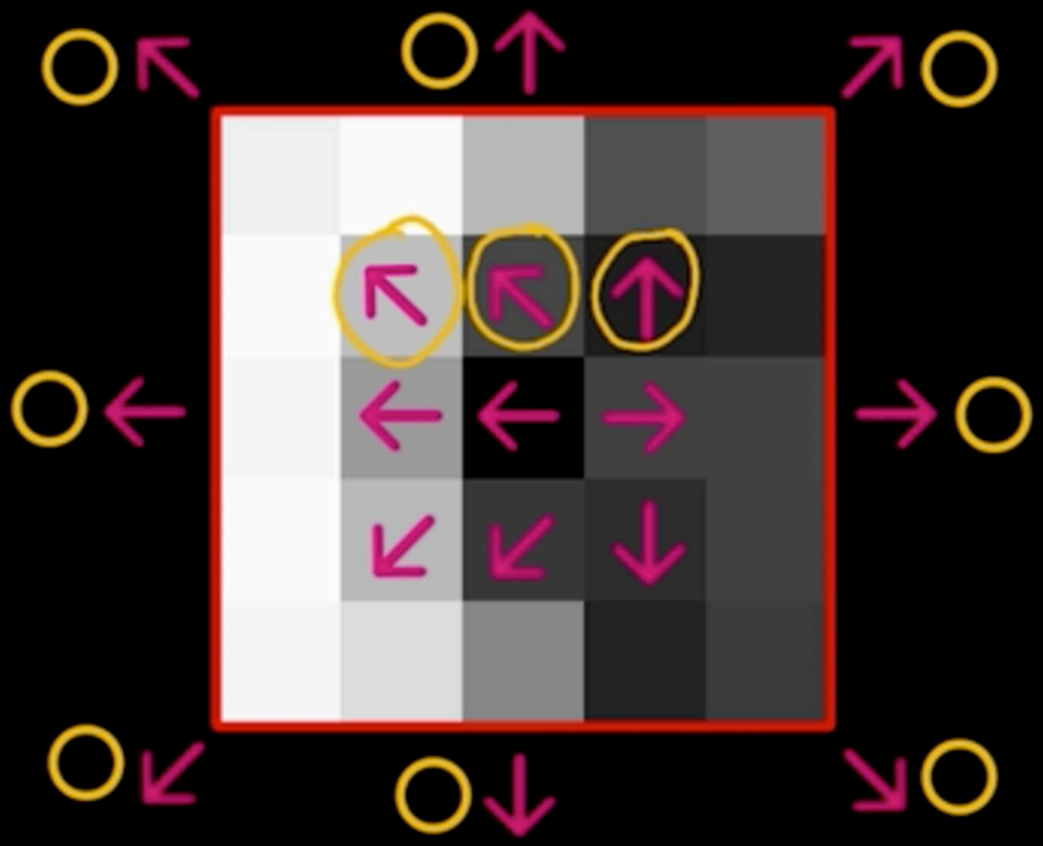


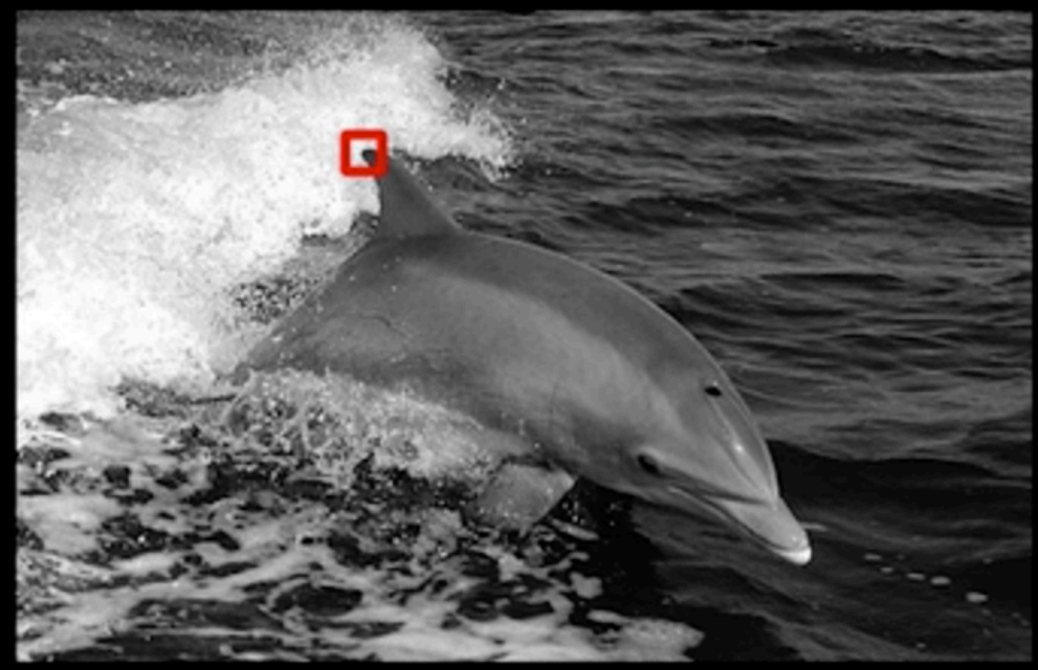
Dominant Orientation



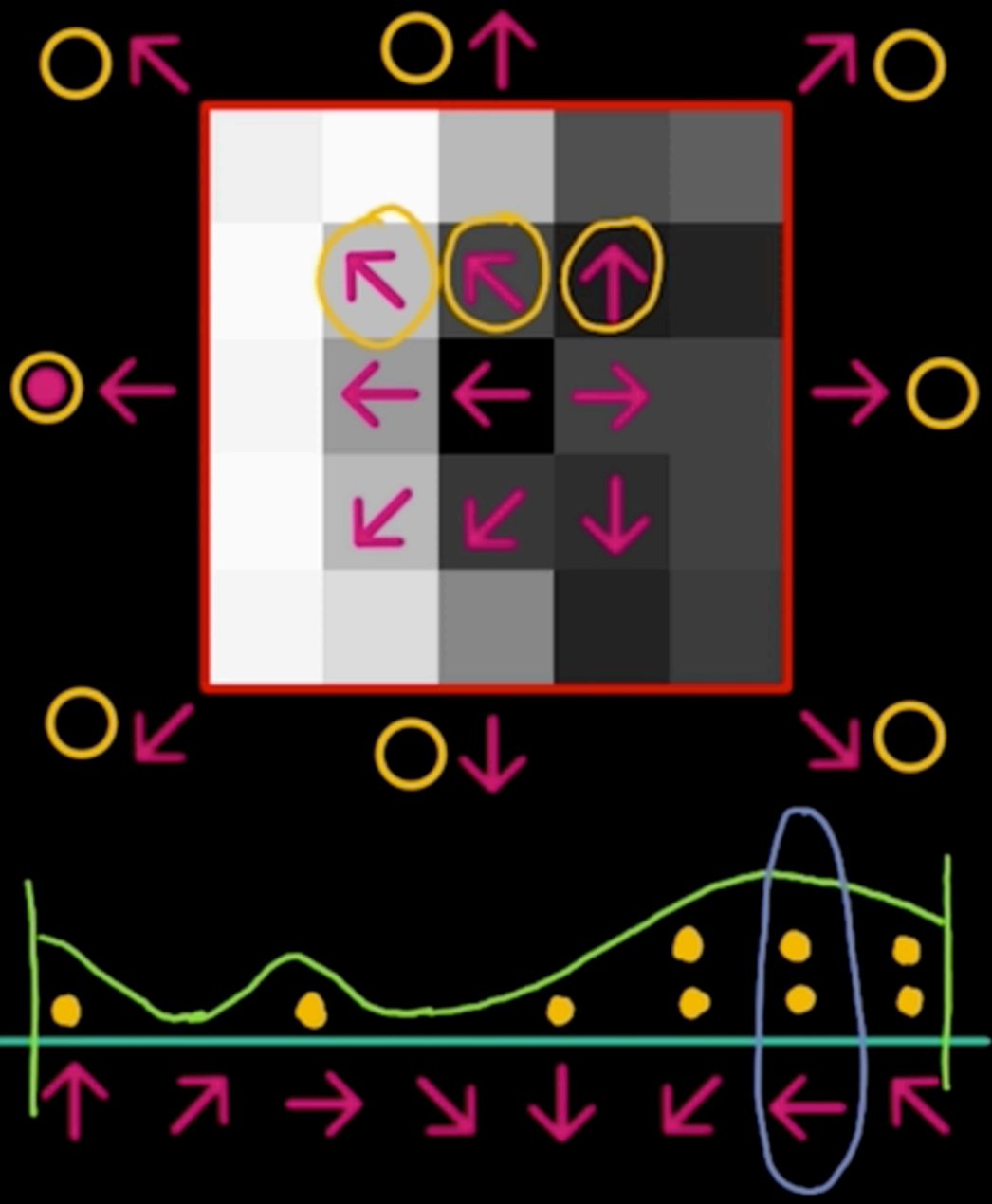


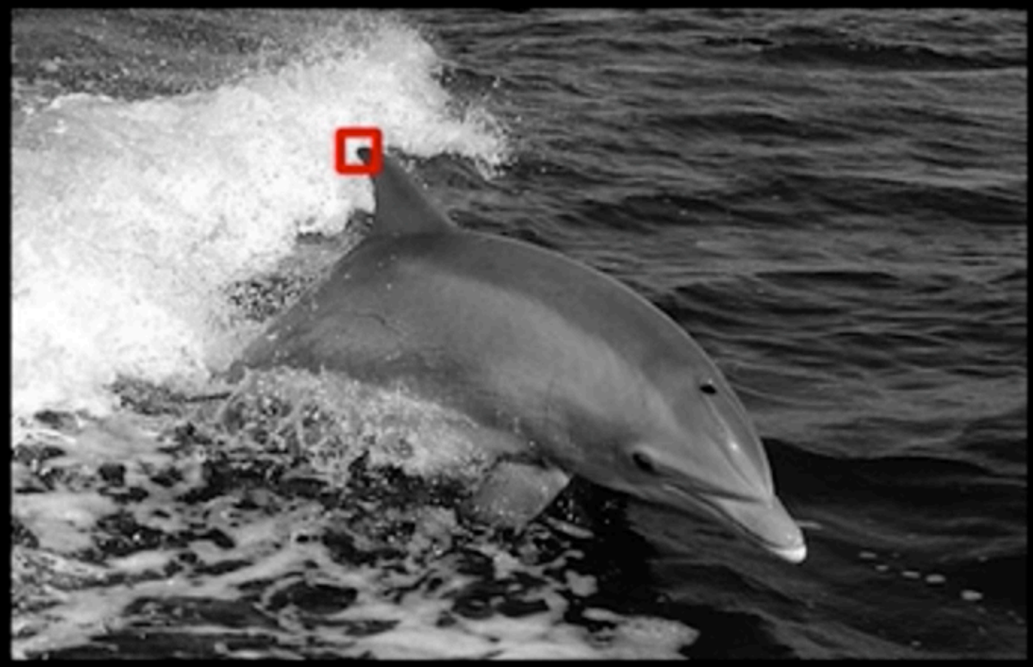
Dominant Orientation



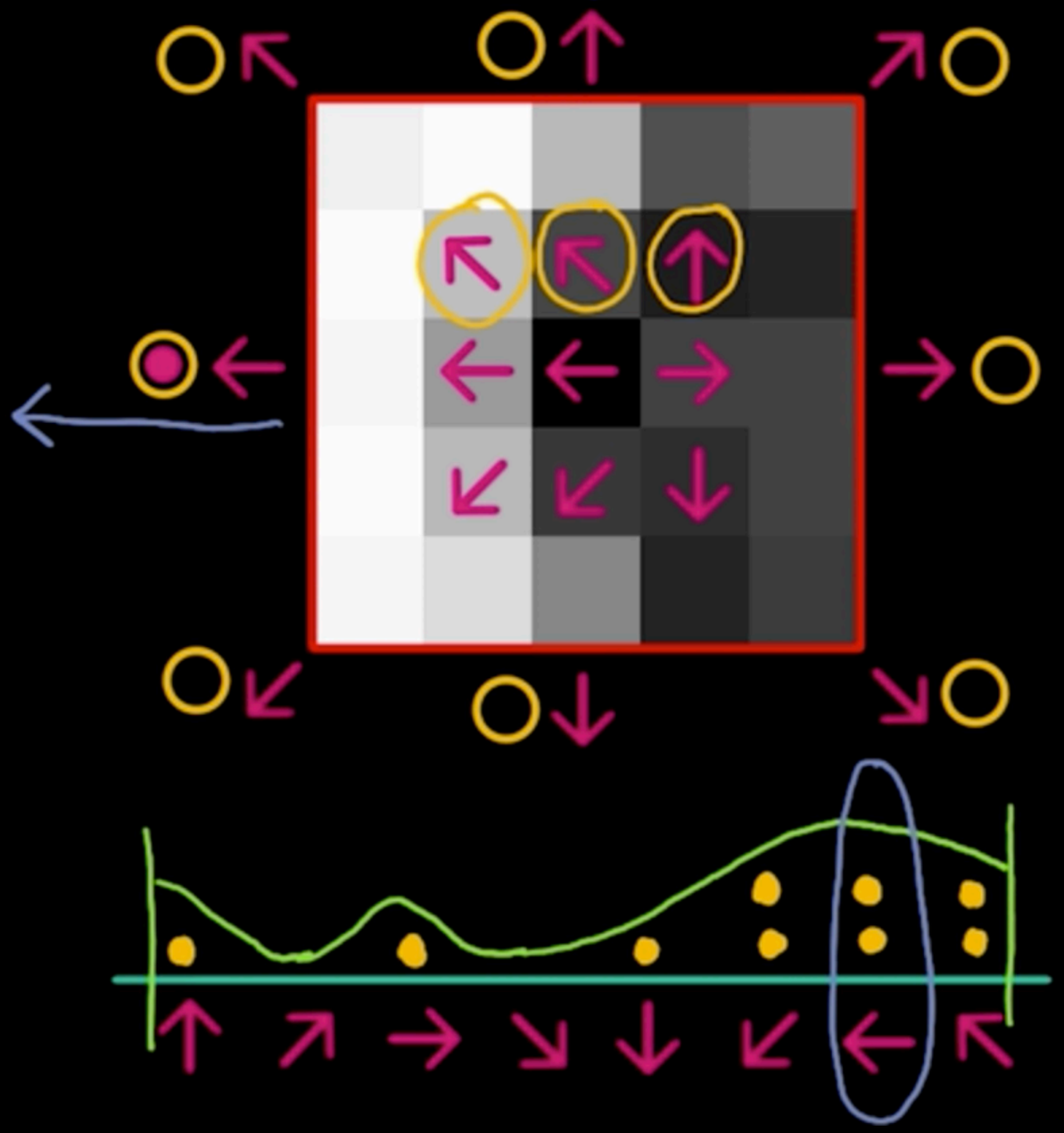


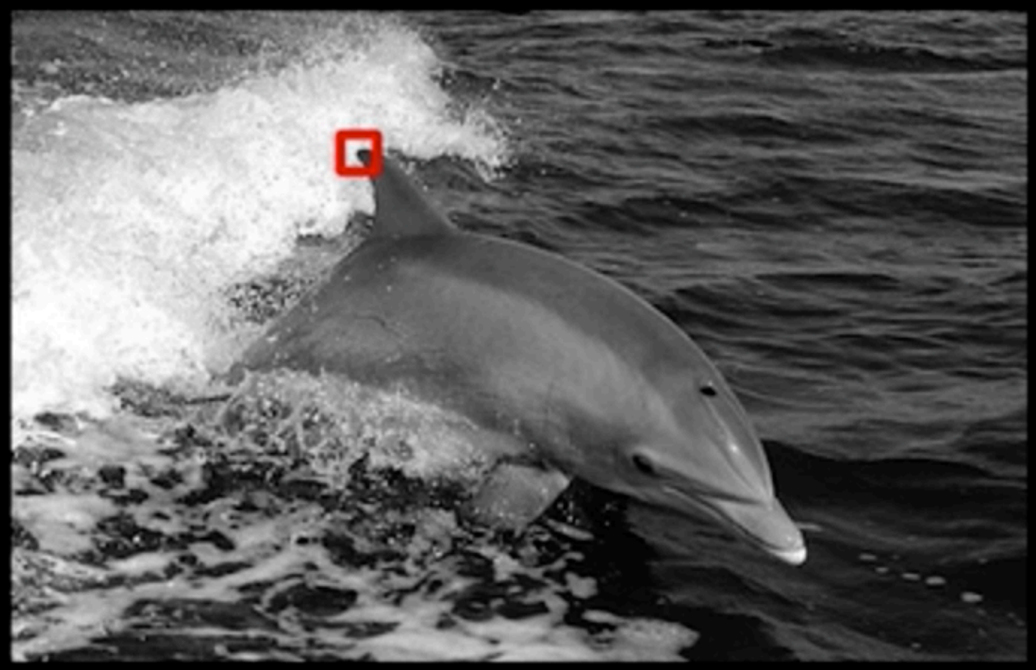
Dominant Orientation





Dominant Orientation





All other directions
recomputed relative to new north

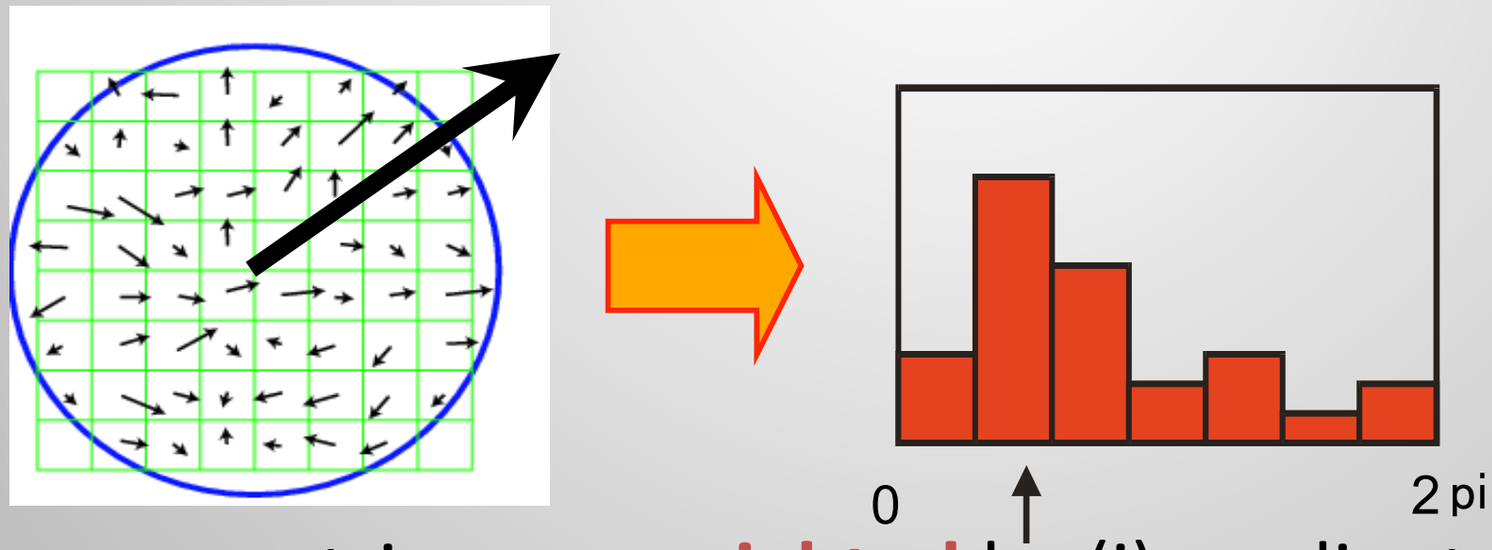
(3) Orientation Assignment

Create histogram of gradient directions, within a region around the keypoint, at selected scale:

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

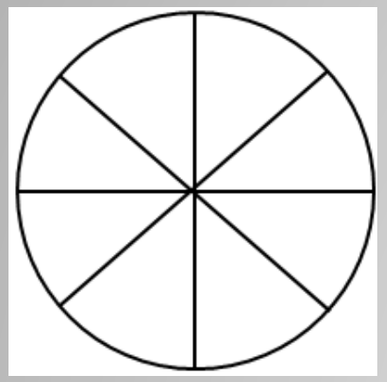
$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

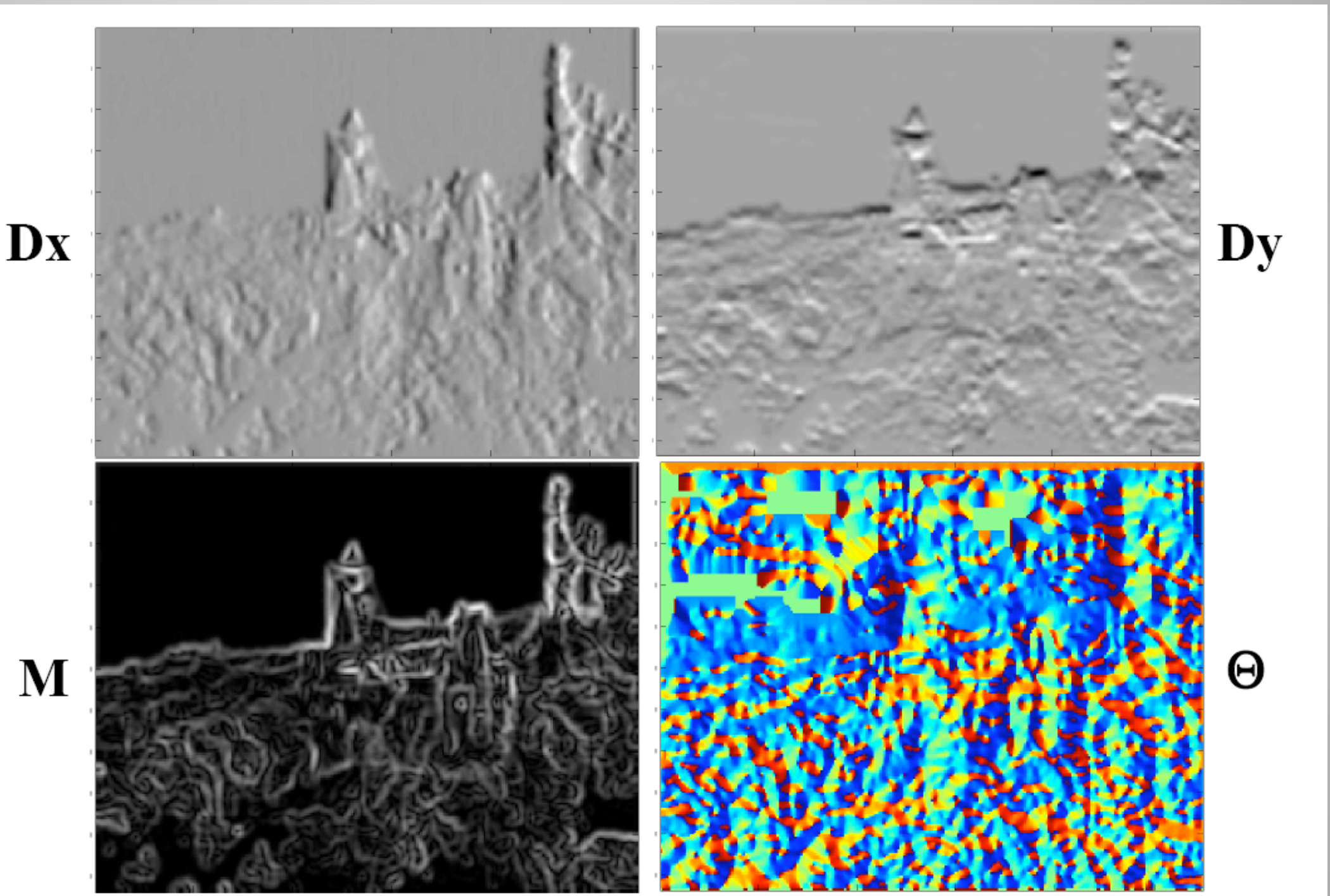
$$\theta(x,y) = a\tan 2((L(x,y+1) - L(x,y-1)) / (L(x+1,y) - L(x-1,y)))$$



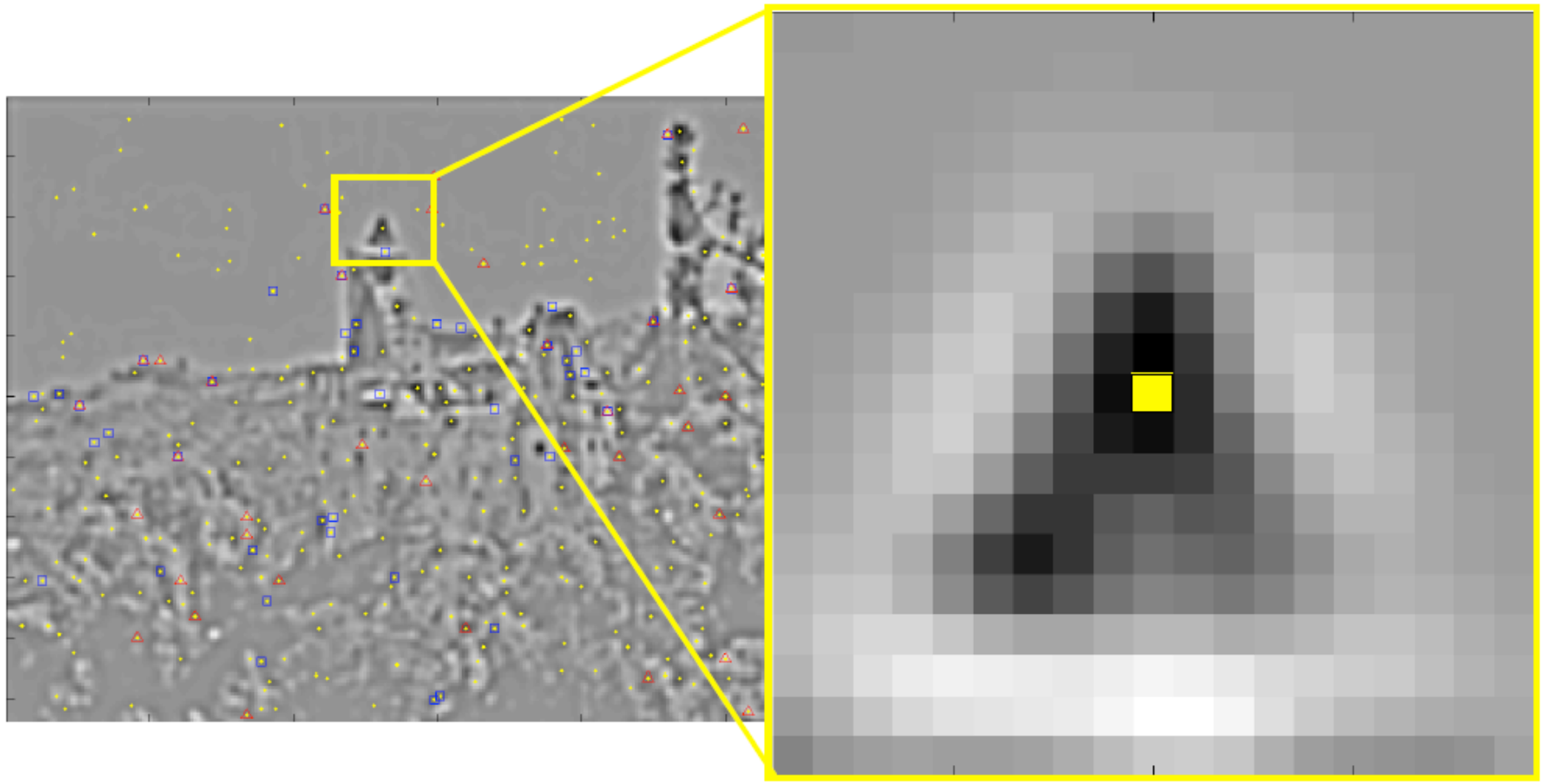
- Histogram entries are **weighted** by (i) gradient magnitude and (ii) a Gaussian function with σ equal to 1.5 times the scale of the keypoint.

(3) Orientation Assignment



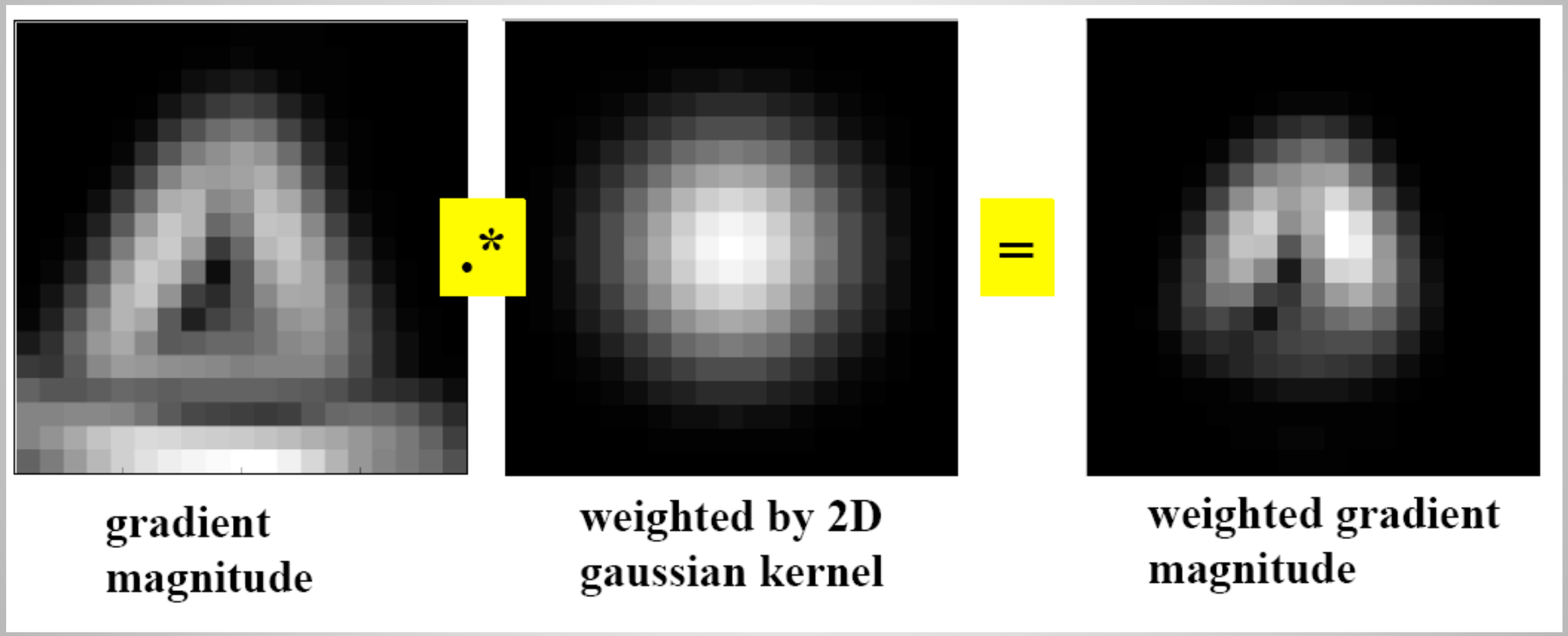


(3) Orientation Assignment



- **Keypoint location = extrema location**
- **Keypoint scale is scale of the DOG image**

(3) Orientation Assignment



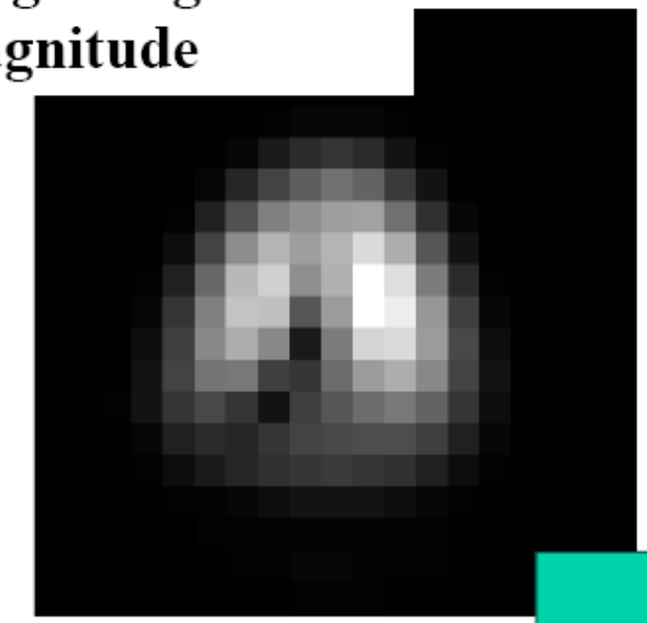
gradient magnitude

weighted by 2D gaussian kernel

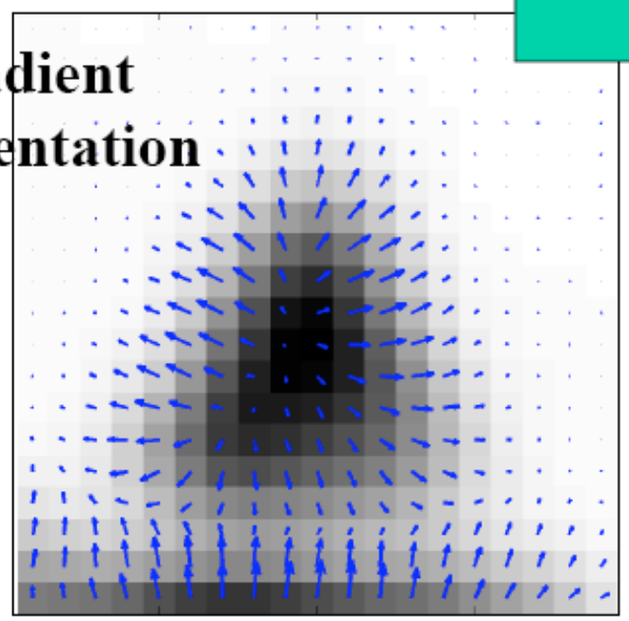
weighted gradient magnitude

(3) Orientation Assignment

weighted gradient magnitude

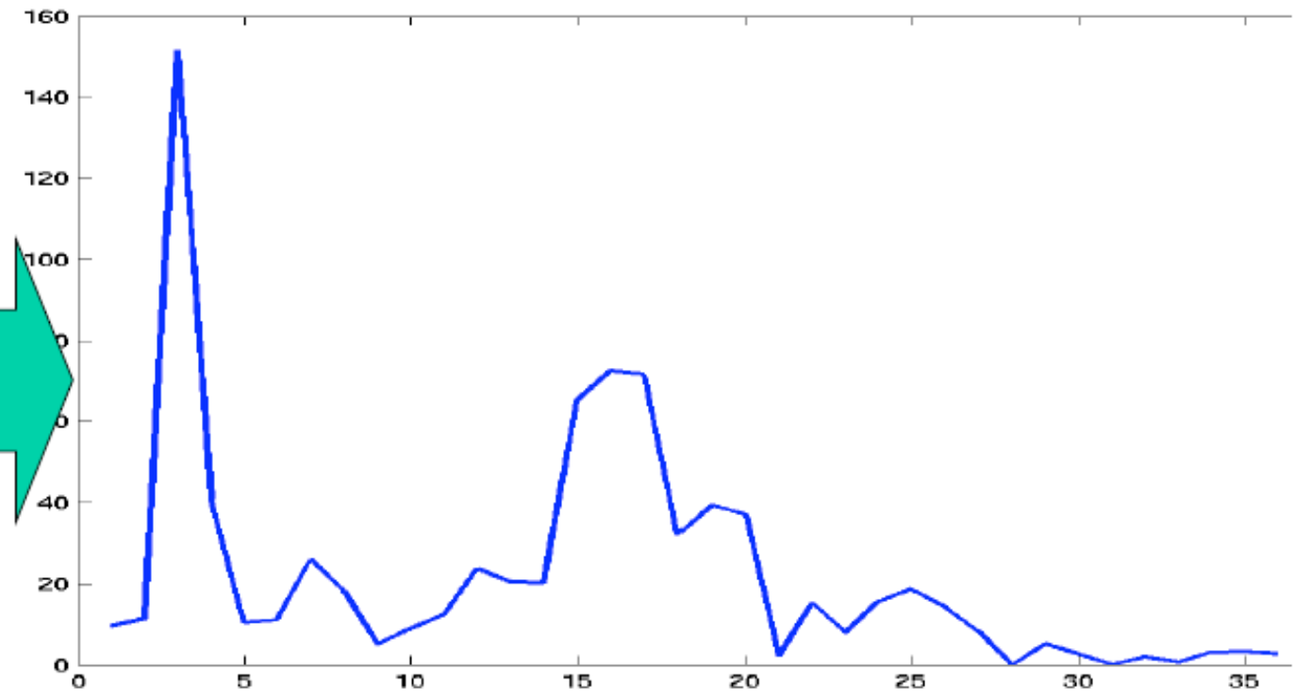


gradient orientation



weighted orientation histogram.

Each bucket contains sum of weighted gradient magnitudes corresponding to angles that fall within that bucket.



36 buckets

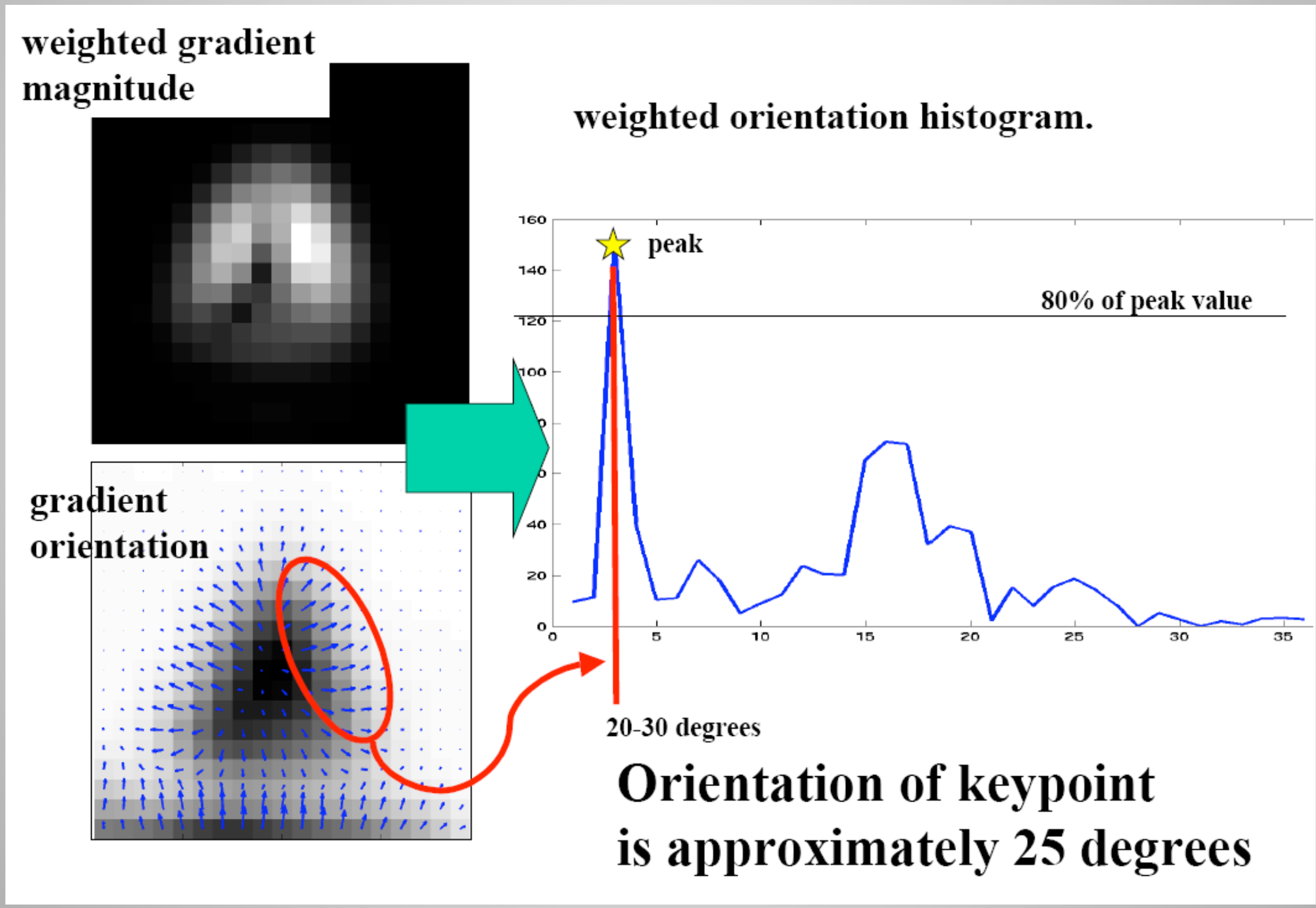
10 degree range of angles in each bucket, i.e.

0 <=ang<10 : bucket 1

10<=ang<20 : bucket 2

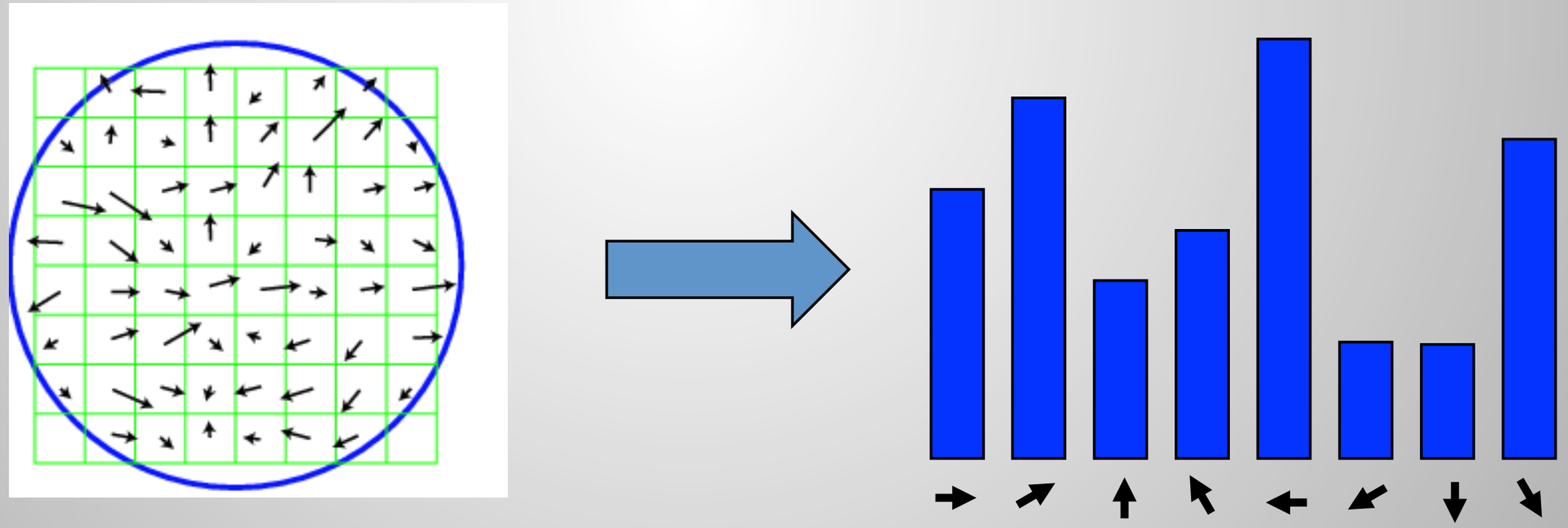
20<=ang<30 : bucket 3 …

(3) Orientation Assignment



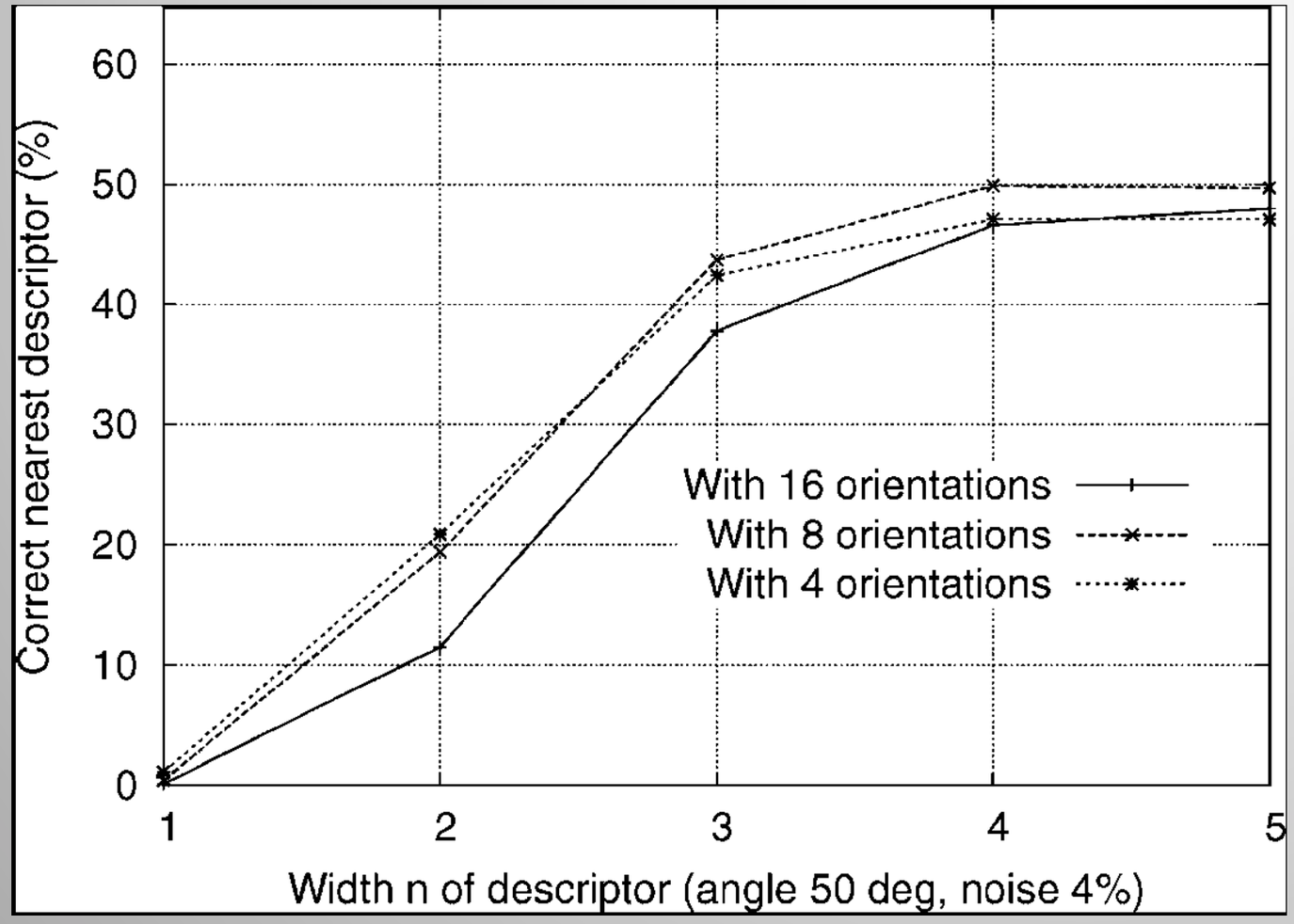
(4) Keypoint Descriptor

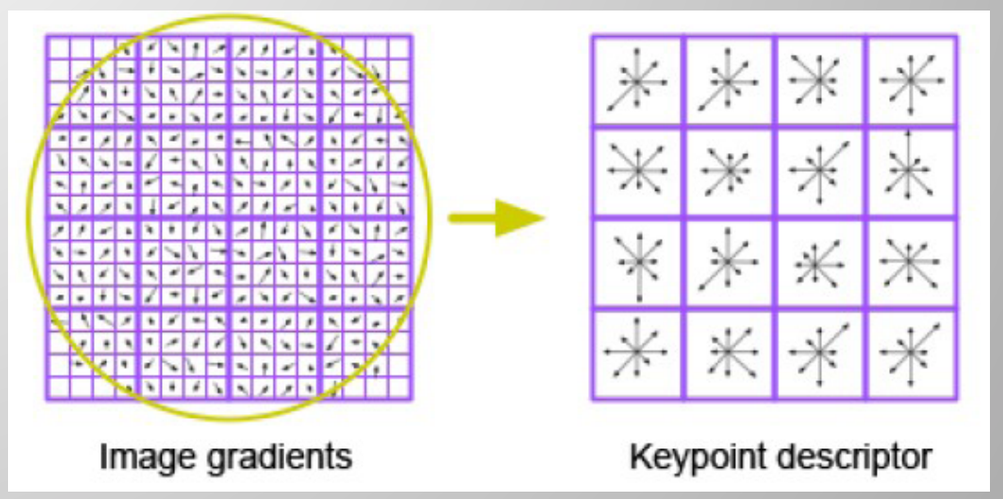
- Partial Voting: distribute histogram entries into adjacent bins (i.e., **additional robustness to shifts)**
  - Each entry is added to all bins, multiplied by a weight of **1-d,** where **d** is the distance from the bin it belongs.

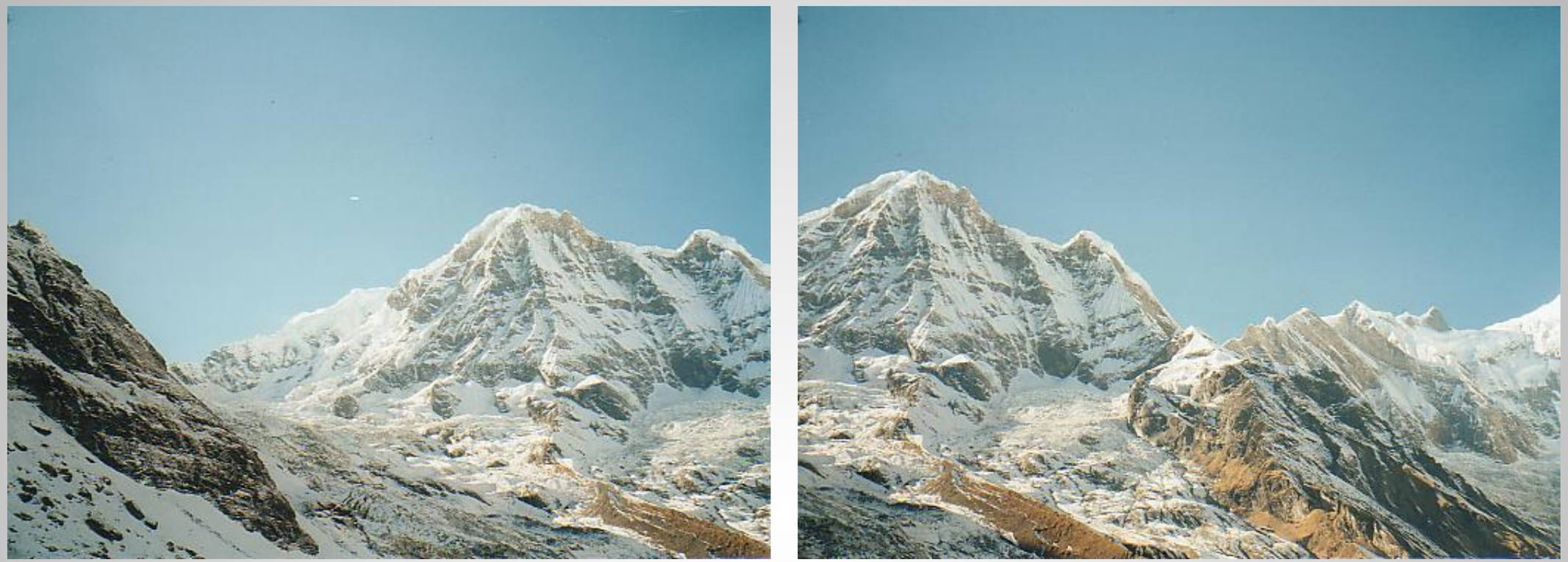


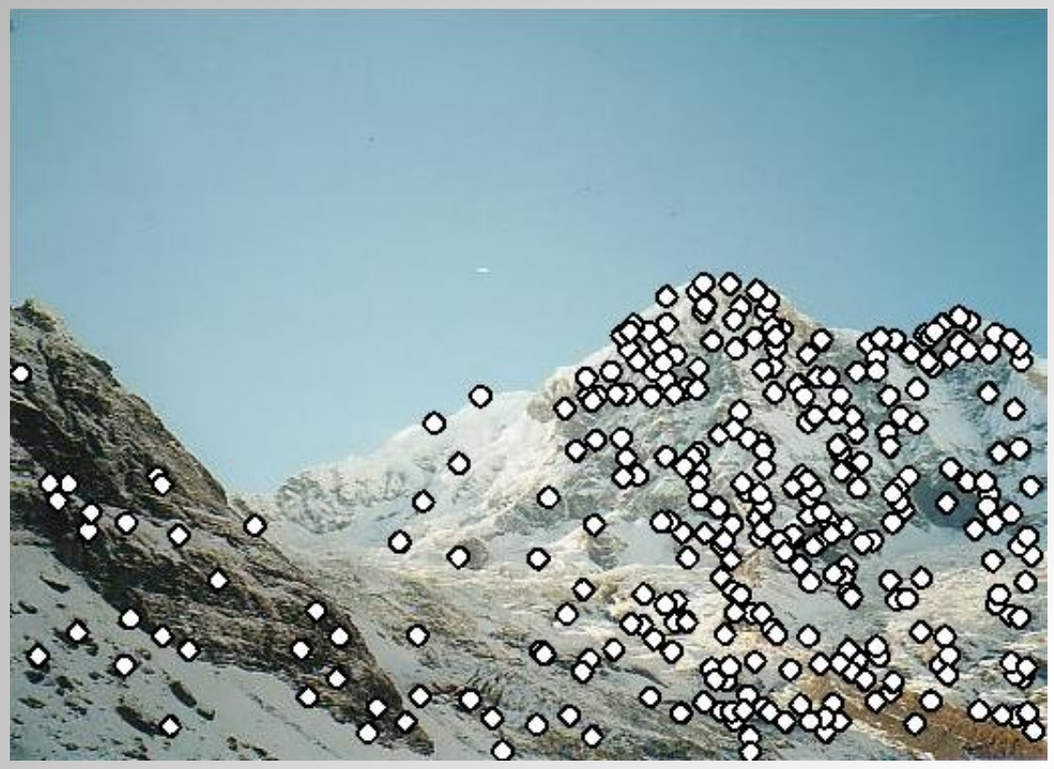
(4) Keypoint Descriptor

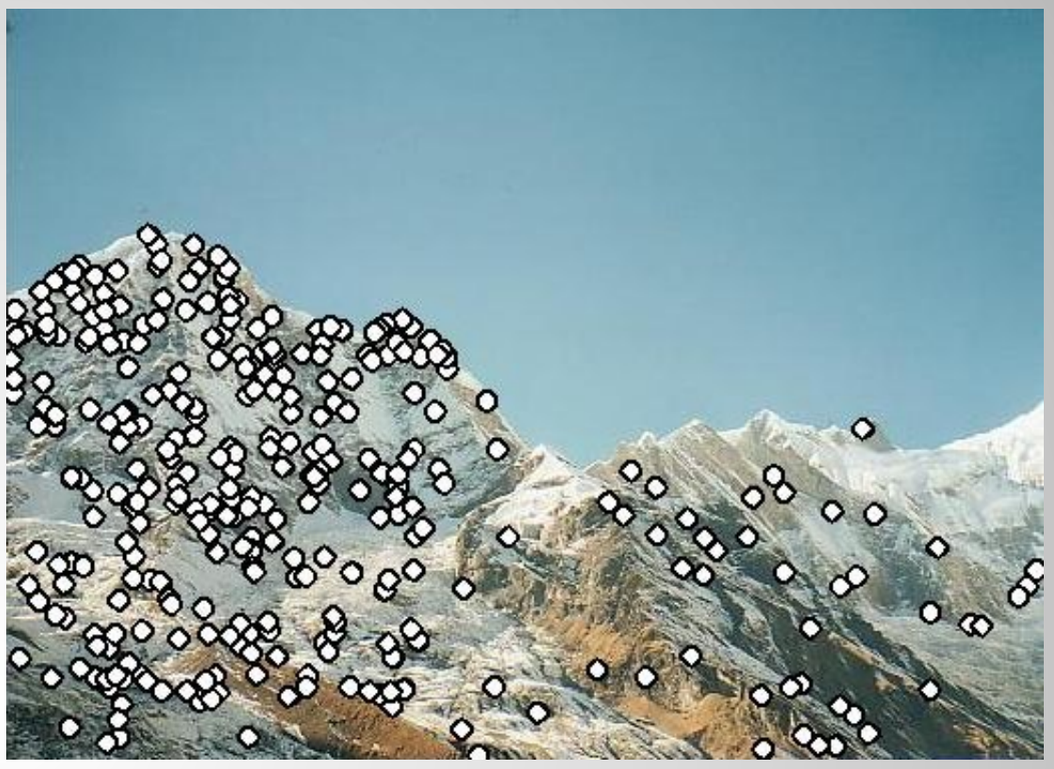
- Descriptor depends on two main parameters:
  - (1) number of orientations
  - n x n array of orientation histograms



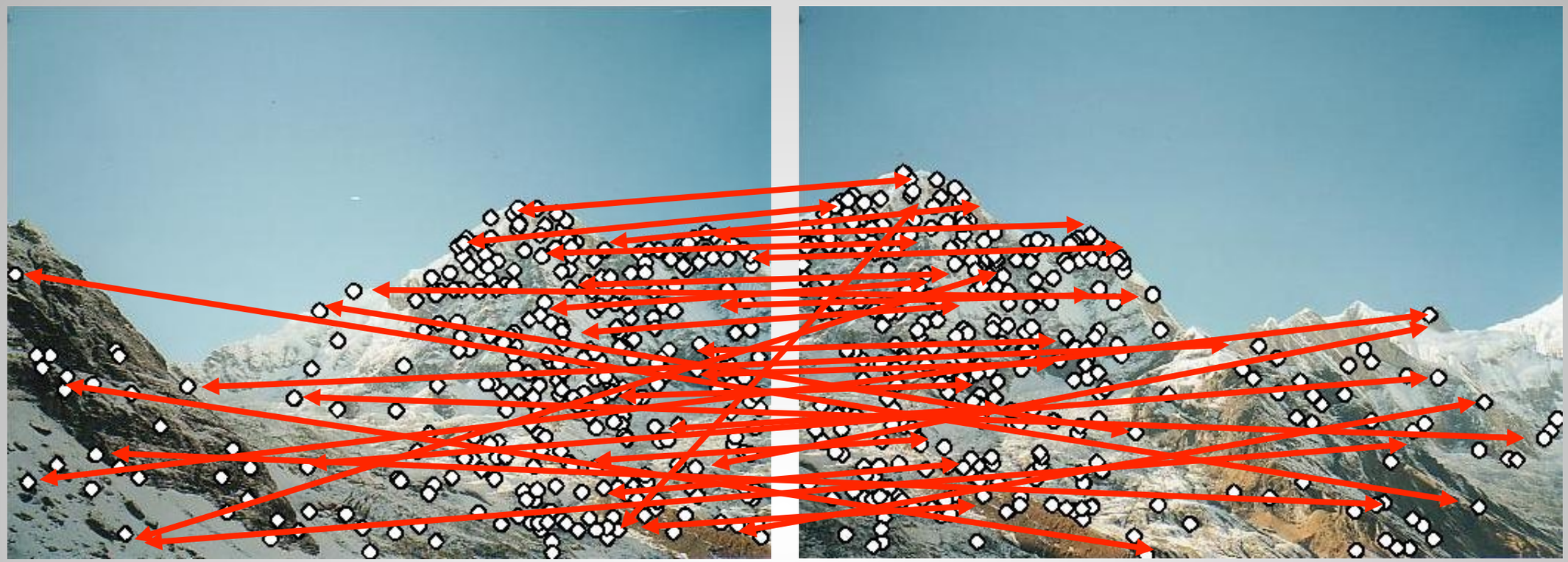




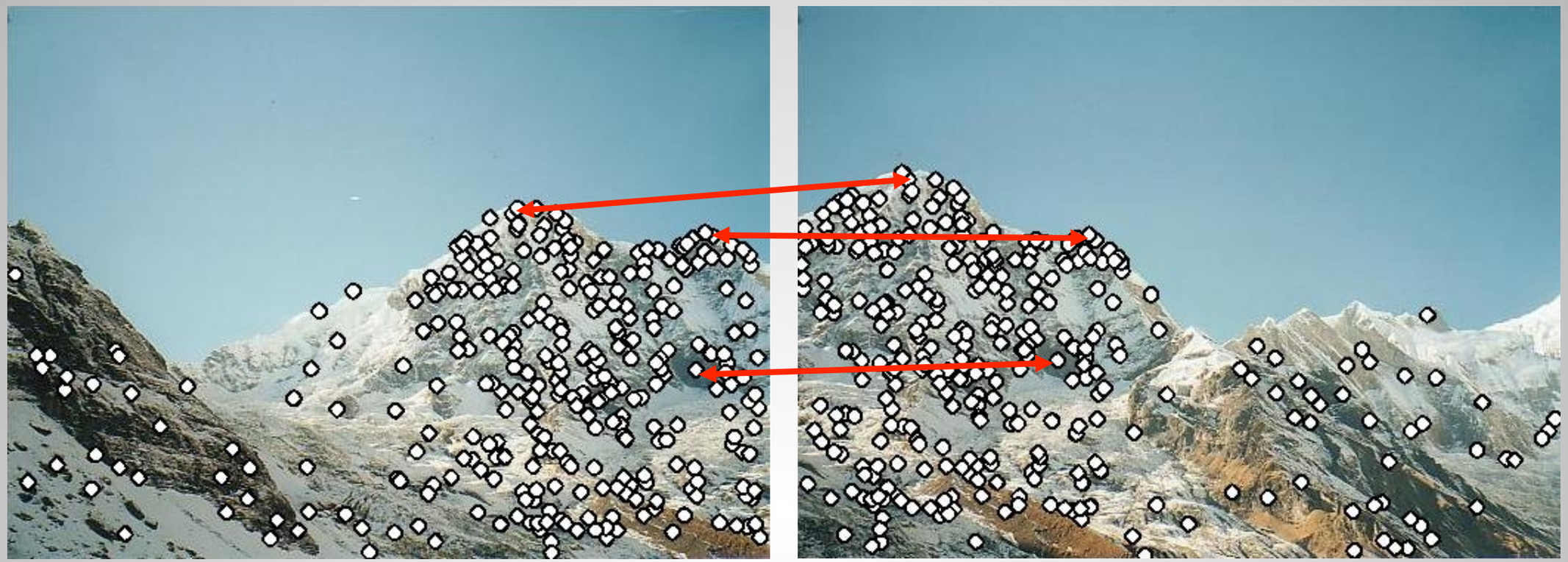




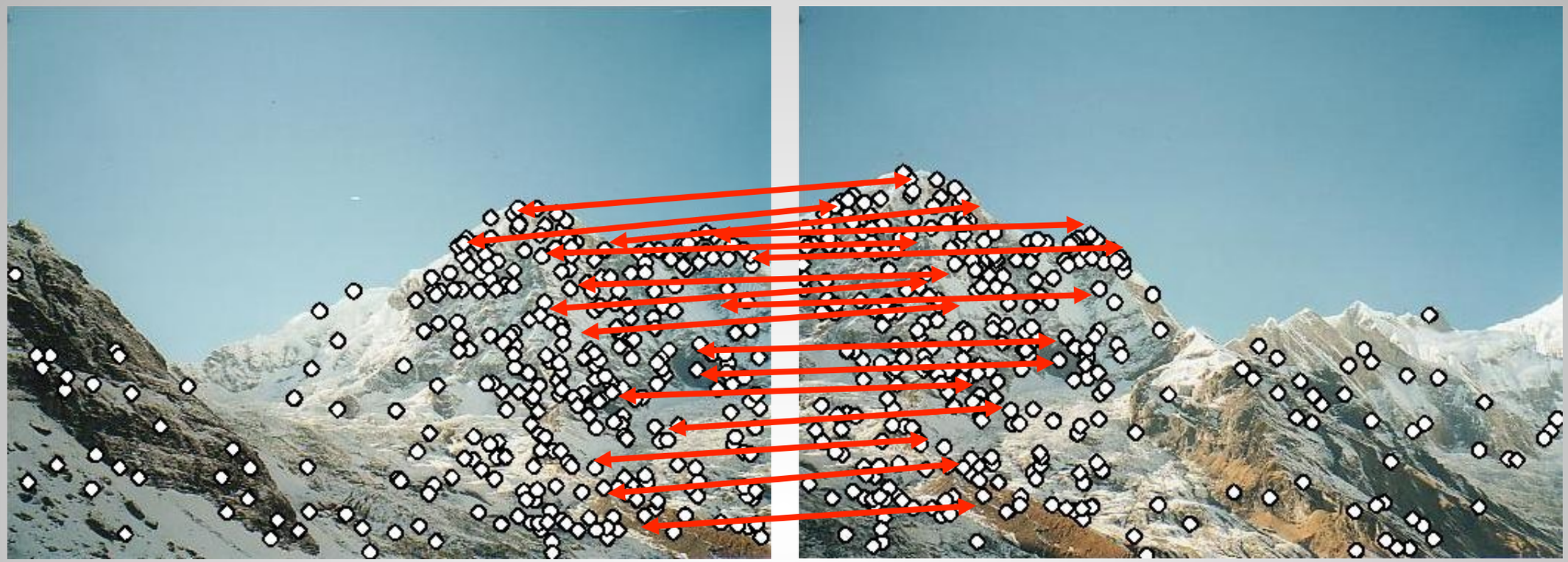
- Extract features



- Extract features
- Compute *putative matches*



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)

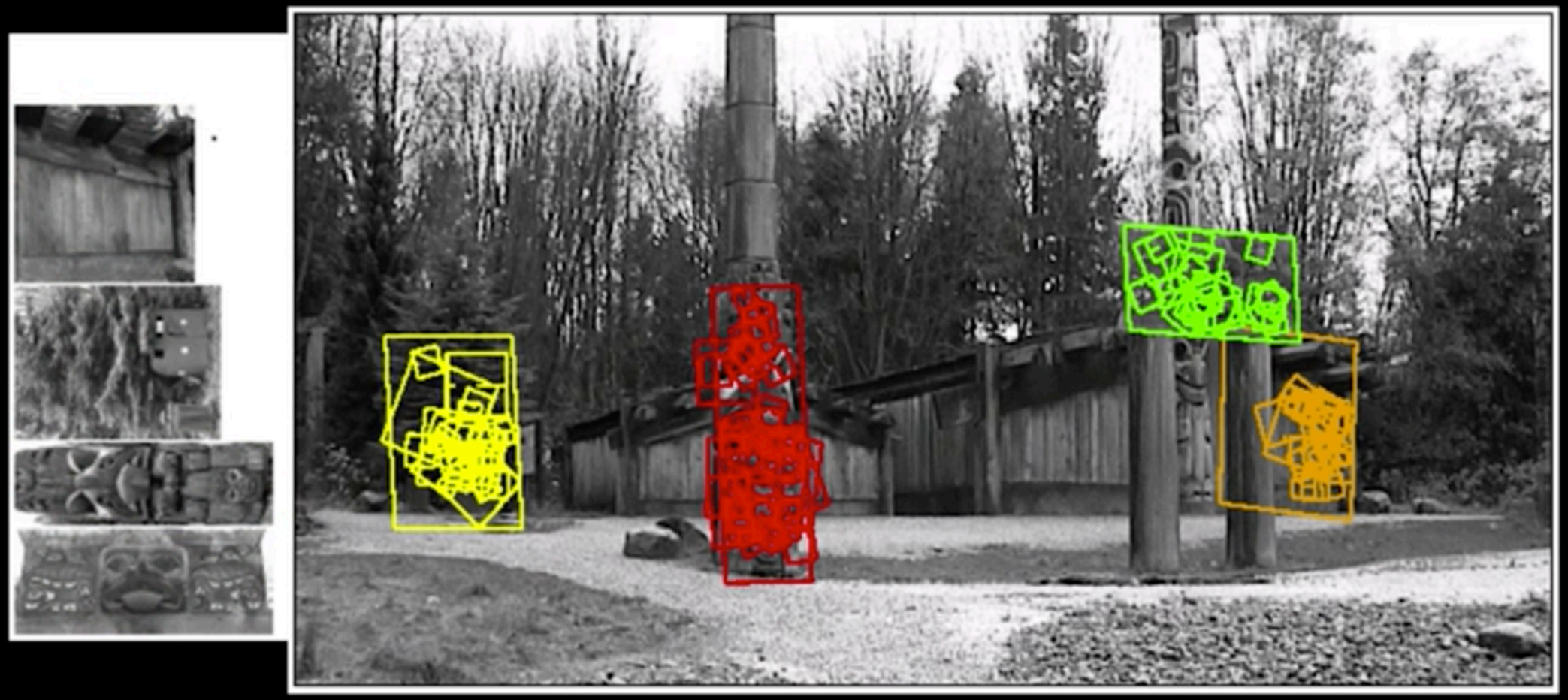


- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
  - *Verify* transformation (search for other matches consistent with *T*)

- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
  - *Verify* transformation (search for other matches consistent with *T*)

Object Recognition

- **For training images:**
  - Extracting keypoints by SIFT.
  - Creating descriptors database.
- **For query images:**
  - Extracting keypoints by SIFT.
  - For each descriptor - finding nearest neighbor in DB.
  - Finding cluster of at-least 3 keypoints.
  - Performing detailed geometric fit check for each cluster.



RANSAC

RANSAC – line fitting example

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using sample
3. **Score** by the fraction of ***inliers*** within a preset threshold of the model
4. **Repeat** 1-3 until the best model is found with high confidence

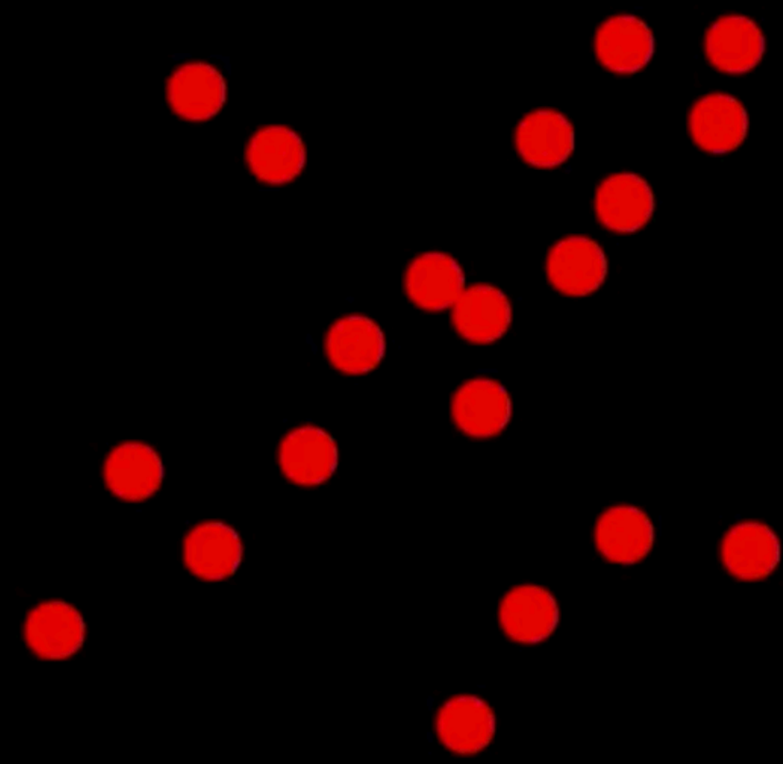
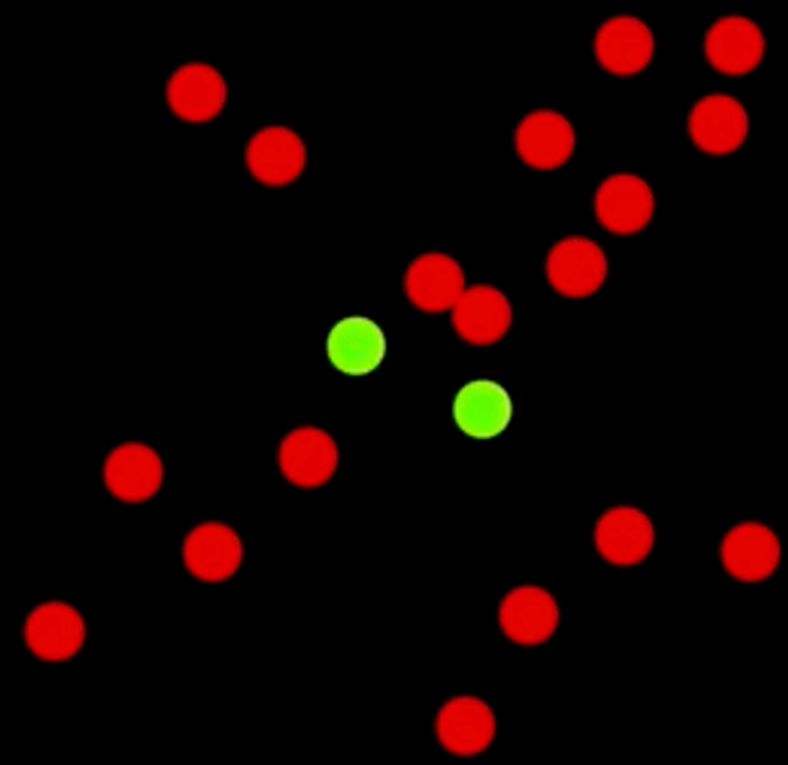


Illustration by Savarese

RANSAC

Algorithm:

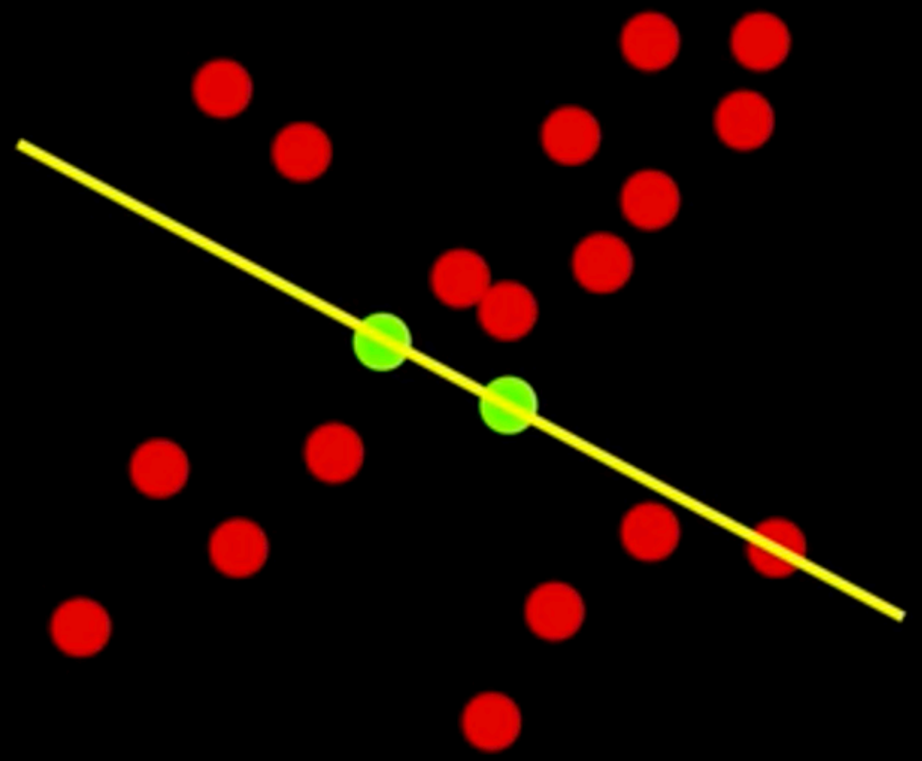
1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using sample
3. **Score** by the fraction of ***inliers*** within a preset threshold of the model
4. **Repeat** 1-3 until the best model is found with high confidence



RANSAC

Algorithm:

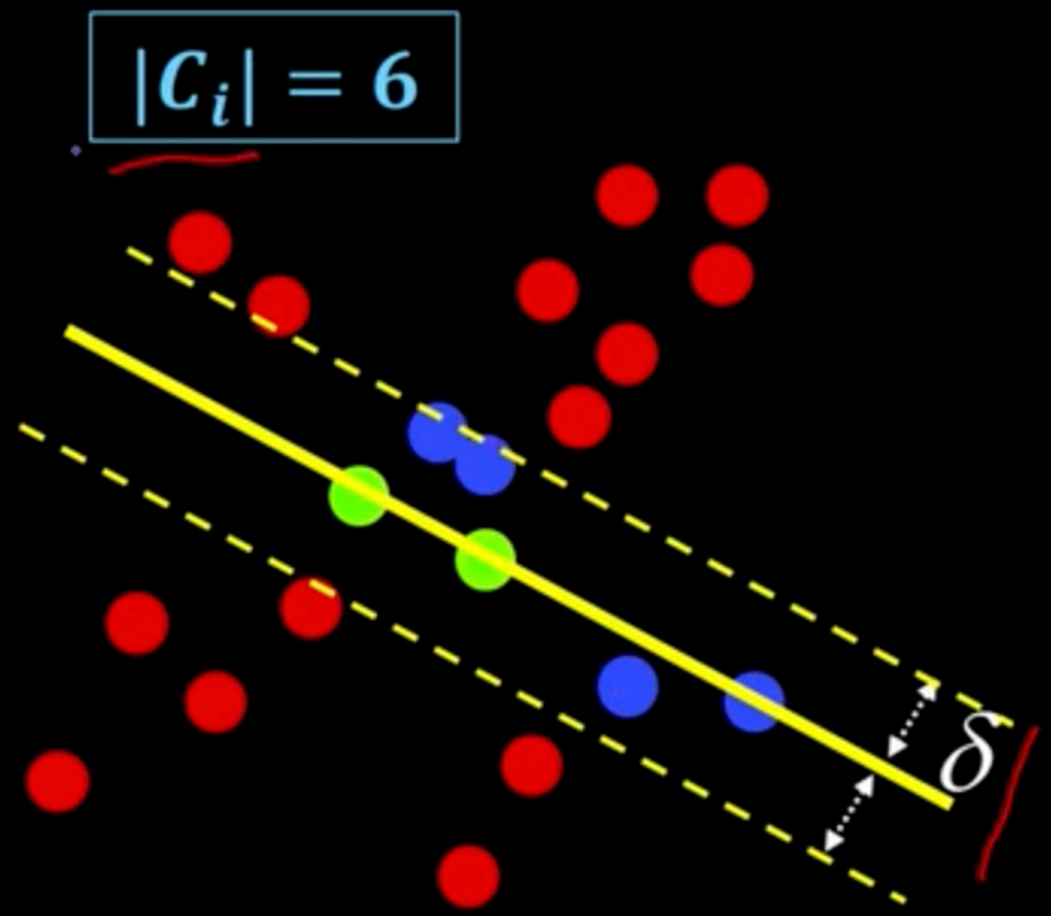
1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using sample
3. **Score** by the fraction of ***inliers*** within a preset threshold of the model
4. **Repeat** 1-3 until the best model is found with high confidence



RANSAC

Algorithm:

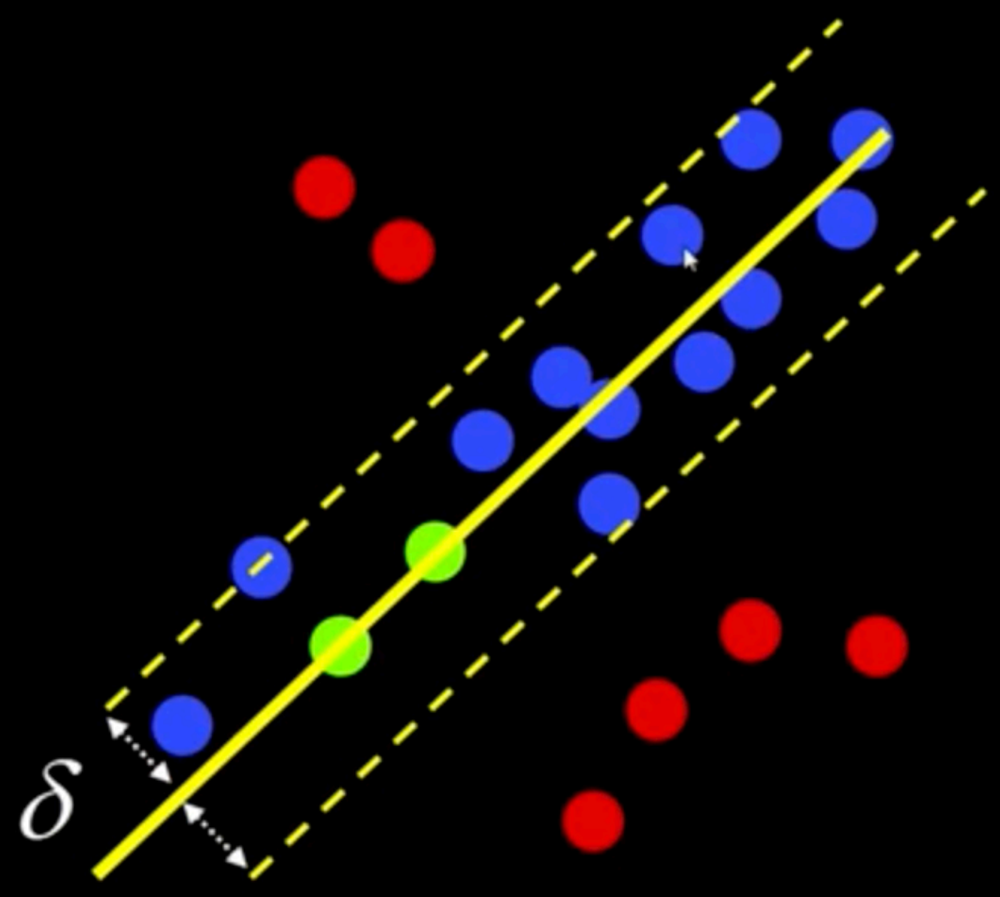
1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using sample
3. **Score** by the fraction of ***inliers*** within a preset threshold of the model
4. **Repeat** 1-3 until the best model is found with high confidence



RANSAC

Algorithm:

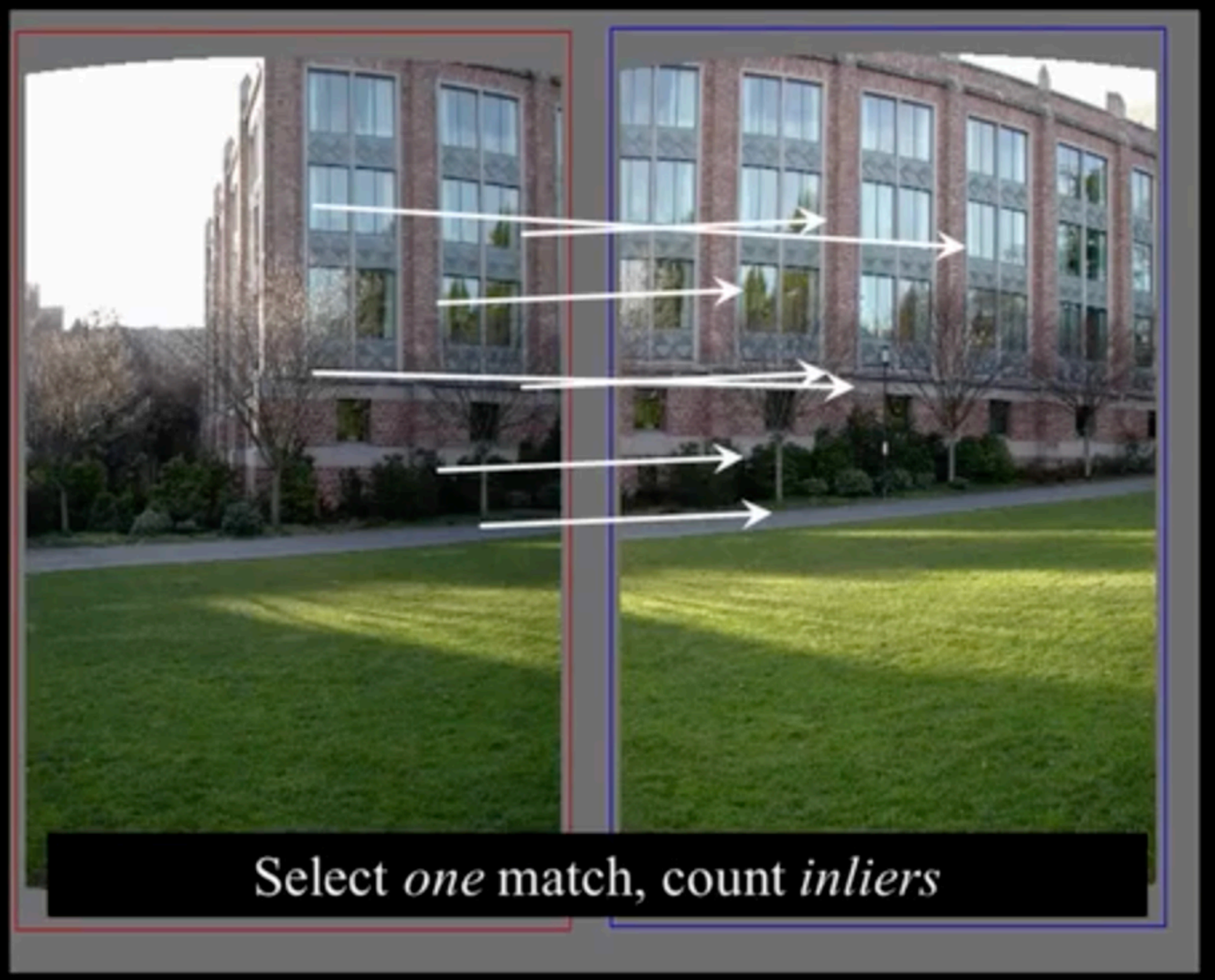
1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using sample
3. **Score** by the fraction of ***inliers*** within a preset threshold of the model
4. **Repeat** 1-3 until the best model is found with high confidence



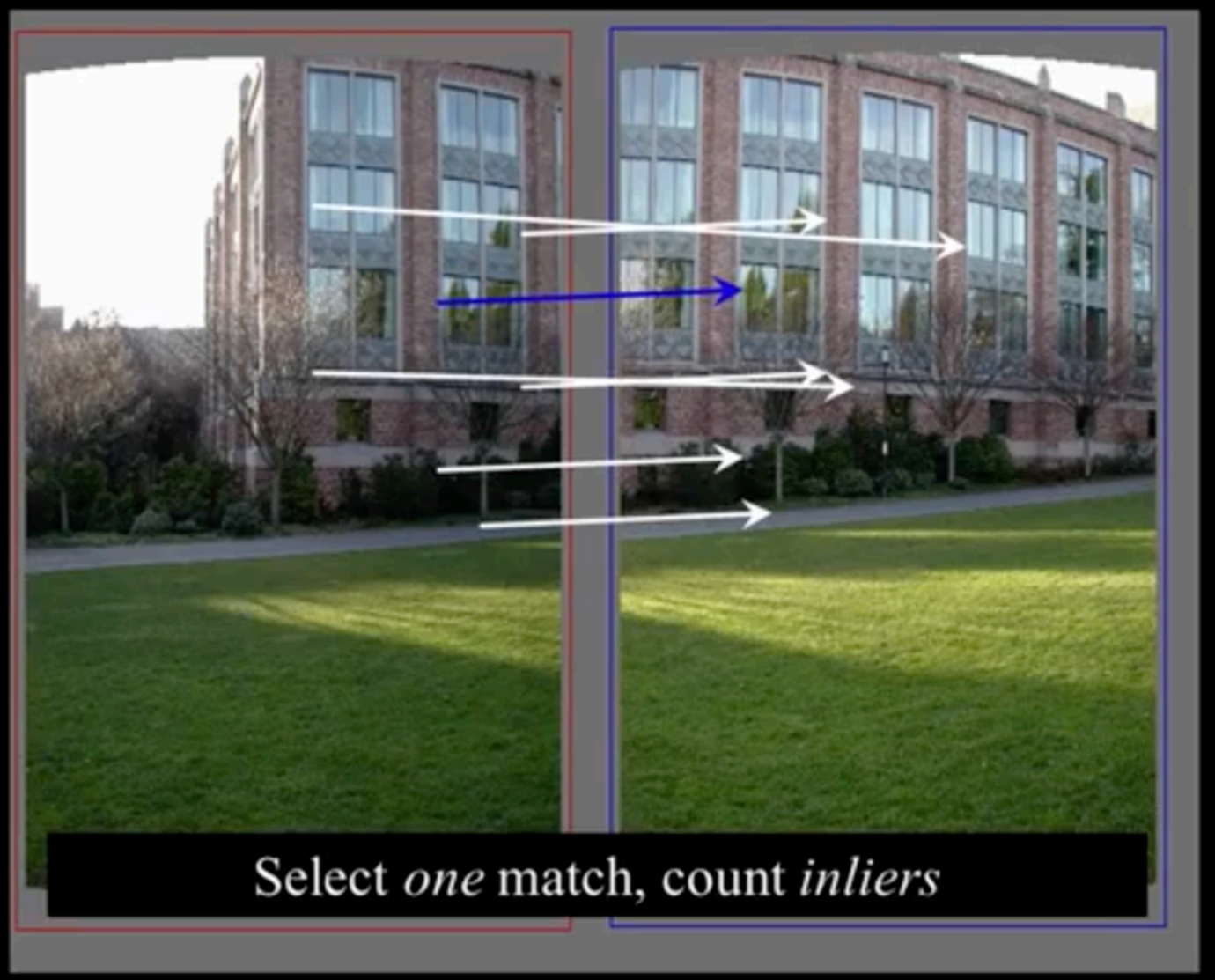
$$|C_i| = 14$$

Matching features

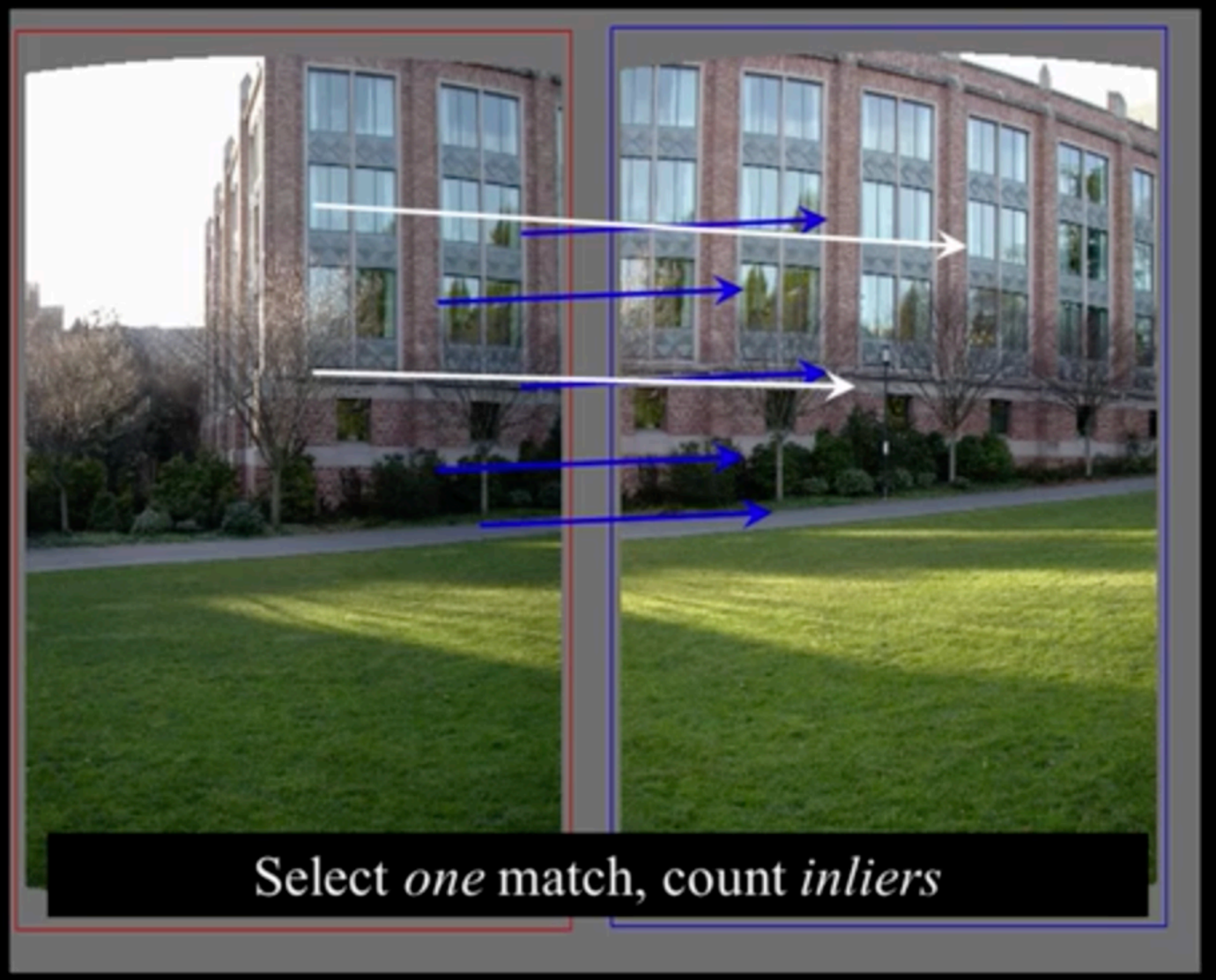
RAndom SAmple Consensus (1)



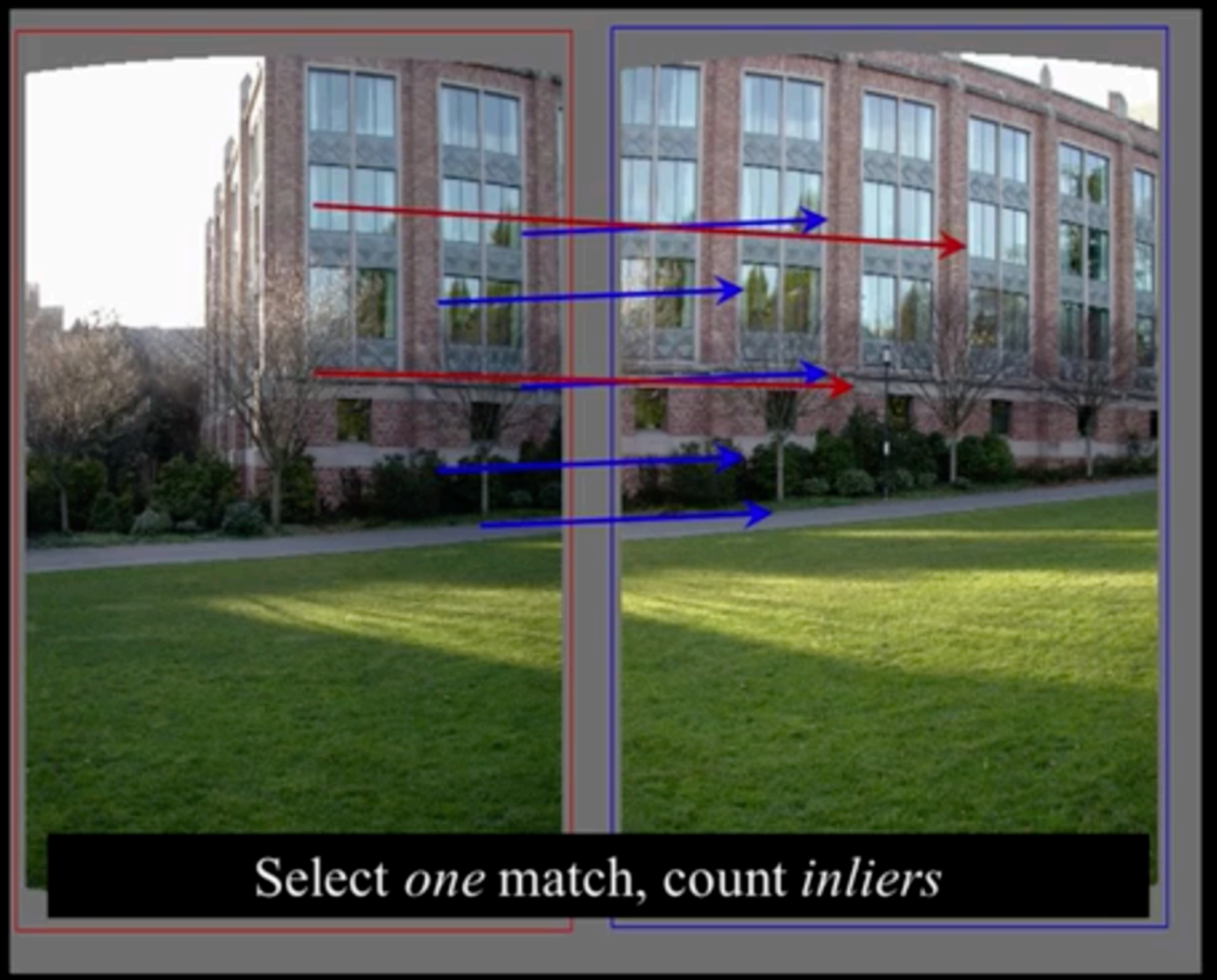
RAndom SAmple Consensus (1)



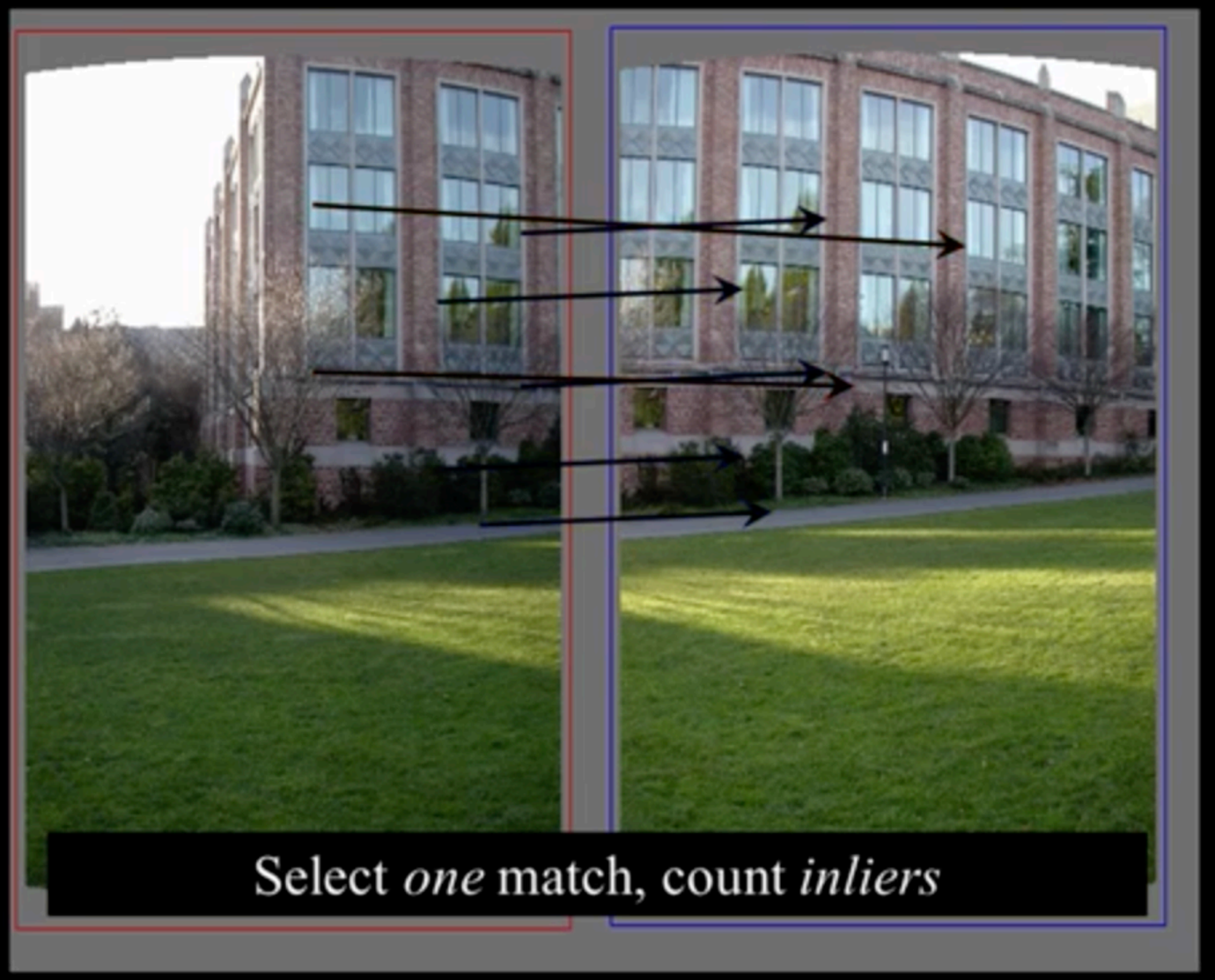
RAndom SAmple Consensus (1)



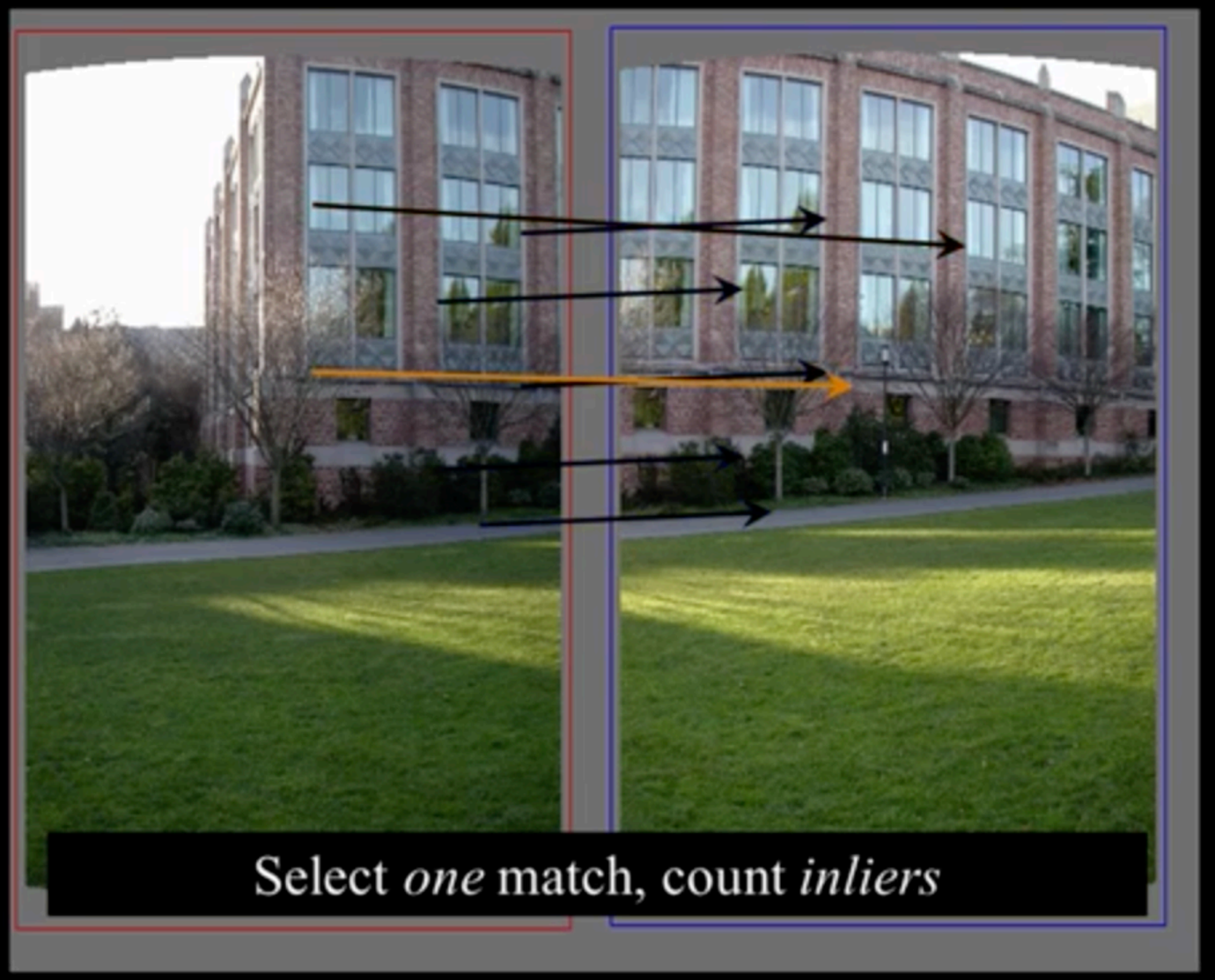
RAndom SAmple Consensus (1)



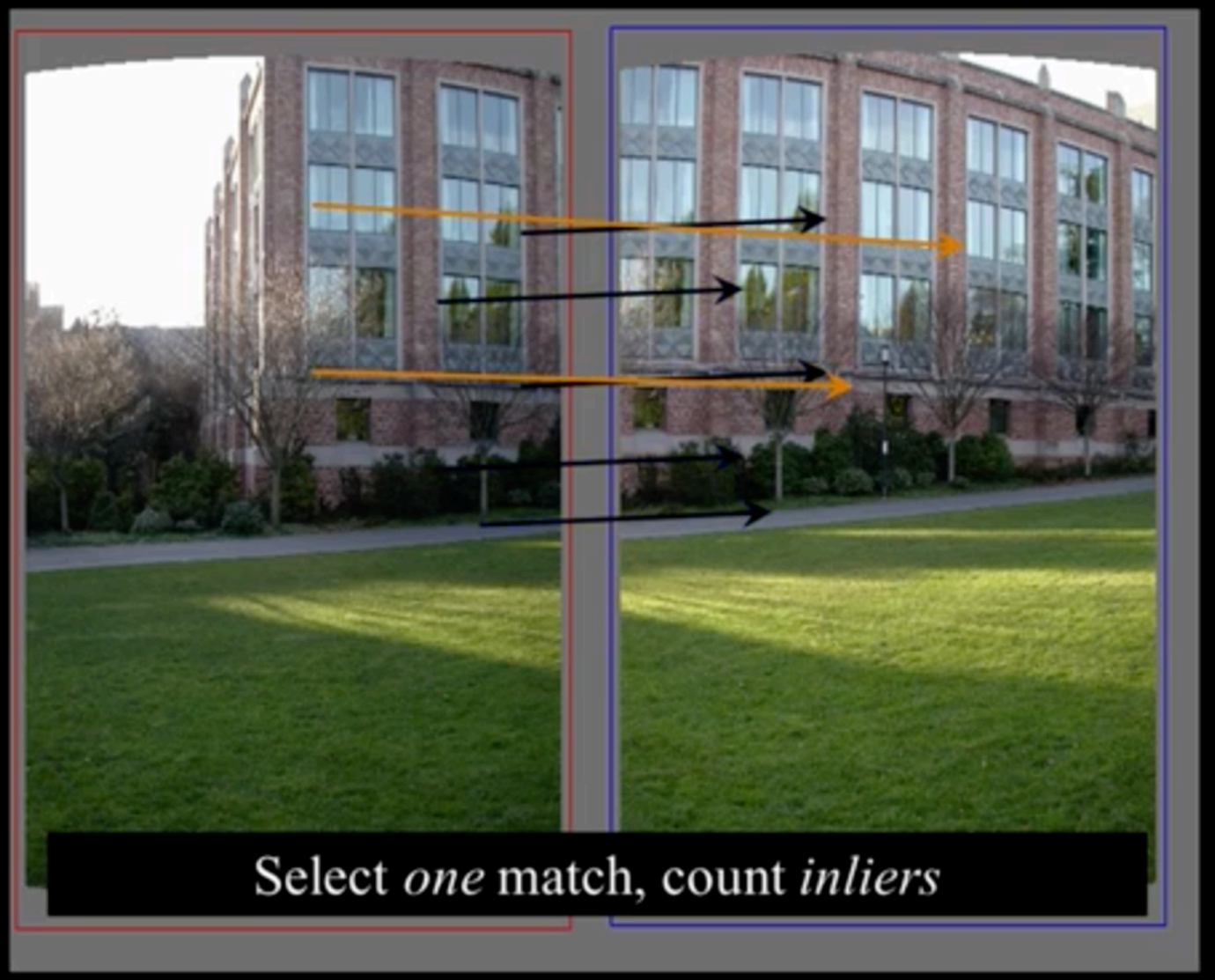
RAndom SAmple Consensus (2)



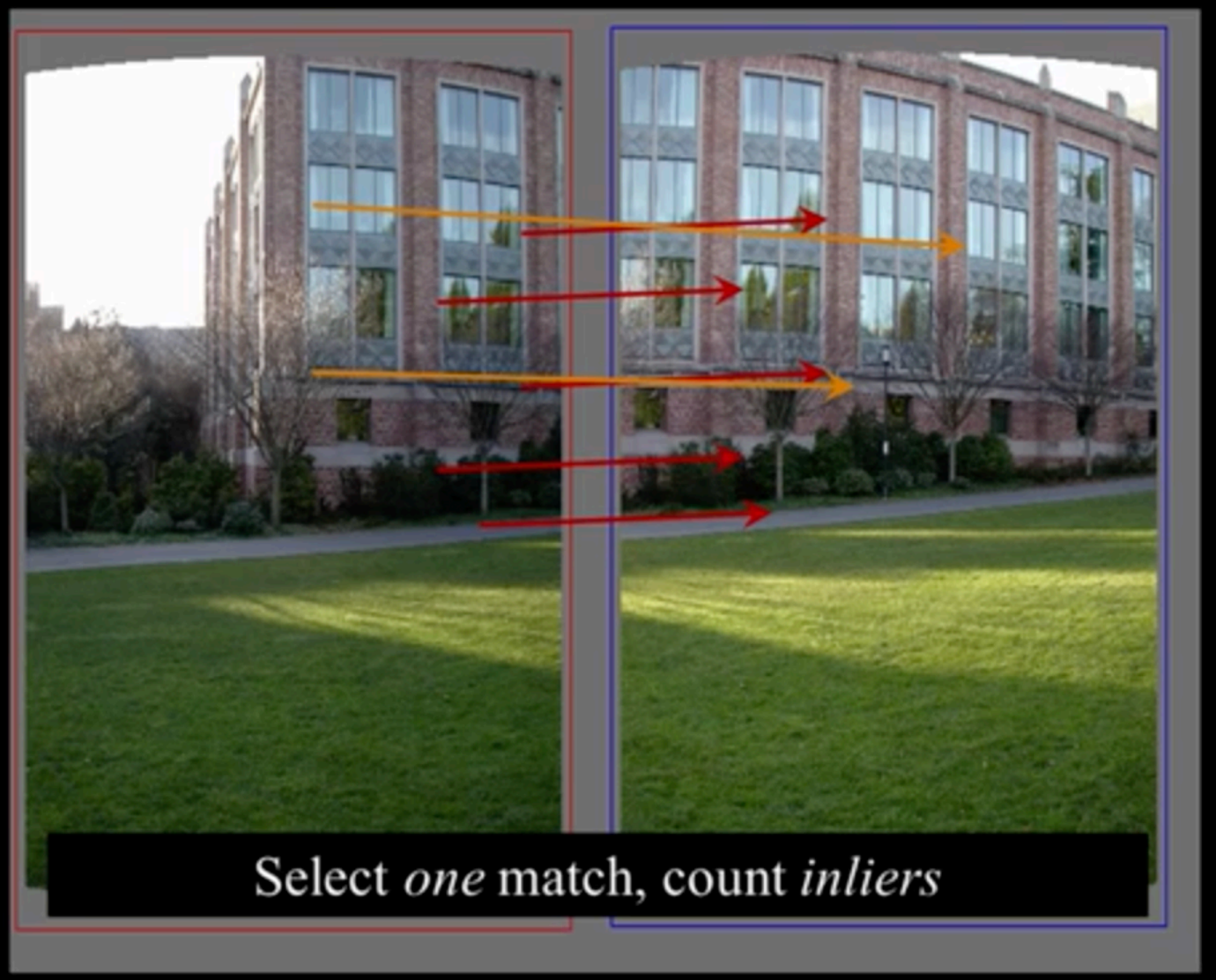
RAndom SAmple Consensus (2)



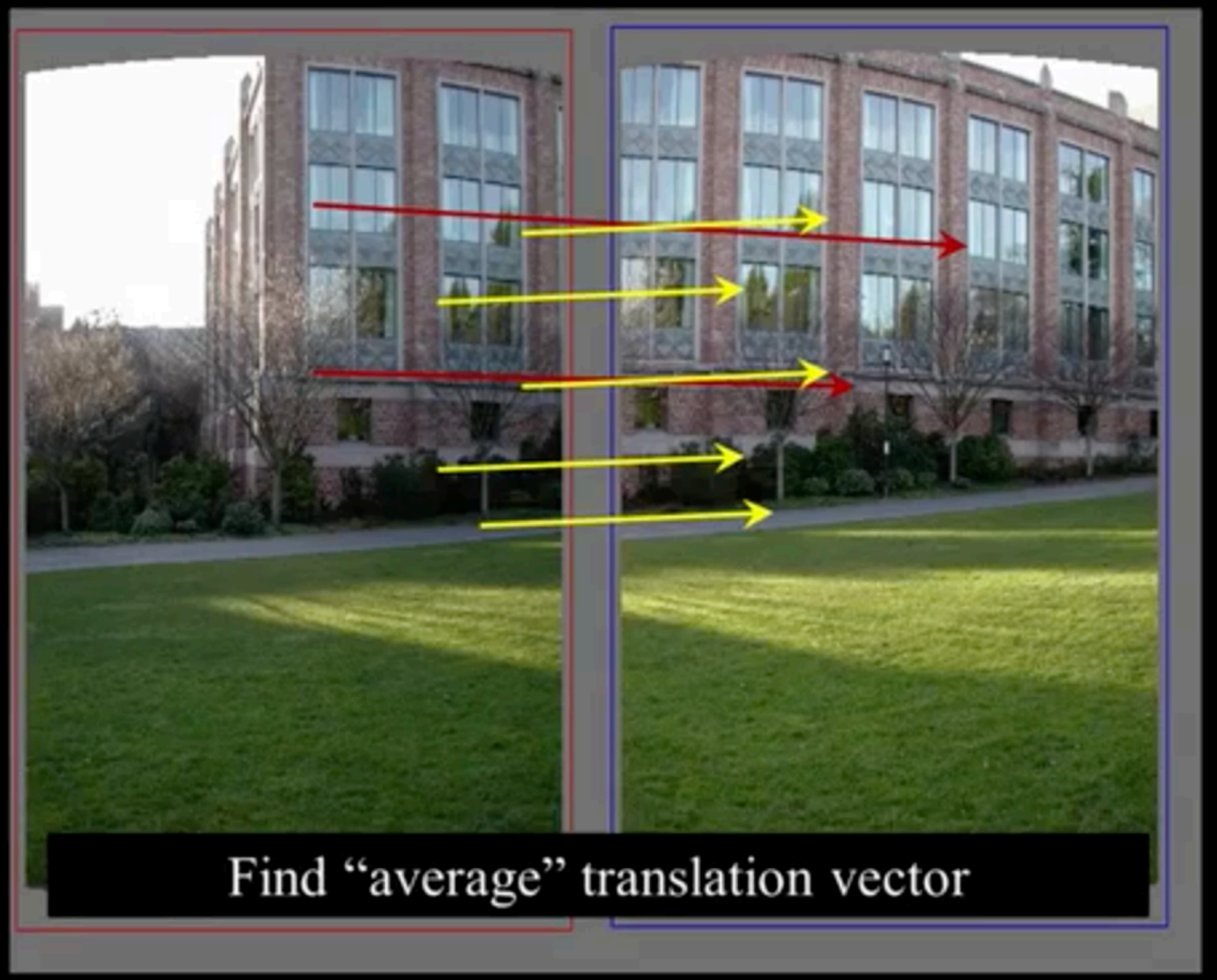
RAndom SAmple Consensus (2)



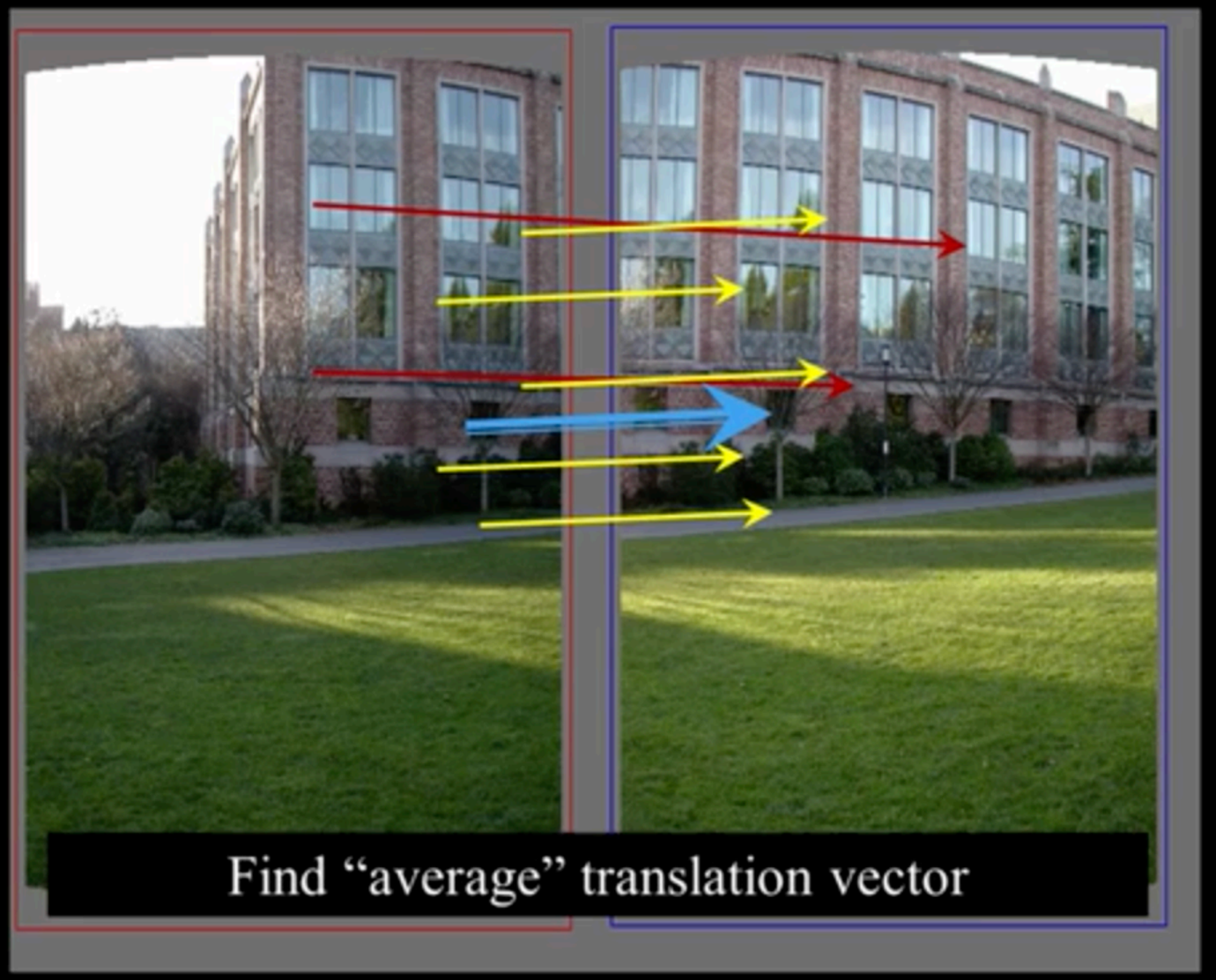
RAndom SAmple Consensus (2)



Least squares fit



Least squares fit



Additional
Key points / features

- Harris, SIFT..
- SURF: Speeded Up Robust Feature
- FAST: Features from Accelerated Segment Test
- BRIEF: Binary Robust Independent Elementary Features
- ORB: Oriented FAST and Rotated BRIEF

Next time:

Feature Engineering 2: **Regions**

Viola Jones, **HOG**, DPM