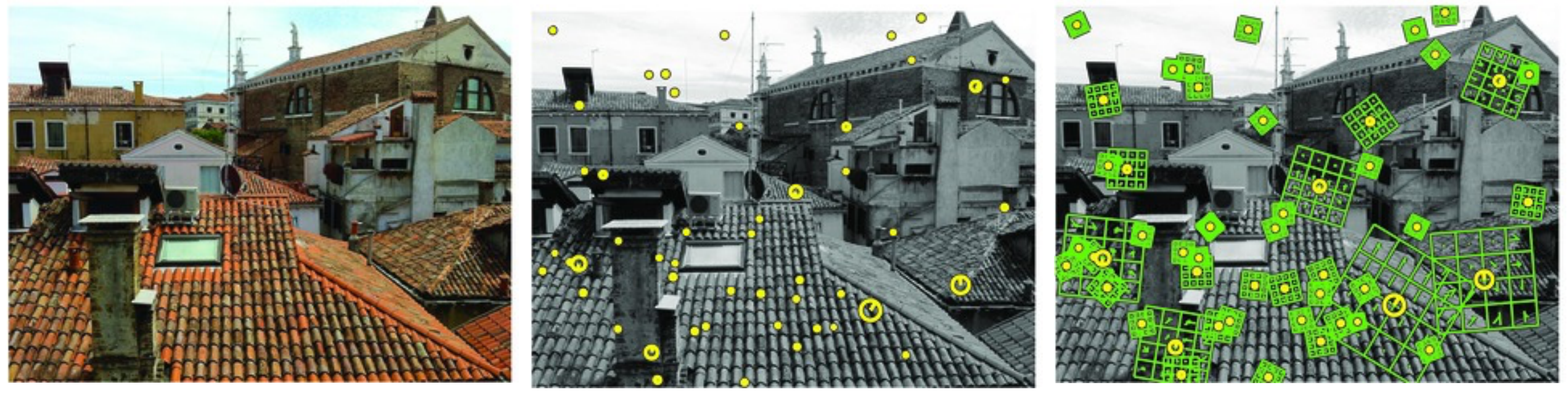
THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision



Lecture 13: Correspondence and SIFT

Menu for Today

Topics:

- **Correspondence** Problem
- **Patch** matching
- **Invariance**, geometric, photometric
- **SIFT** = Scale Invariant Feature Transform

Readings:

- **Today's** Lecture: Szeliski Chapter 7, Forsyth & Ponce 5.4

Reminders:

- **Midterm** — we will be grading over the next week
- **Assignment 3**: Texture Synthesis due on **Monday**
- **Assignment 4**: RANSAC and Panorama Stitching out on Monday

Today's "**fun**" Example: Recognizing Panoramas

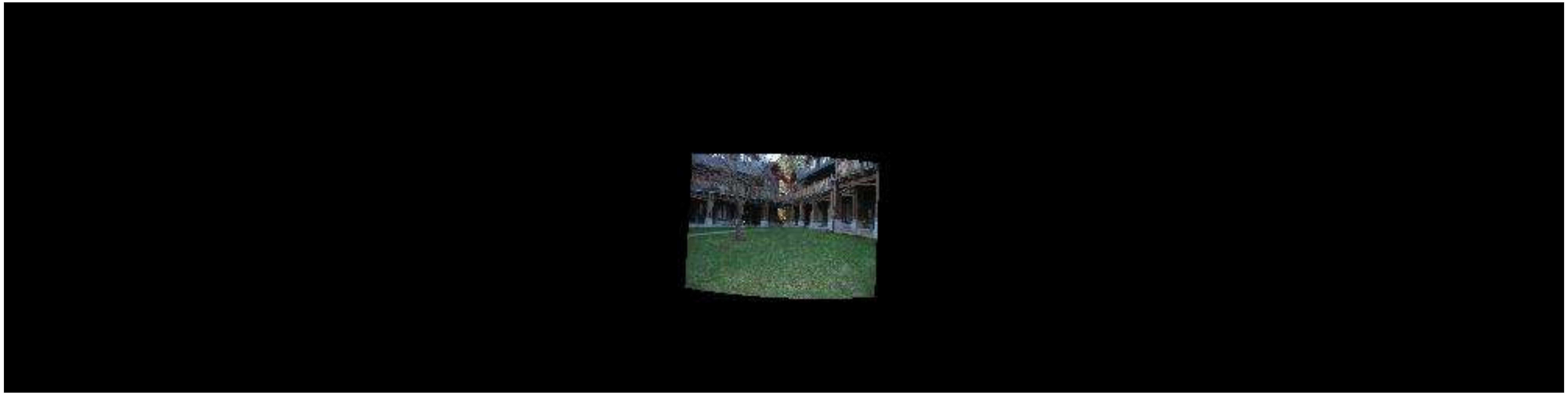


Figure Credit: Matthew Brown and David Lowe

Today's "**fun**" Example: Recognizing Panoramas

Figure Credit: Matthew Brown and David Lowe

Today's "**fun**" Example: Recognizing Panoramas

Figure Credit: Matthew Brown and David Lowe

Today’s “**fun**” Example: Recognizing Panoramas

Figure Credit: Matthew Brown and David Lowe

Today's "**fun**" Example: Recognizing Panoramas

Figure Credit: Matthew Brown and David Lowe

Today's "**fun**" Example: Recognizing Panoramas

Figure Credit: Matthew Brown and David Lowe

Today's "**fun**" Example: Recognizing Panoramas

Figure Credit: Matthew Brown and David Lowe

Today’s “**fun**” Example: Recognizing Panoramas

Figure Credit: Matthew Brown and David Lowe

Today's "**fun**" Example: Recognizing Panoramas

Figure Credit: Matthew Brown and David Lowe

Correspondence Problem

A basic problem in Computer Vision is to establish matches (correspondences) between images.

This has **many** applications: rigid/non-rigid tracking, object recognition, image registration, structure from motion, stereo...

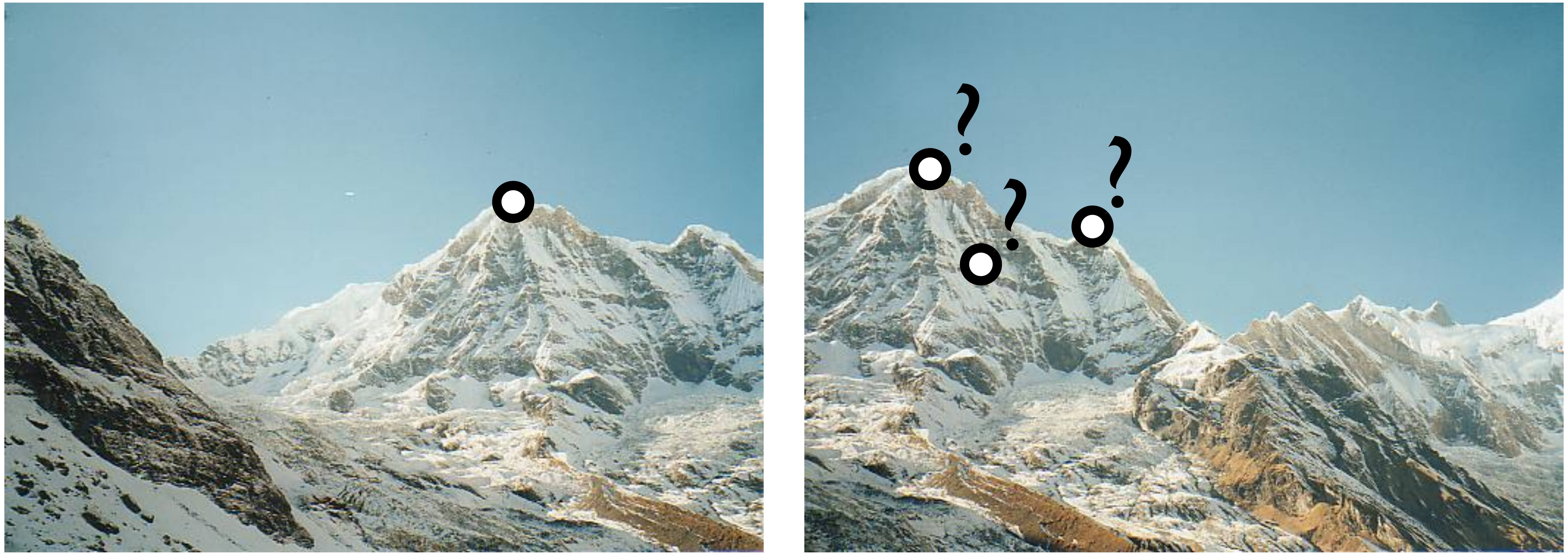


Image **Panoramas**

Building Rome in a Day

The Colosseum: 2,106 images, 819,242 points matched

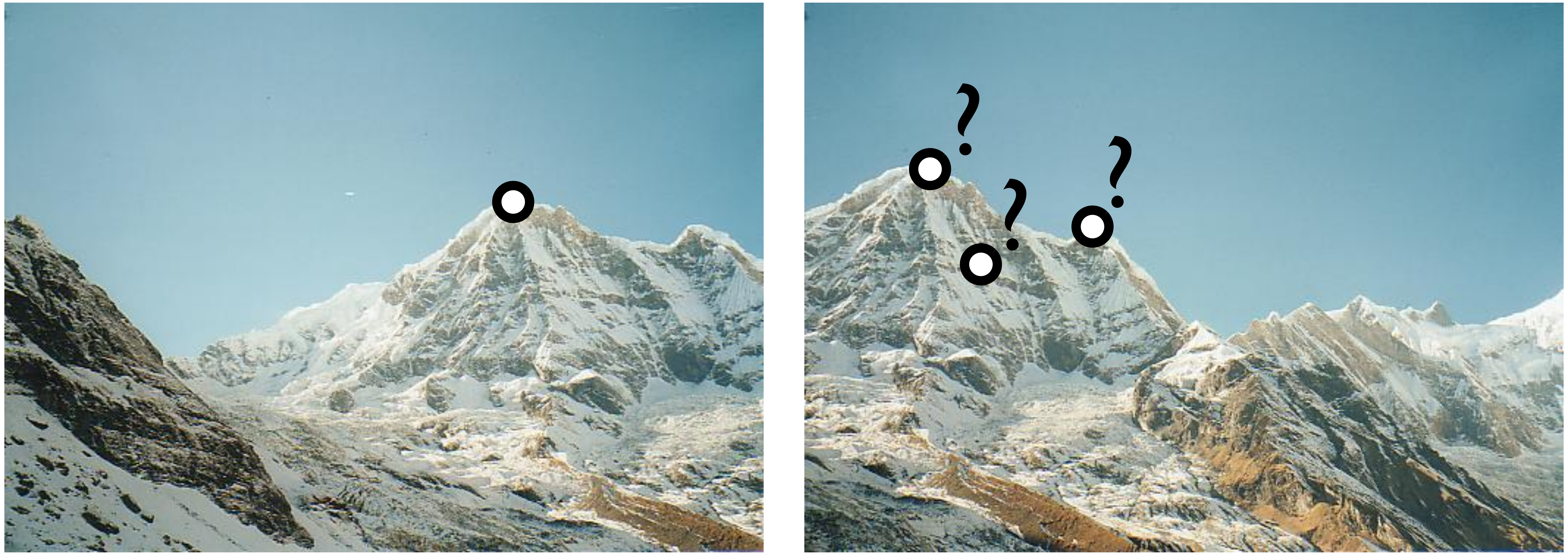
Building Rome in a Day

The Colosseum: 2,106 images, 819,242 points matched

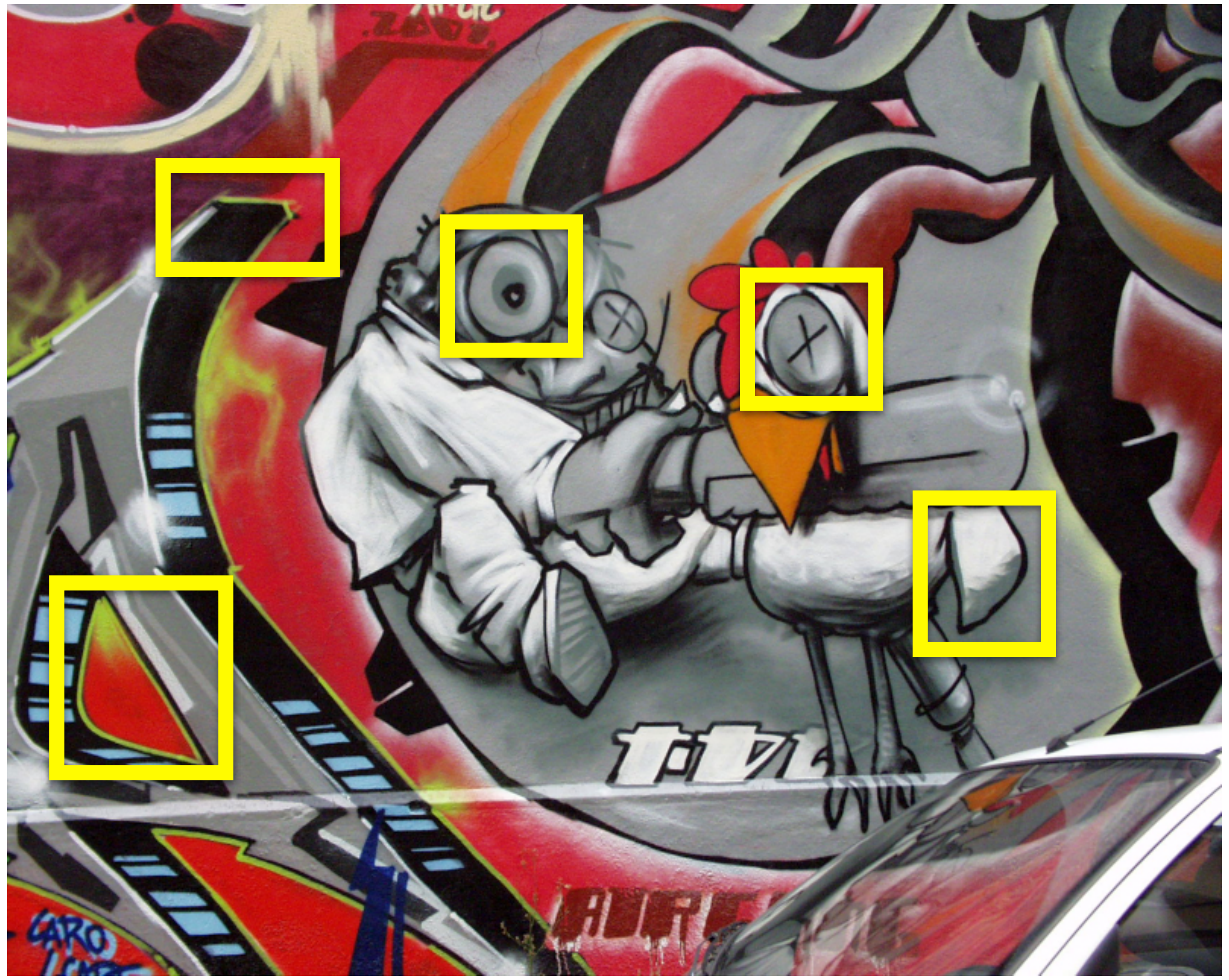
Correspondence Problem

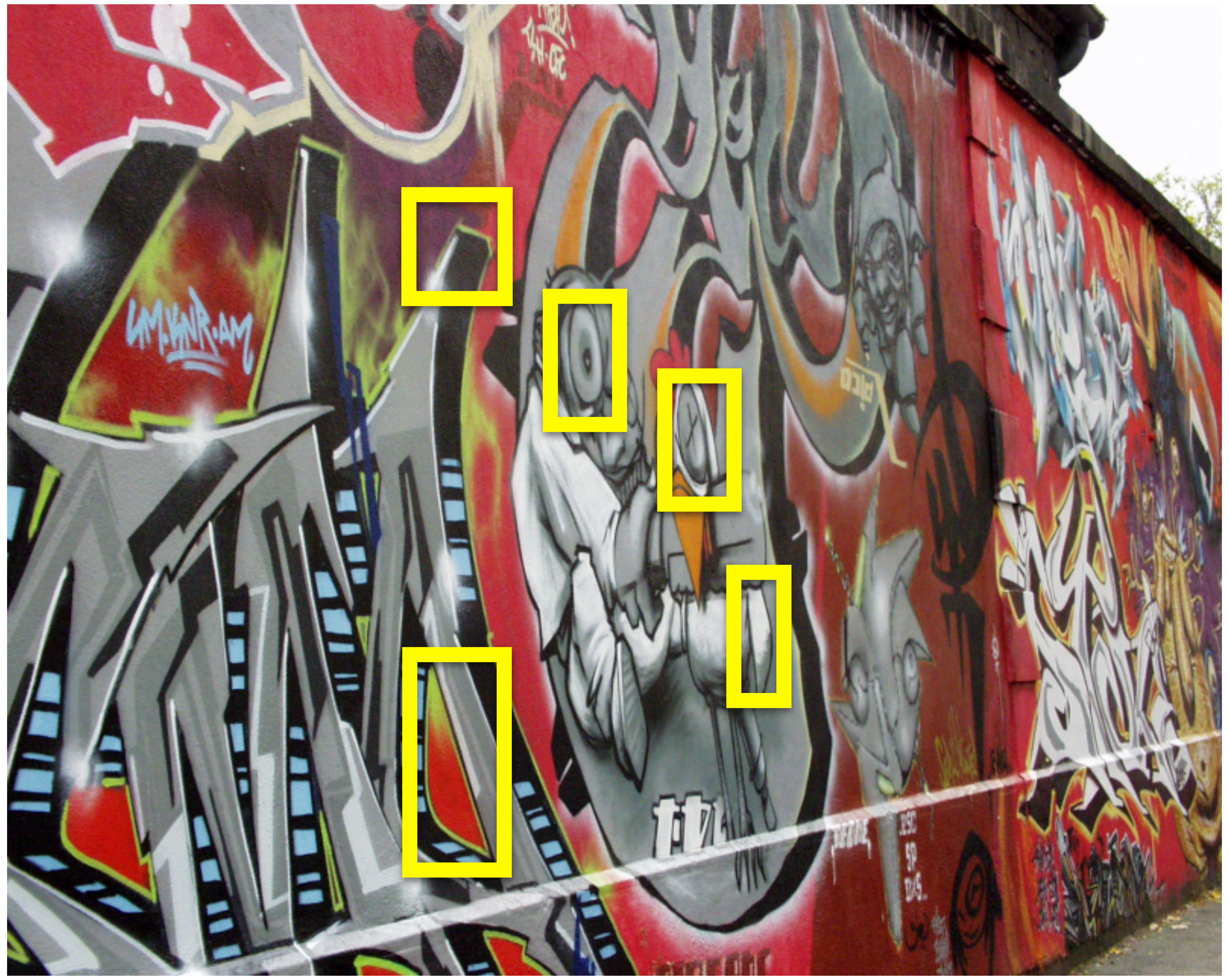
A basic problem in Computer Vision is to establish matches (correspondences) between images.

This has **many** applications: rigid/non-rigid tracking, object recognition, image registration, structure from motion, stereo...



Back to **Good Local Features**





Where are the good features, and
how do we match them?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Photometric Transformations

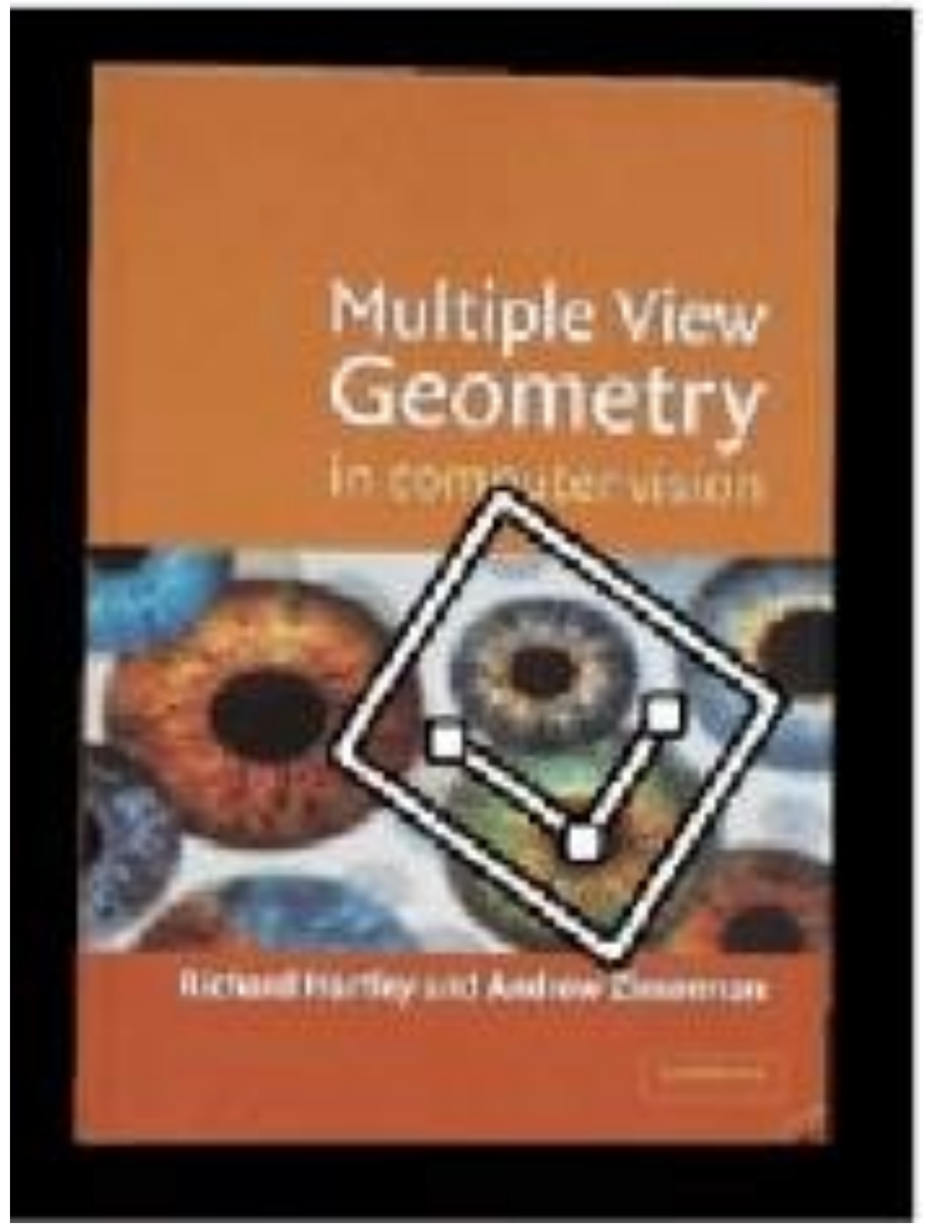
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Photometric Transformations

What can we use to deal with this?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Geometric Transformations

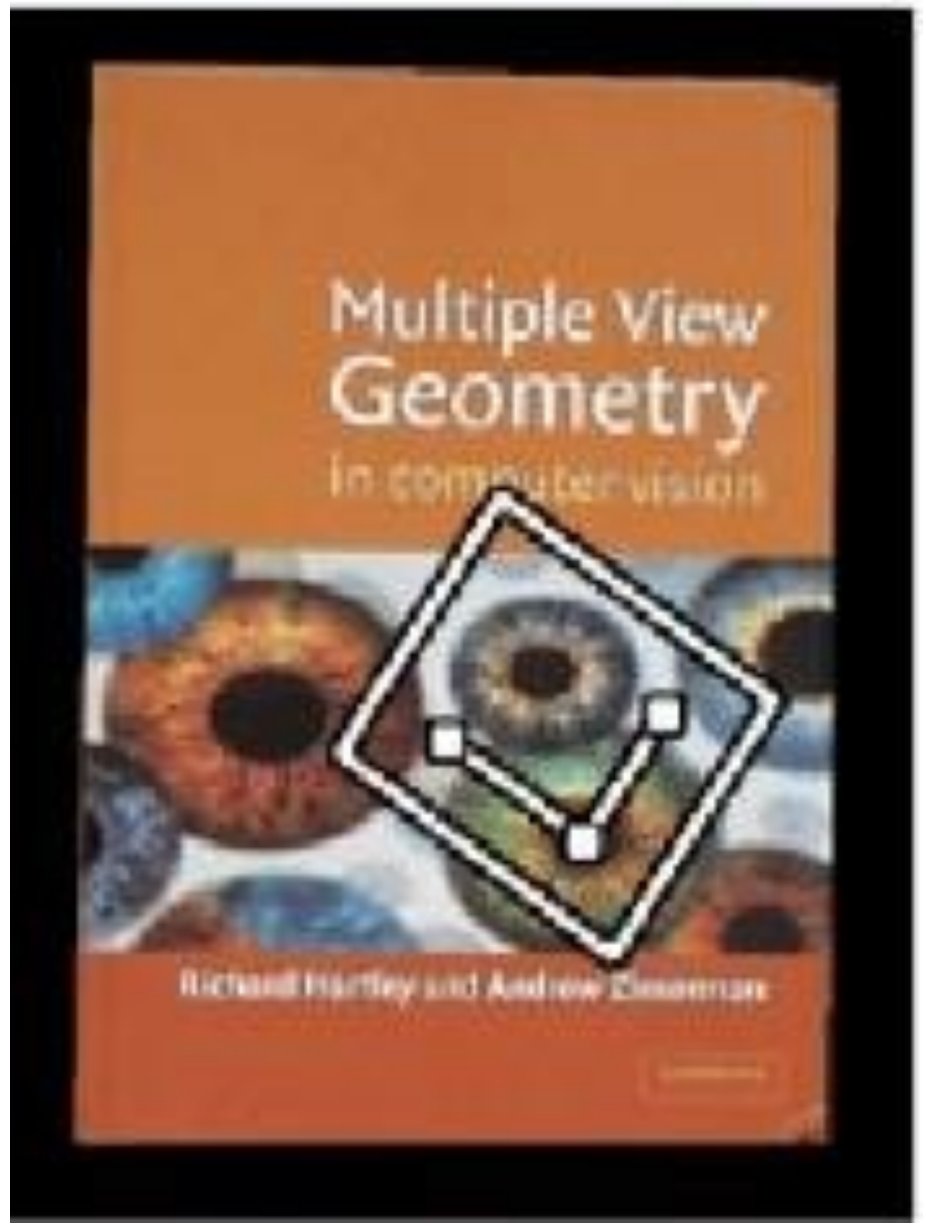


objects will appear at different scales,
translation and rotation

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Geometric Transformations

How can we deal with this?



objects will appear at different scales,
translation and rotation

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Lets assume for the moment we can figure out where the good features (patches) are … how do we **match** them?

Lets assume for the moment we can figure out where the good features (patches) are … how do we **match** them?

How do we localize good features to match (think back 1-2 lectures)?

Lets assume for the moment we can figure out where the good features (patches) are … how do we **match** them?

How do we localize good features to match (think back 1-2 lectures)?

Harris, Blob are locally distinct (this is minimally what we need)

Back to **Good Local Features**

How do we know which **corner** goes with which?

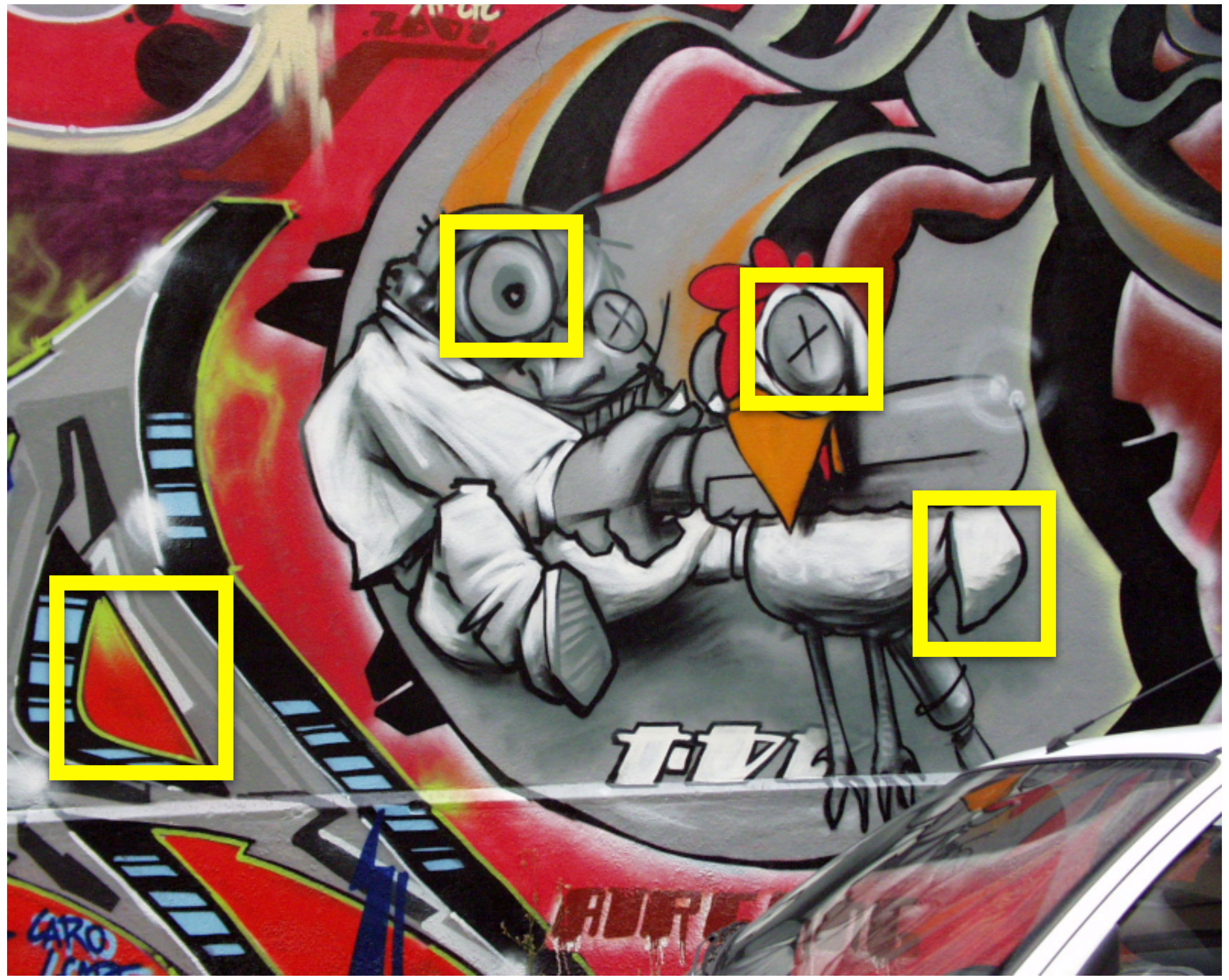
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

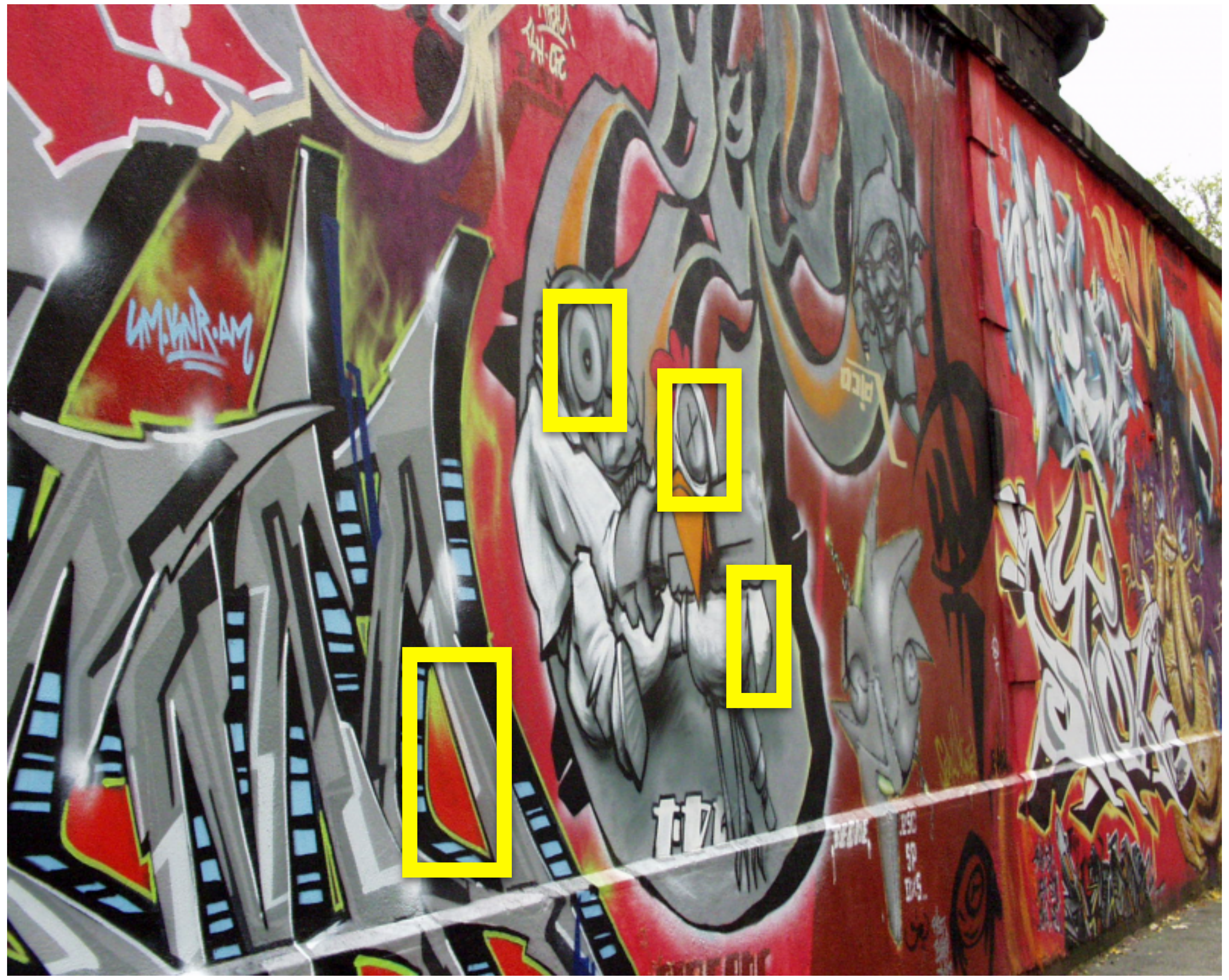
Back to **Good Local Features**

How do we know which **blob** goes with which?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Back to **Good Local Features**

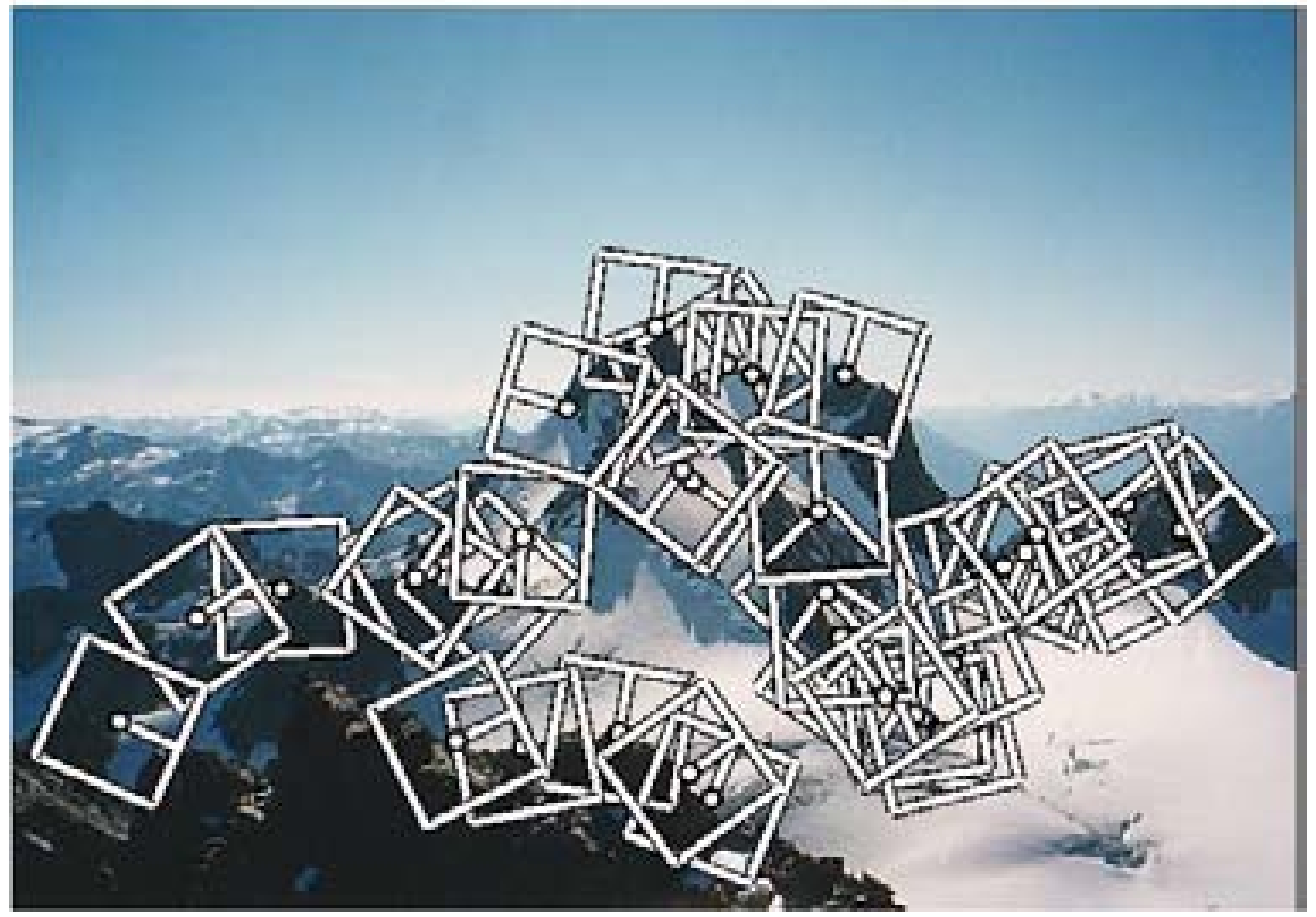




Patch around the local feature is very informative

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Feature **Detector**



Corners/Blobs

Regions

Edges

Straight Lines

Feature **Descriptor**

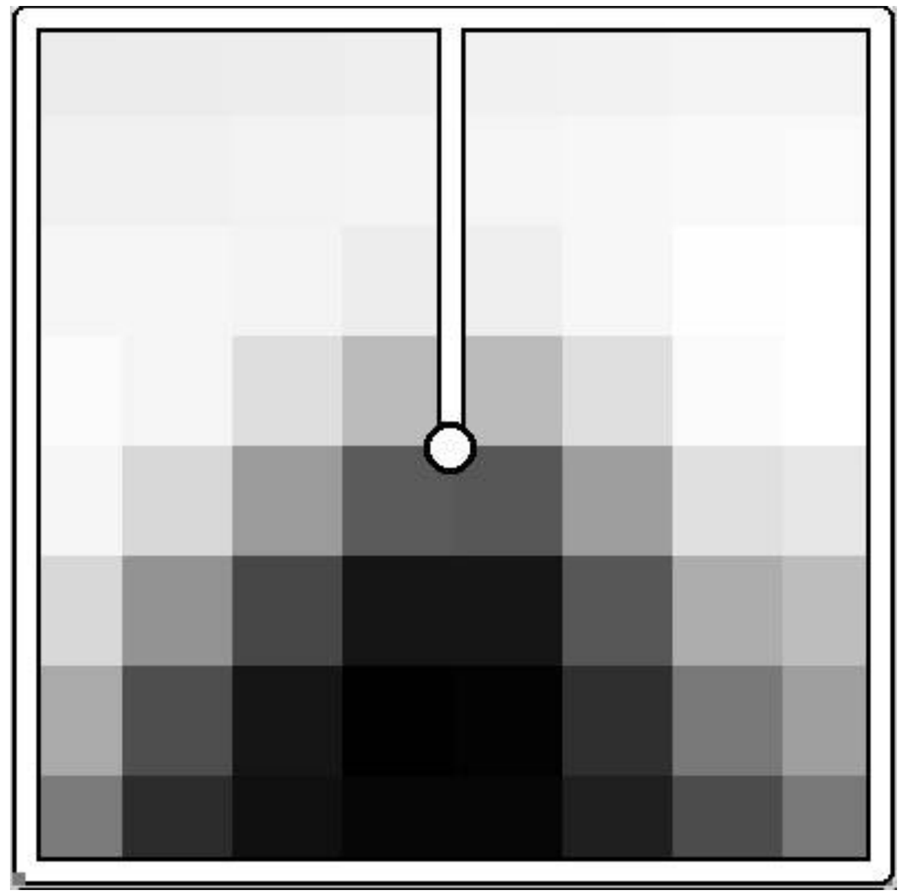
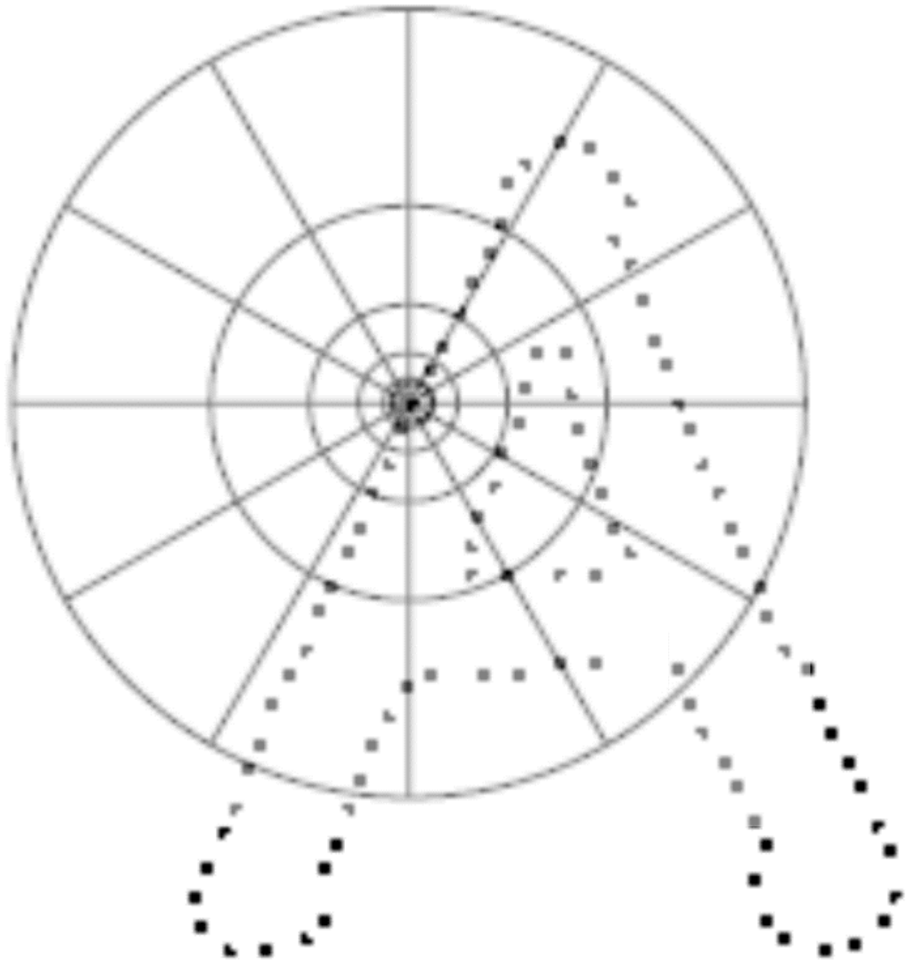
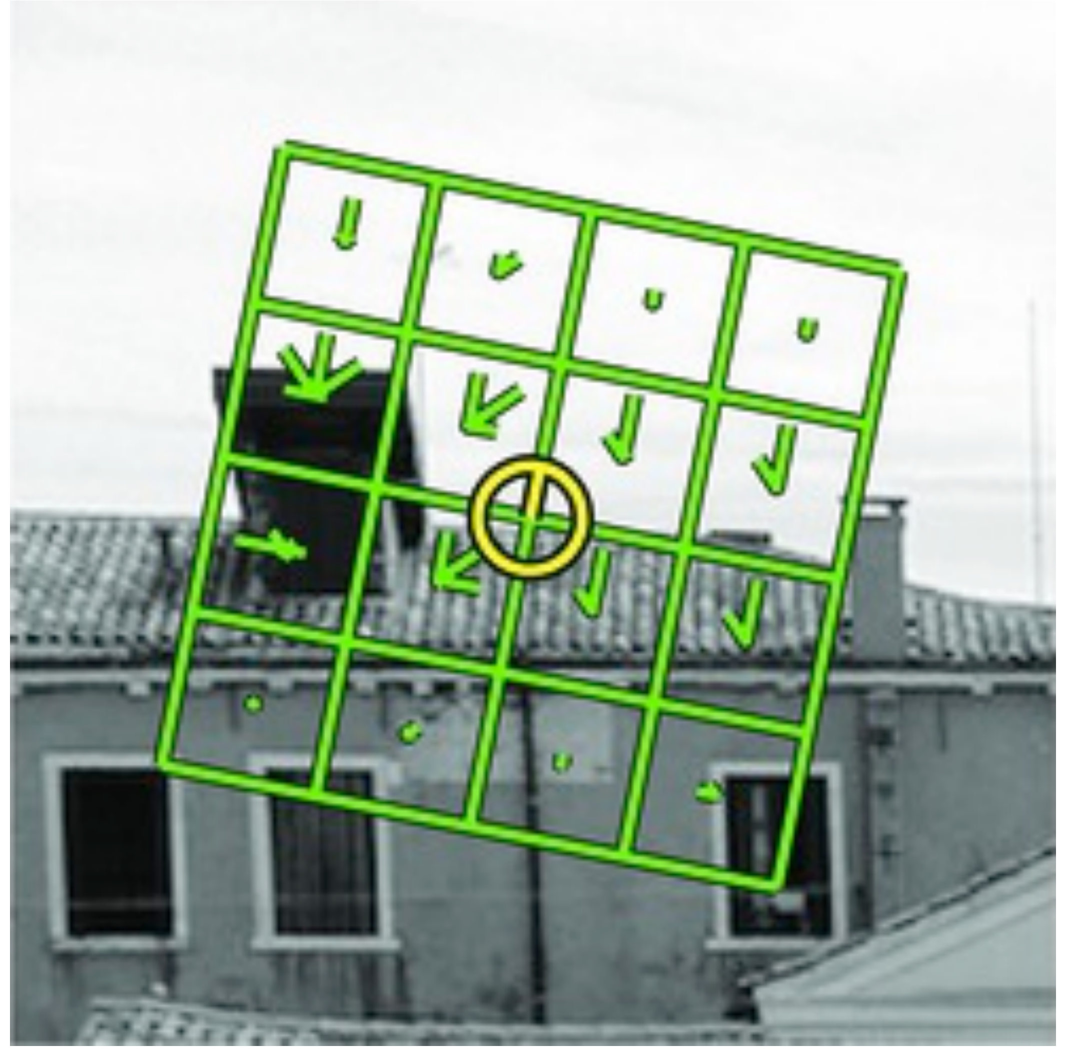


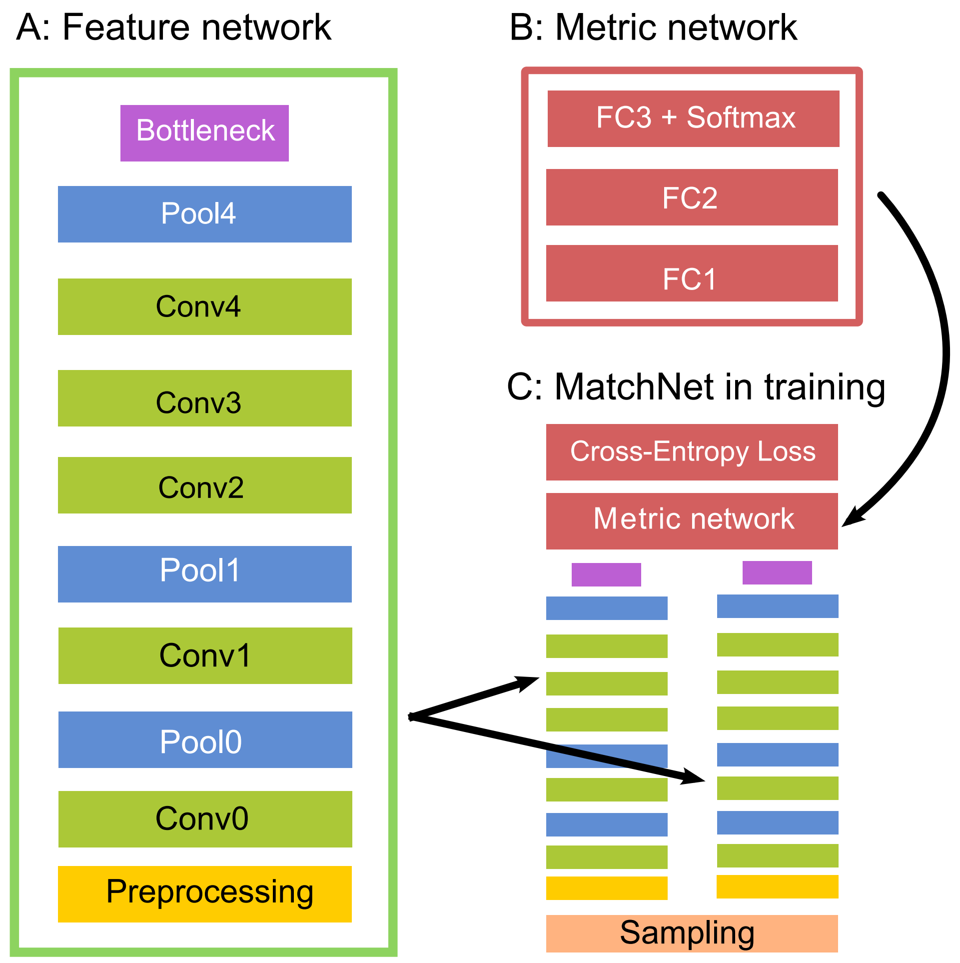
Image Patch



Shape Context



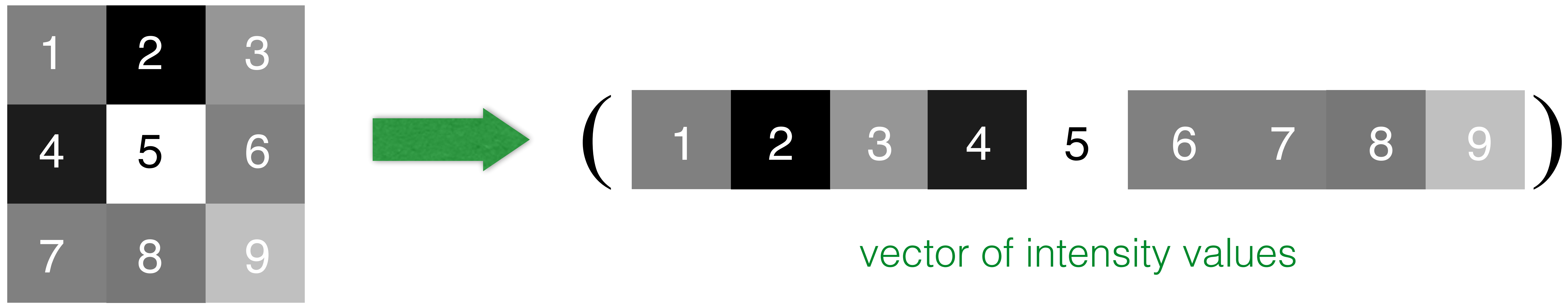
SIFT



Learned Descriptors

Intensity Image

Just use the pixel values of the patch



Perfectly fine if geometry and appearance is unchanged

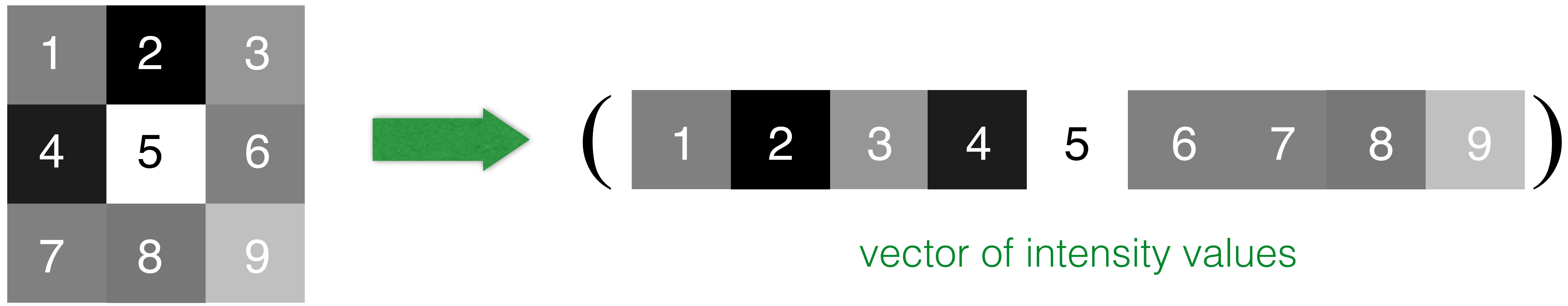
(a.k.a. template matching)

What are the problems?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Intensity Image

Just use the pixel values of the patch



Perfectly fine if geometry and appearance is unchanged
(a.k.a. template matching)

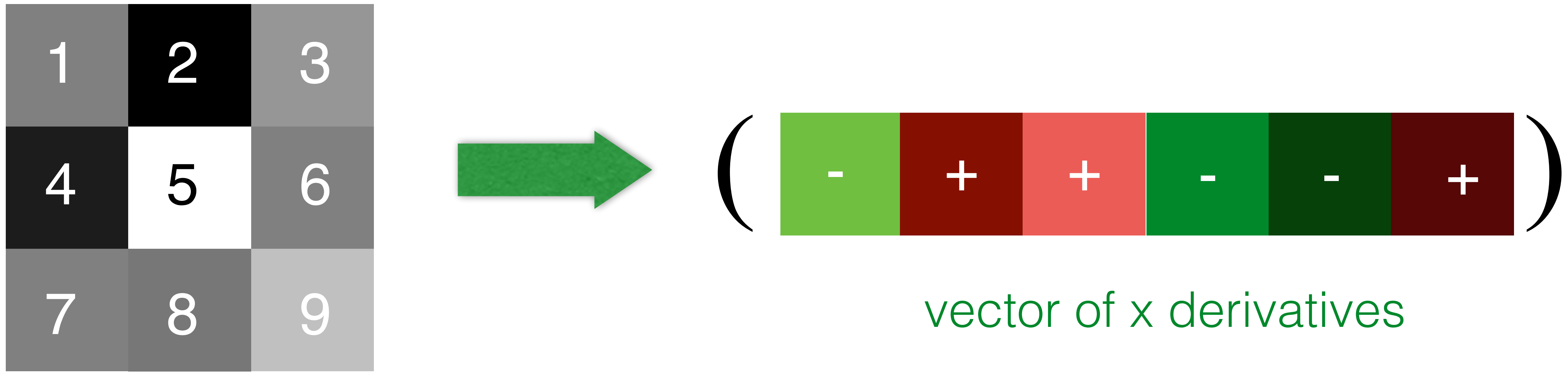
What are the problems?

How can you be less sensitive to absolute intensity values?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Image **Gradients** / **Edges**

Use pixel differences



vector of x derivatives

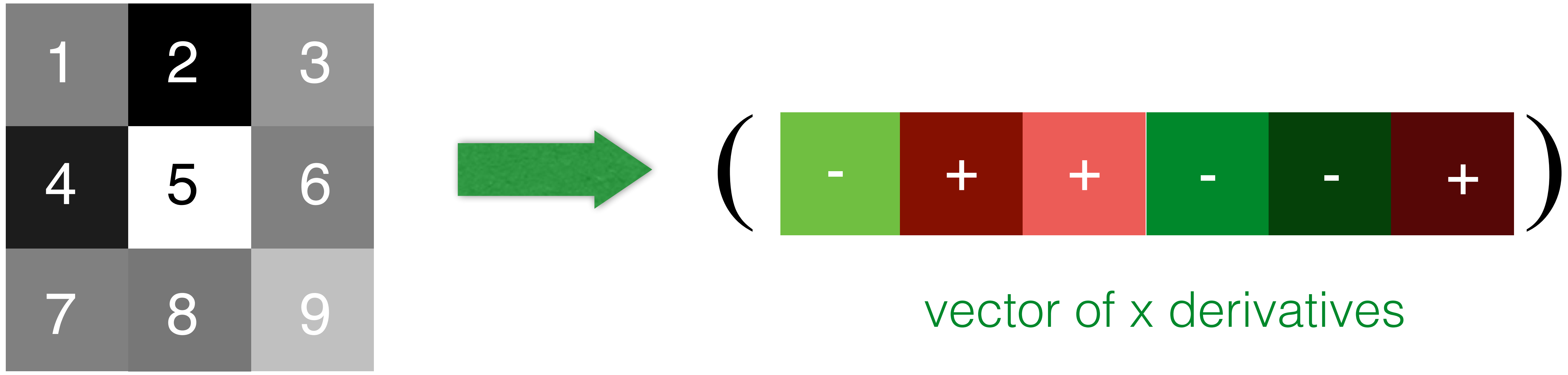
Feature is invariant to absolute intensity values

What are the problems?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Image **Gradients** / **Edges**

Use pixel differences



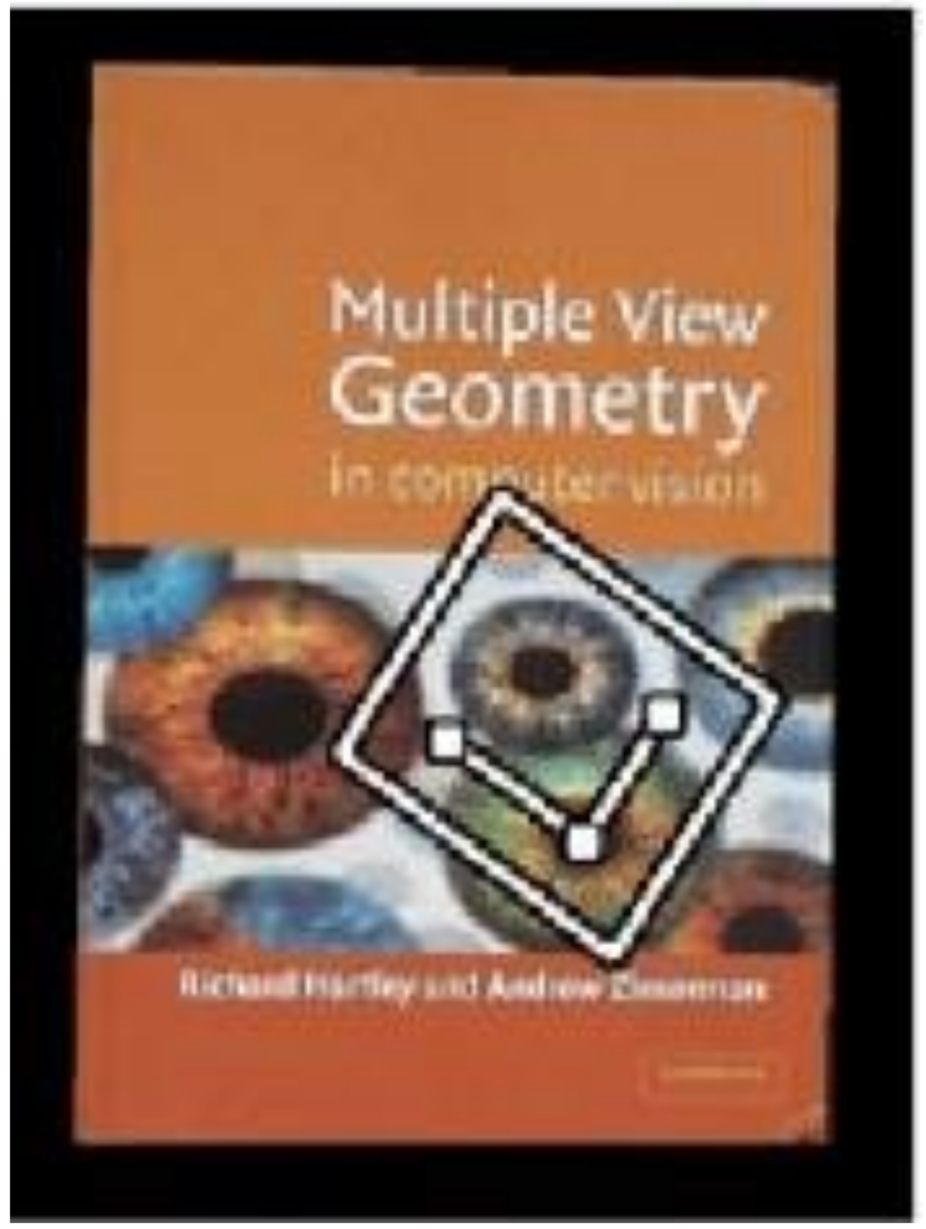
Feature is invariant to absolute intensity values

What are the problems?

How can you be less sensitive to deformations?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Geometric Transformations

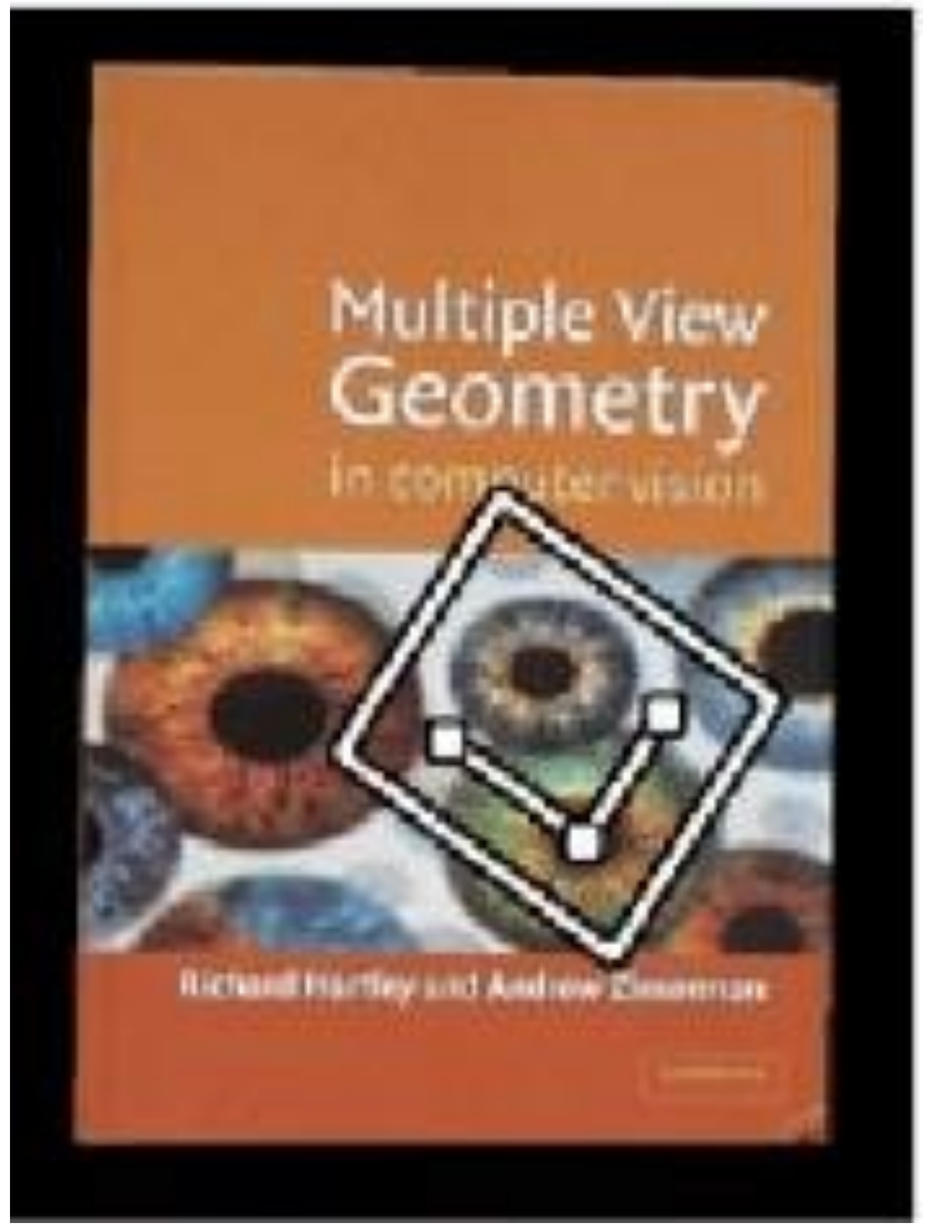


objects will appear at different scales,
translation and rotation

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Geometric Transformations

How can we deal with this?



objects will appear at different scales,
translation and rotation

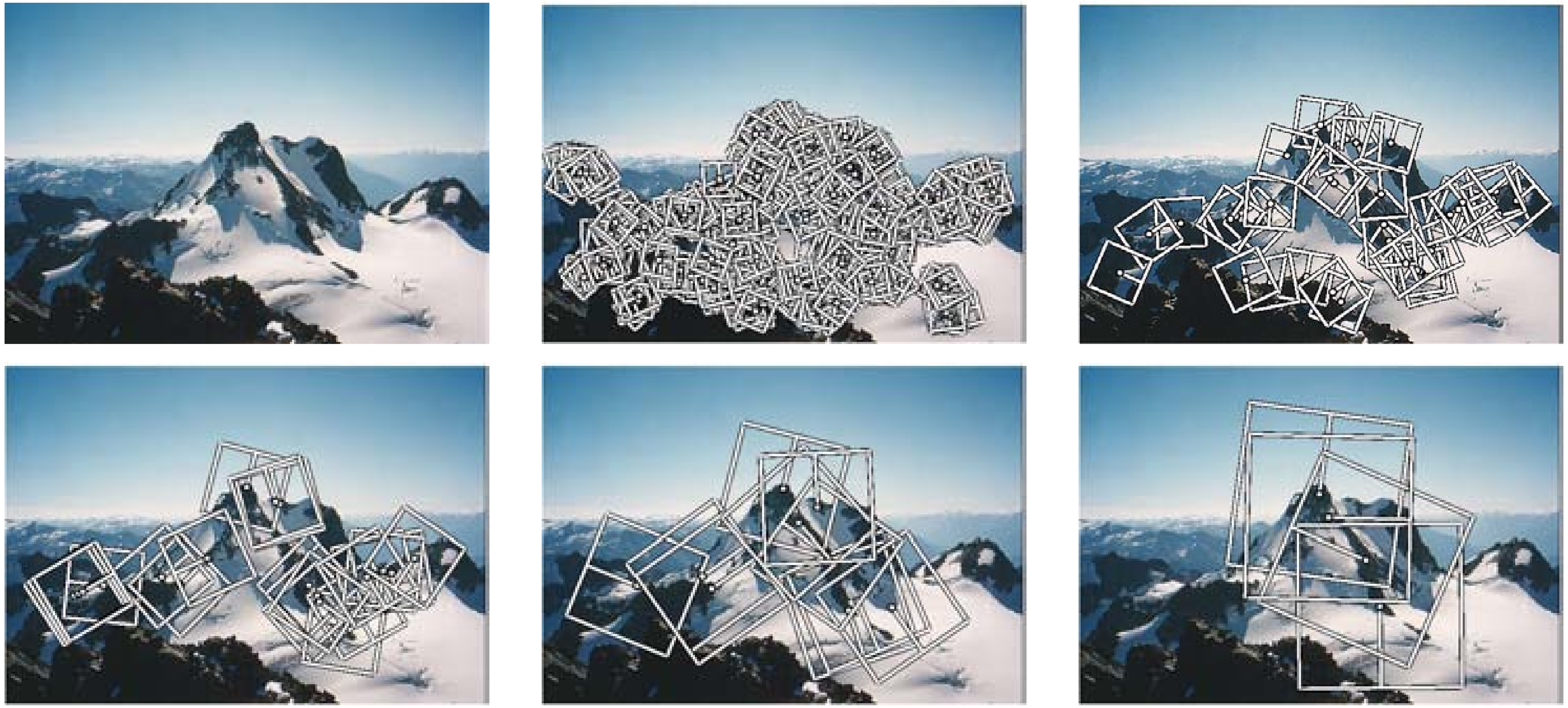
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Local Coordinate Frame

One way to achieve invariance is to use **local coordinate frames** that follow the surface transformation (covariant) and compute features descriptors in them

Strategy #1: Detecting **Scale / Orientation**

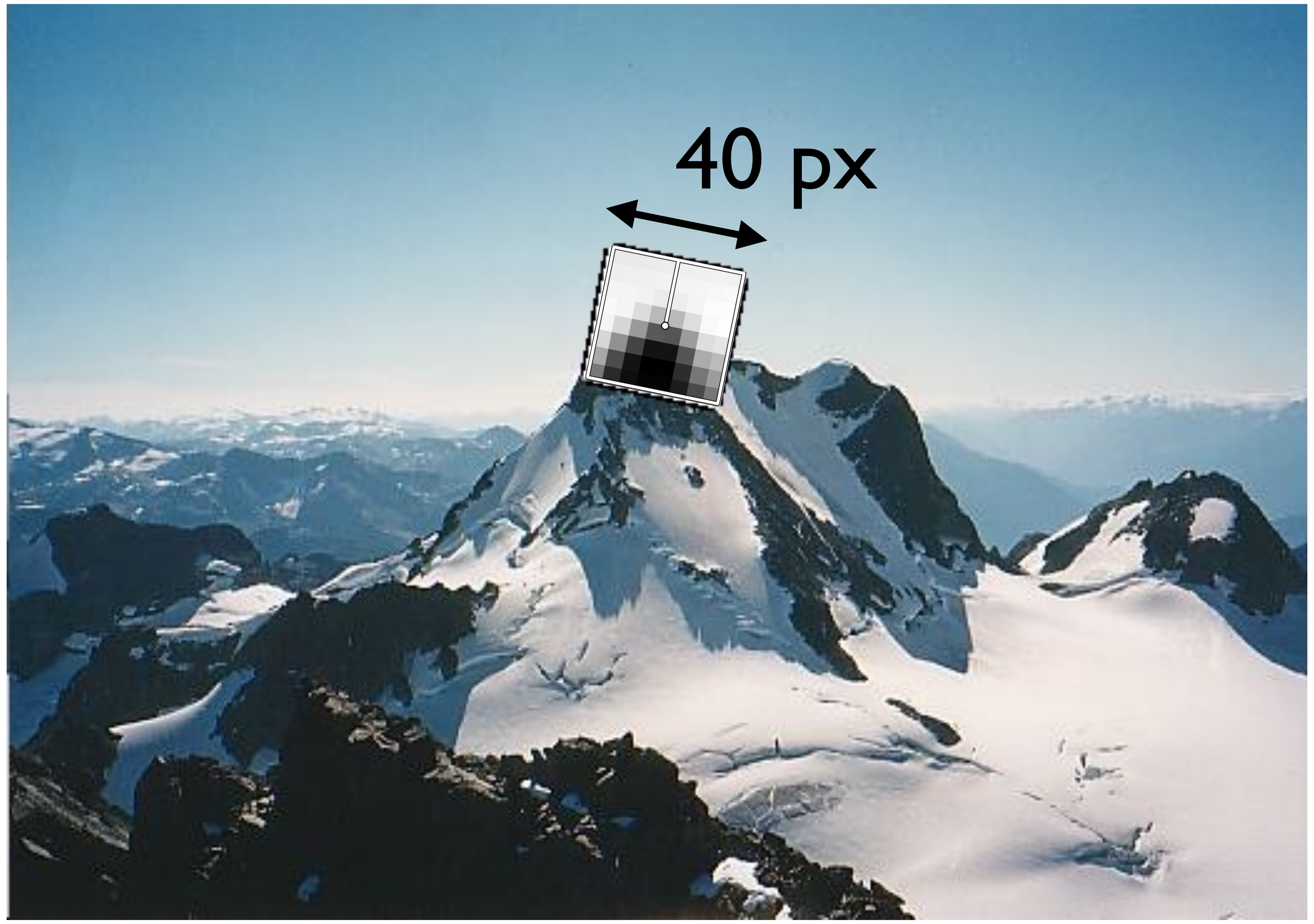
A common approach is to detect a local scale and orientation for each feature point

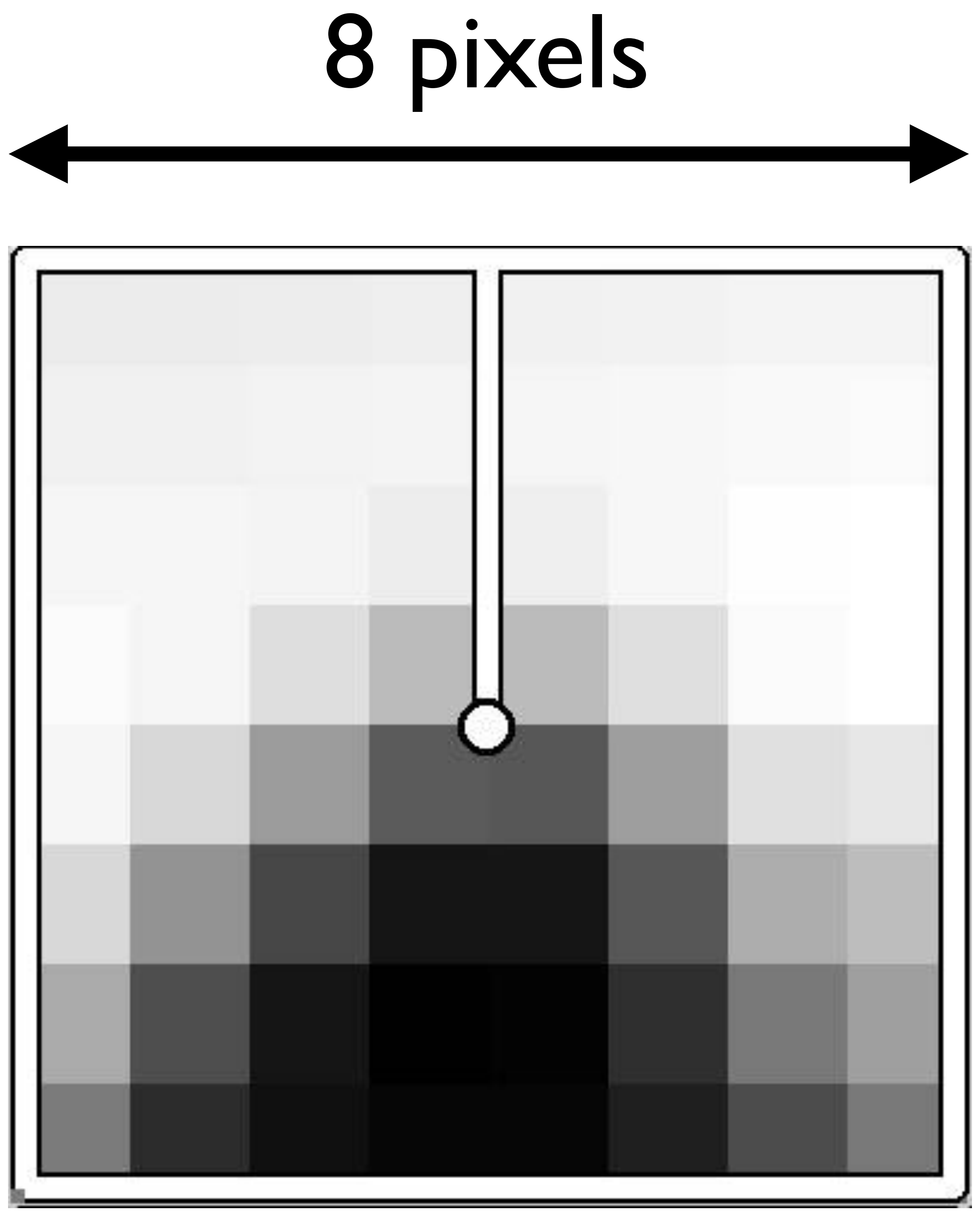


e.g., extract Harris at multiple scales and align to the local gradient

Strategy #1: Detecting **Scale / Orientation**

A common approach is to detect a local scale and orientation for each feature point

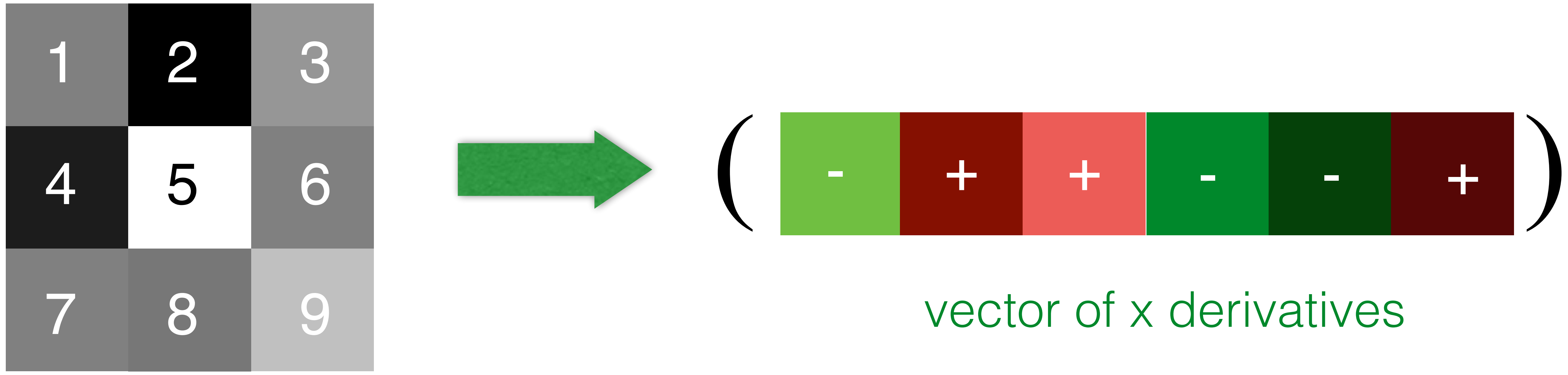




e.g., extract Harris at multiple scales and align to the local gradient

Strategy #2: Represent Distributions over Gradients

Use pixel differences



Feature is invariant to absolute intensity values

Where does **SIFT** fit in?

Representation	Result is. . .	Approach	Technique
intensity	dense (2D)	template matching	(normalized) correlation, SSD
edge	relatively sparse (1D)	derivatives	$\nabla^2 G$, Canny
"corner" / "blob"	sparse (0D)	locally distinct features	Harris, SIFT

Object **Recognition** with Scale Invariant Feature Transform

Task: Identify objects or scenes and determine their pose and model parameters

Applications:

— Industrial automation and inspection

— Mobile robots, toys, user interfaces

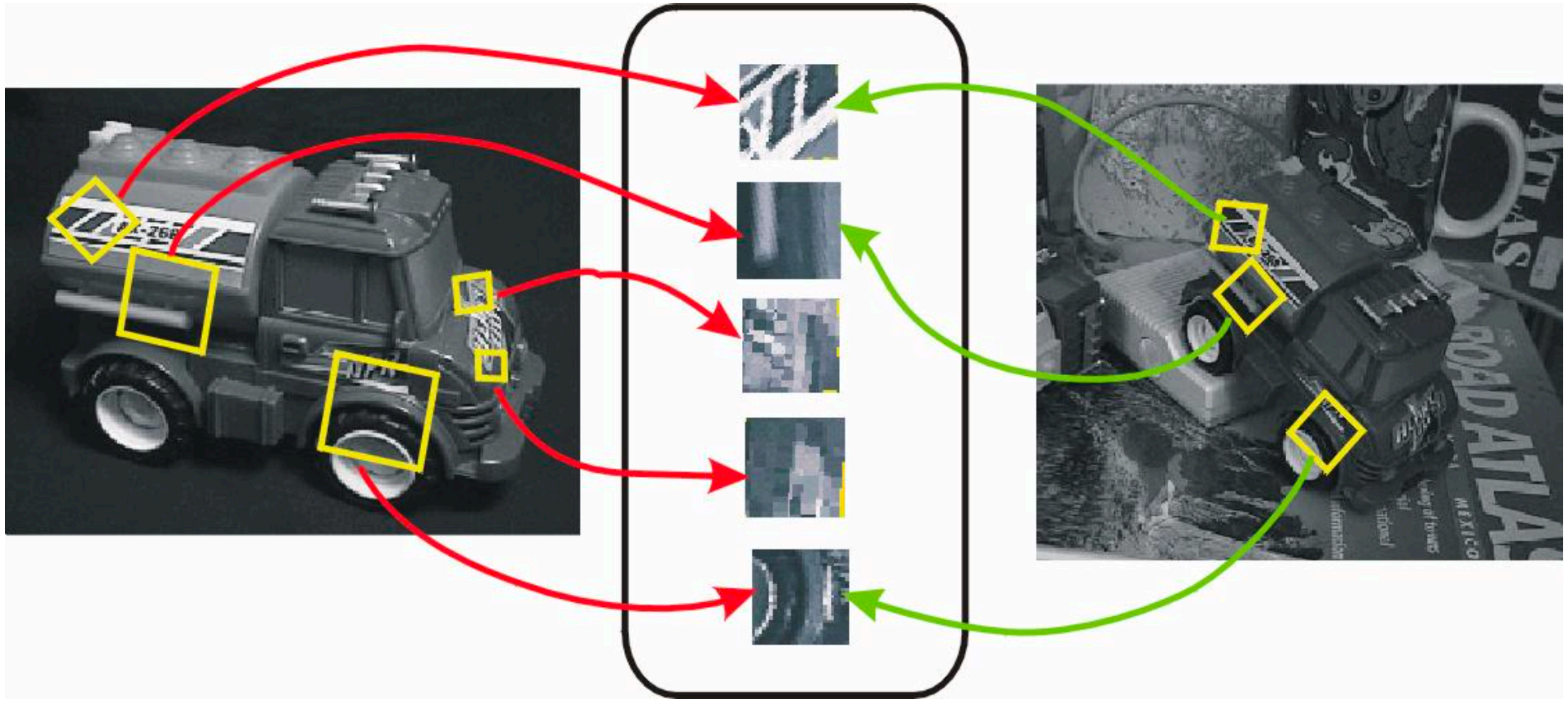
— Location recognition

— Digital camera panoramas

— 3D scene modeling, augmented reality

David Lowe's Invariant Local Features

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



SIFT Features

David Lowe's Invariant Local Features

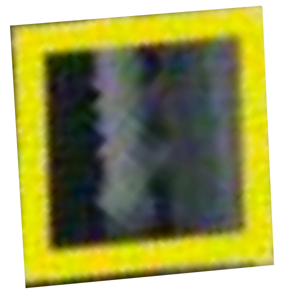
Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters

David Lowe's Invariant Local Features

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

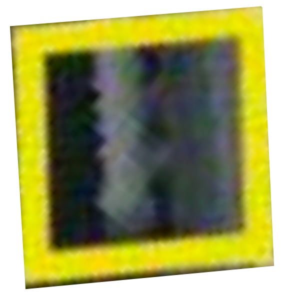
David Lowe's Invariant Local Features

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



David Lowe’s Invariant Local Features

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



Advantages of Invariant Local Features

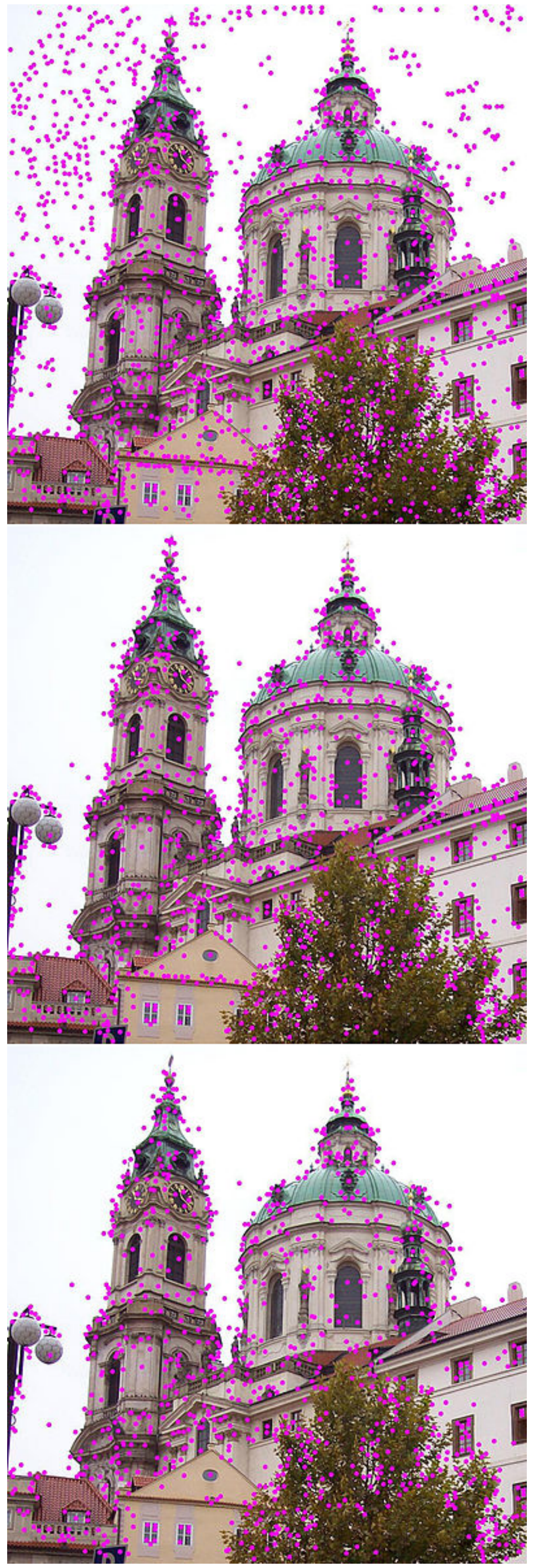
Locality: features are local, so robust to occlusion and clutter (no prior segmentation)

Distinctiveness: individual features can be matched to a large database of objects

Quantity: many features can be generated for even small objects

Efficiency: close to real-time performance

Scale Invariant Feature Transform (**SIFT**)

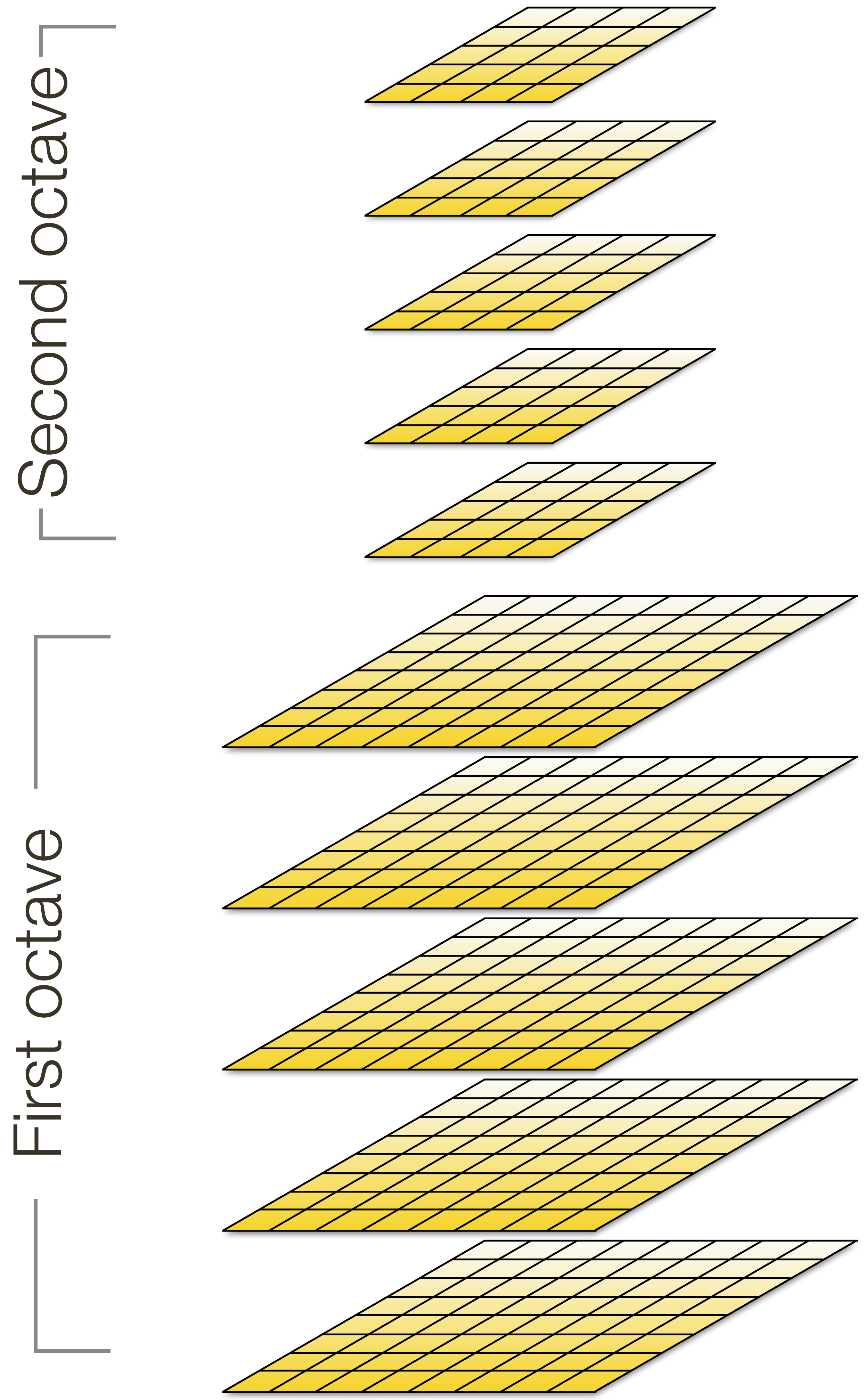


SIFT describes both a **detector** and **descriptor**

1. Multi-scale extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

1. Multi-scale Extrema Detection

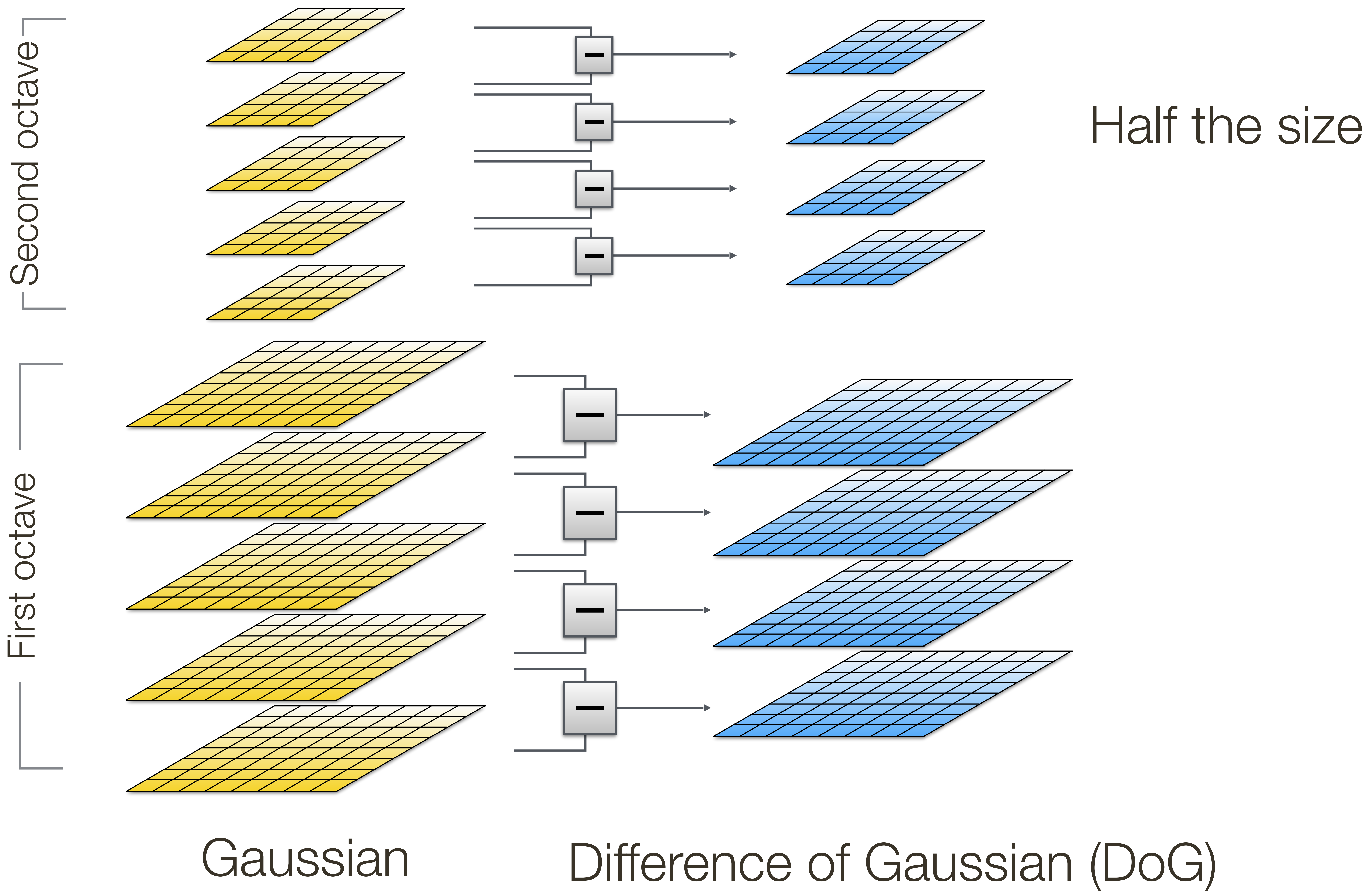


Gaussian

Half the size

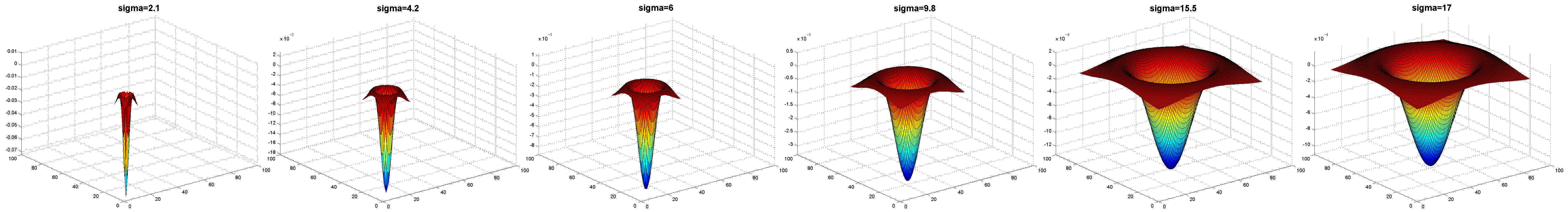
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

1. Multi-scale Extrema Detection



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Recall: Applying **Laplacian** Filter at Different **Scales**



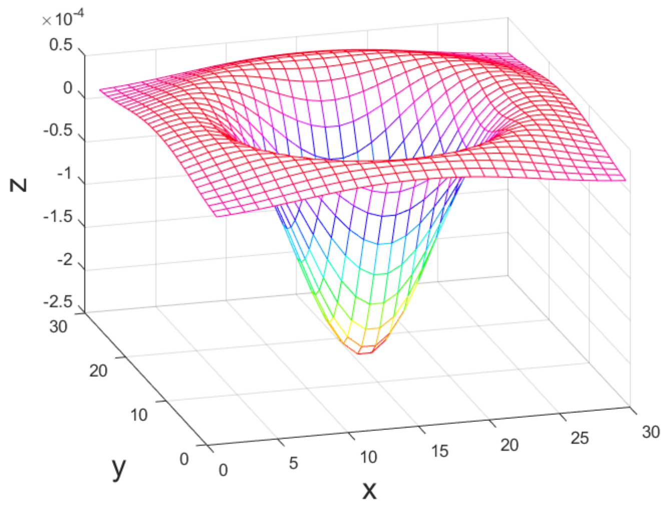
Full size

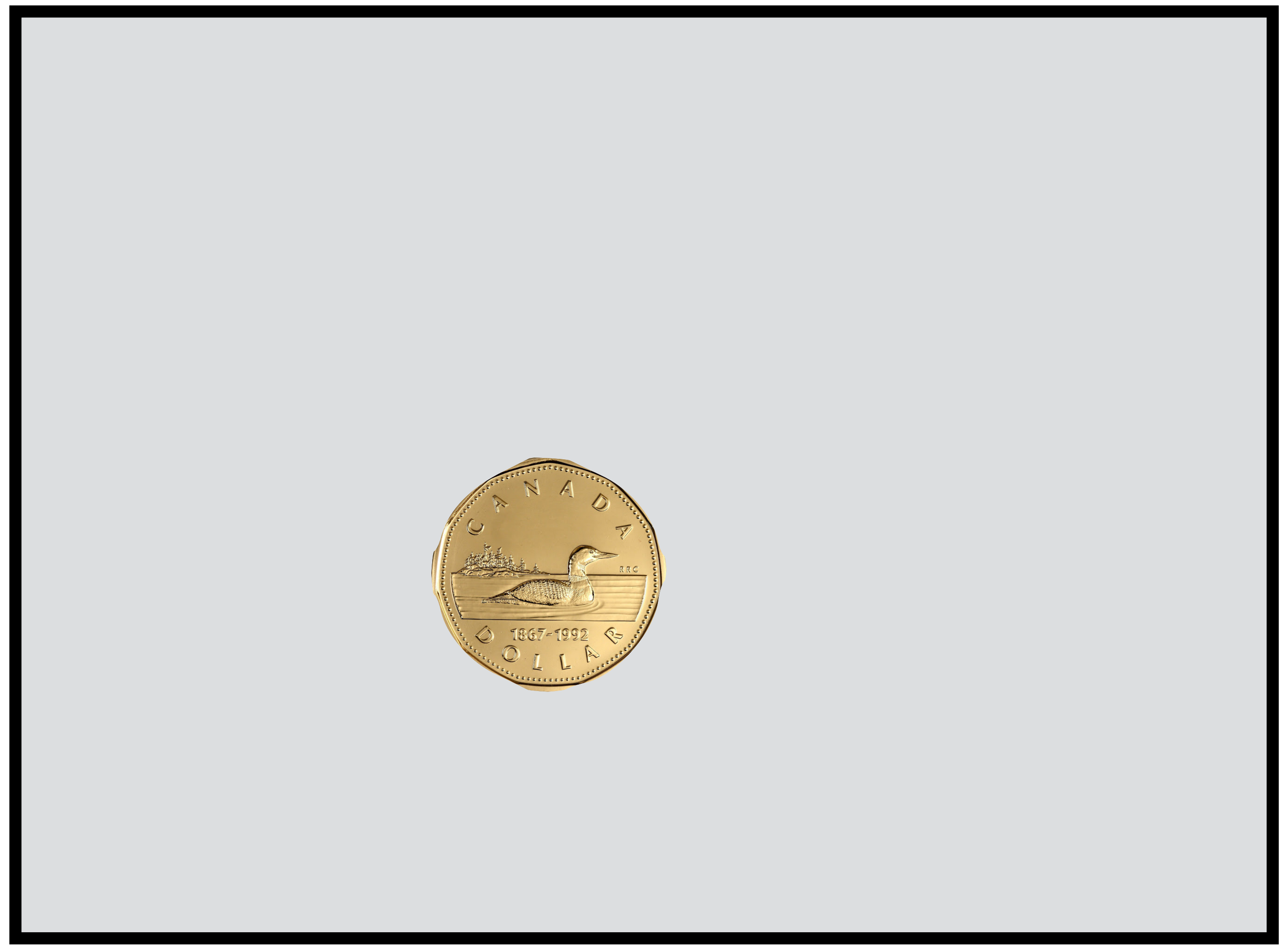
3/4 size

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Searching over **Scale**-space

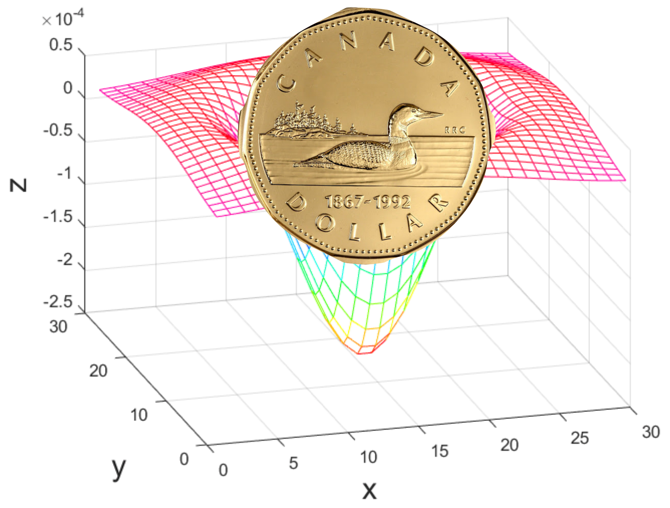
σ

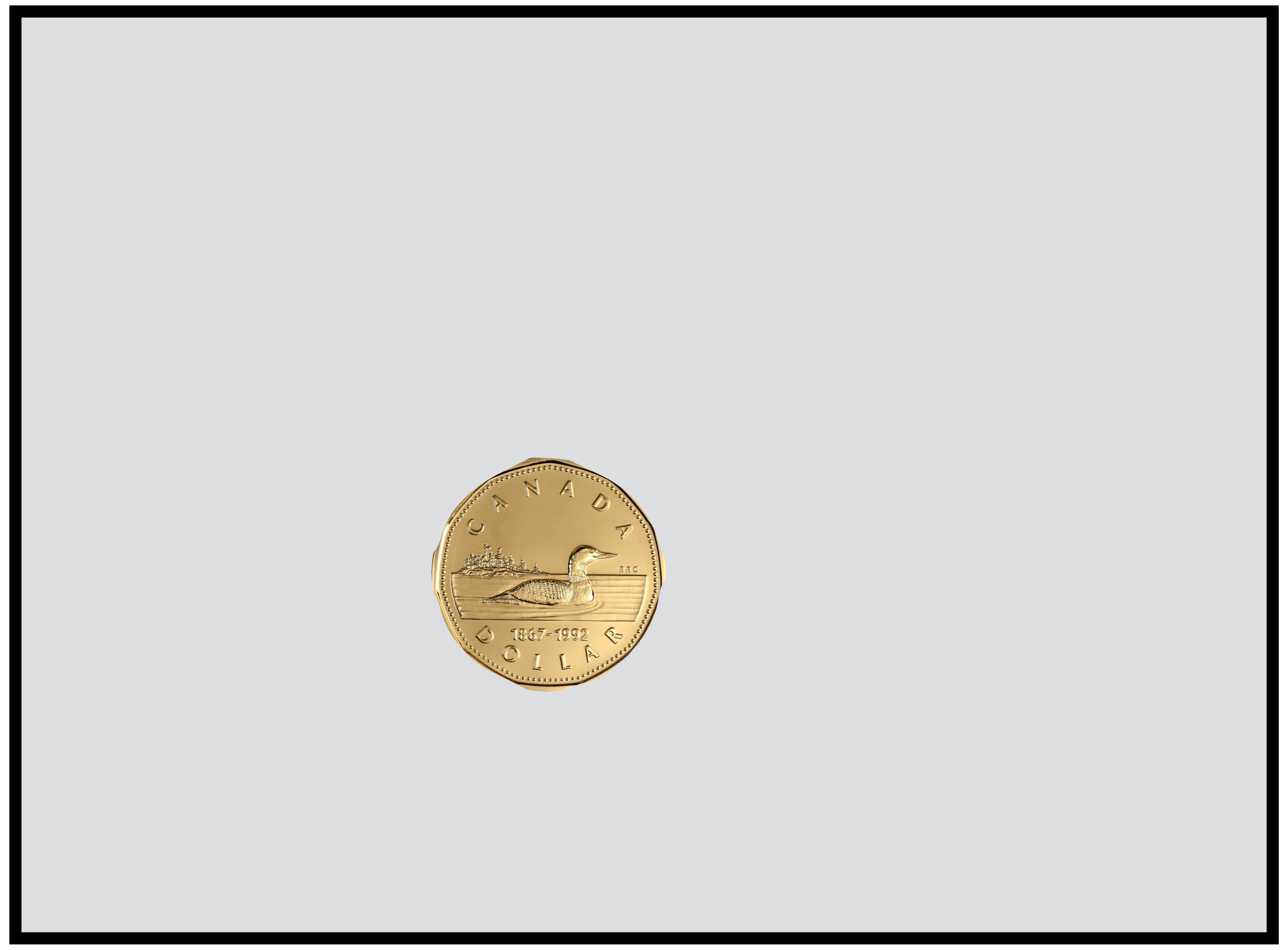




Searching over **Scale**-space

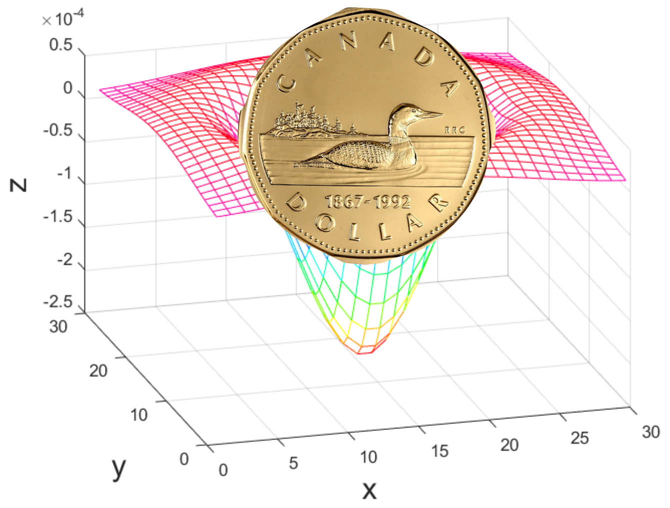
σ

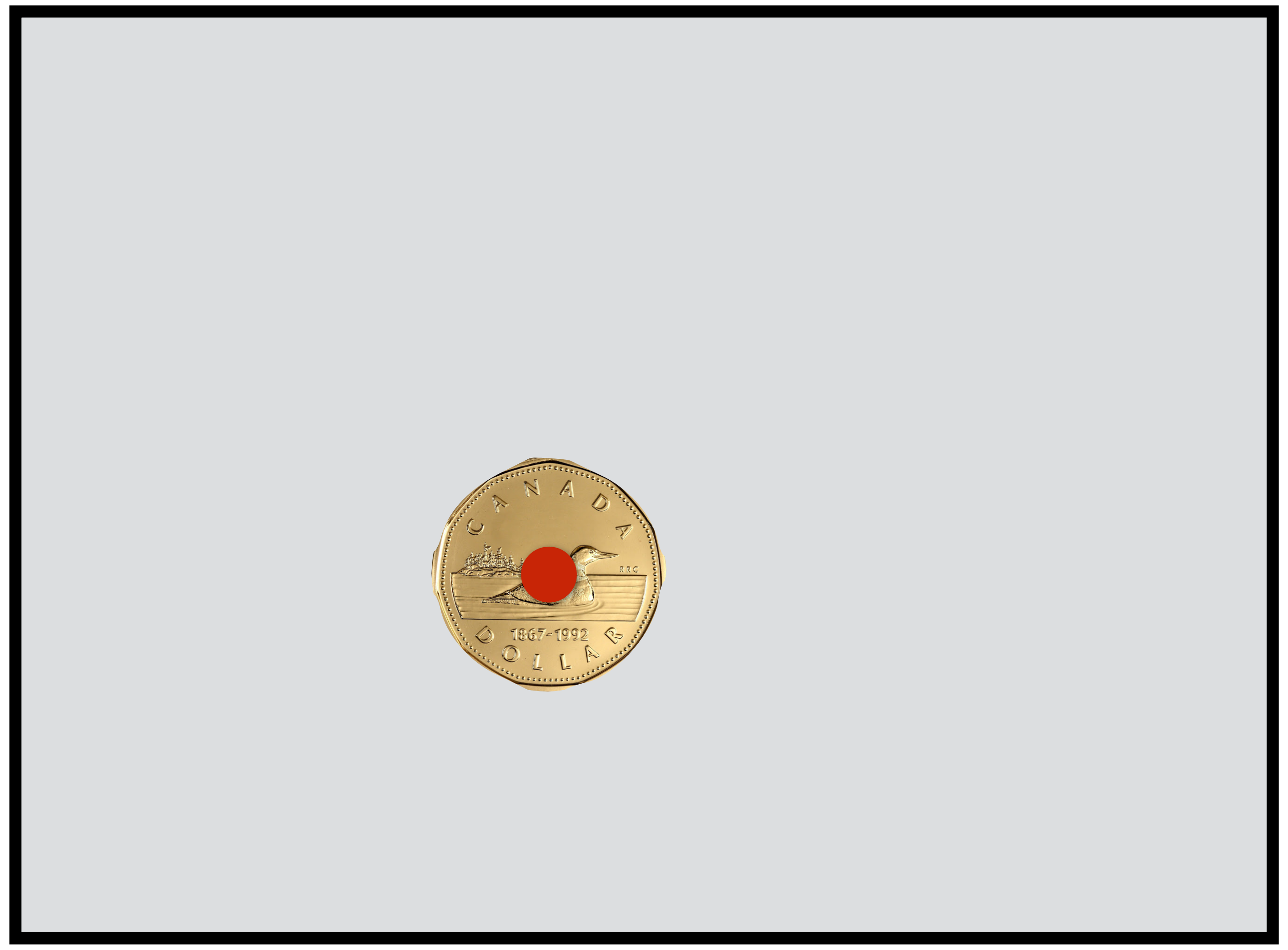




Searching over **Scale**-space

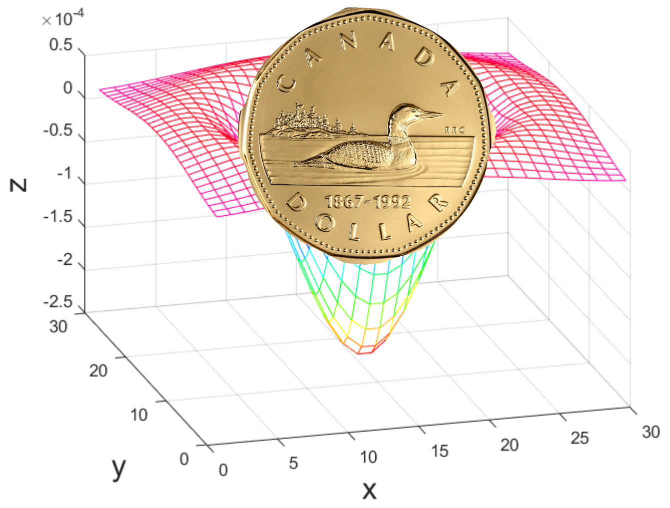
σ

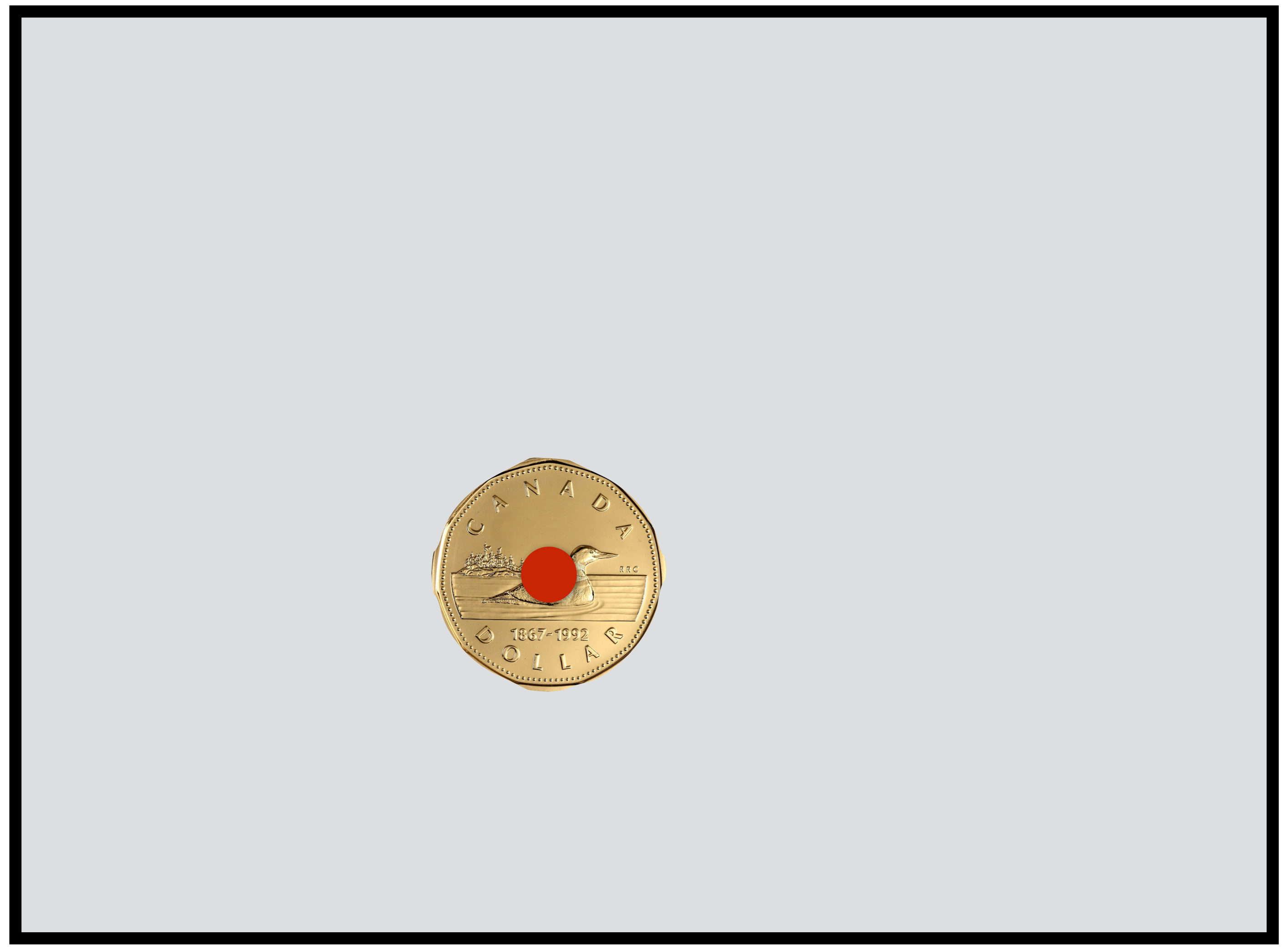




Searching over **Scale**-space

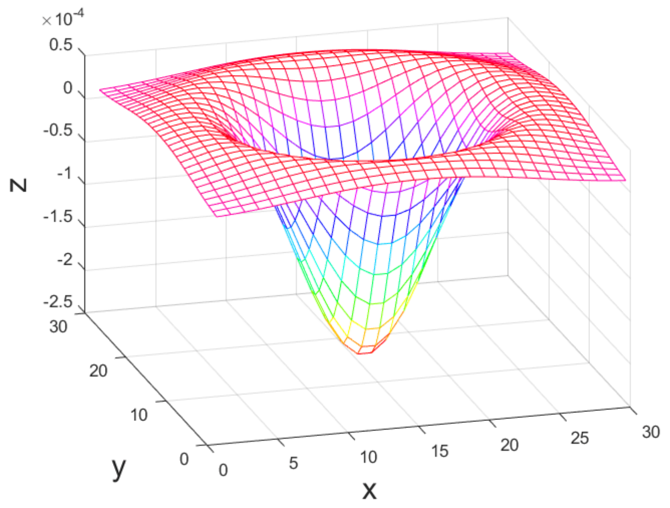
σ



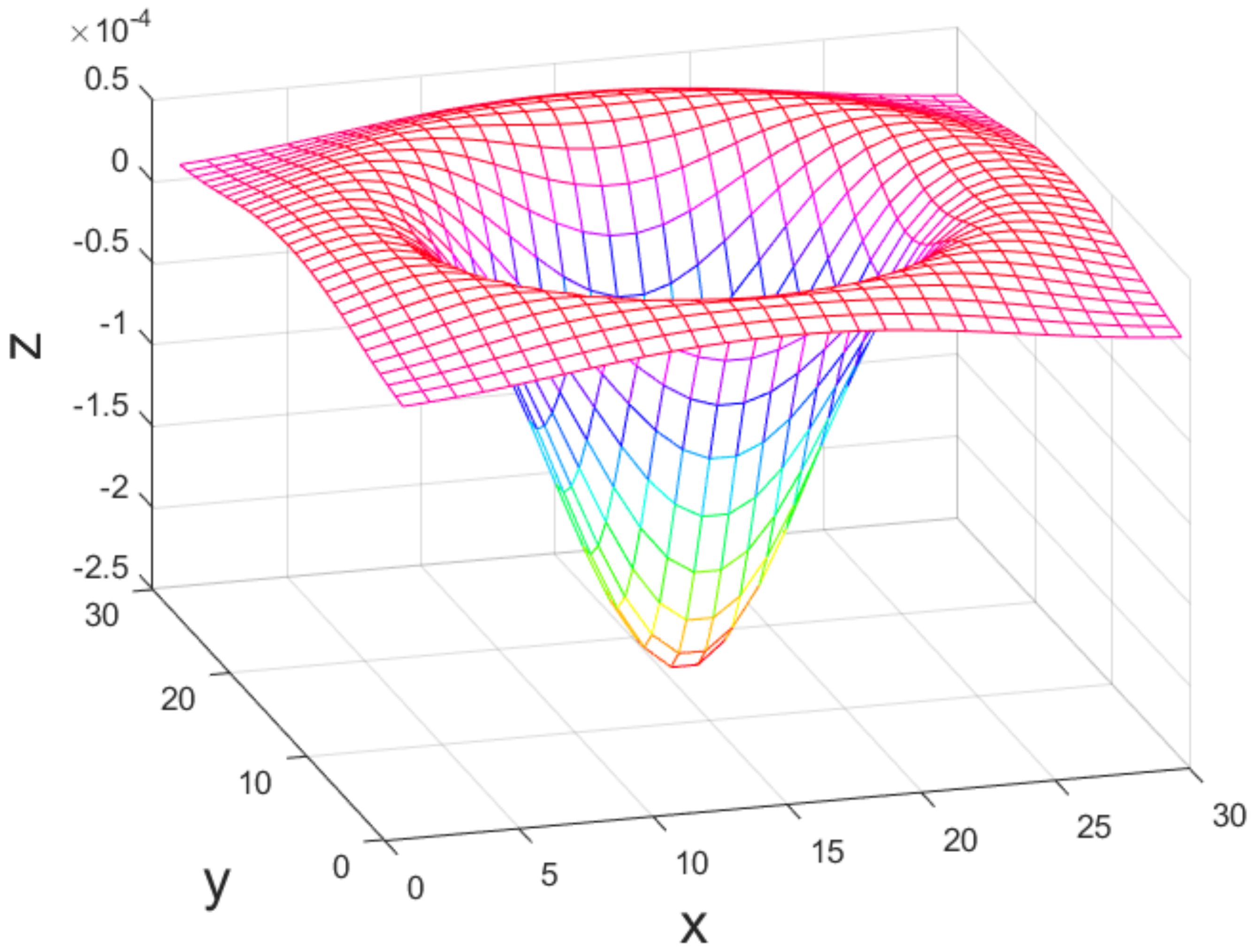


Searching over **Scale**-space

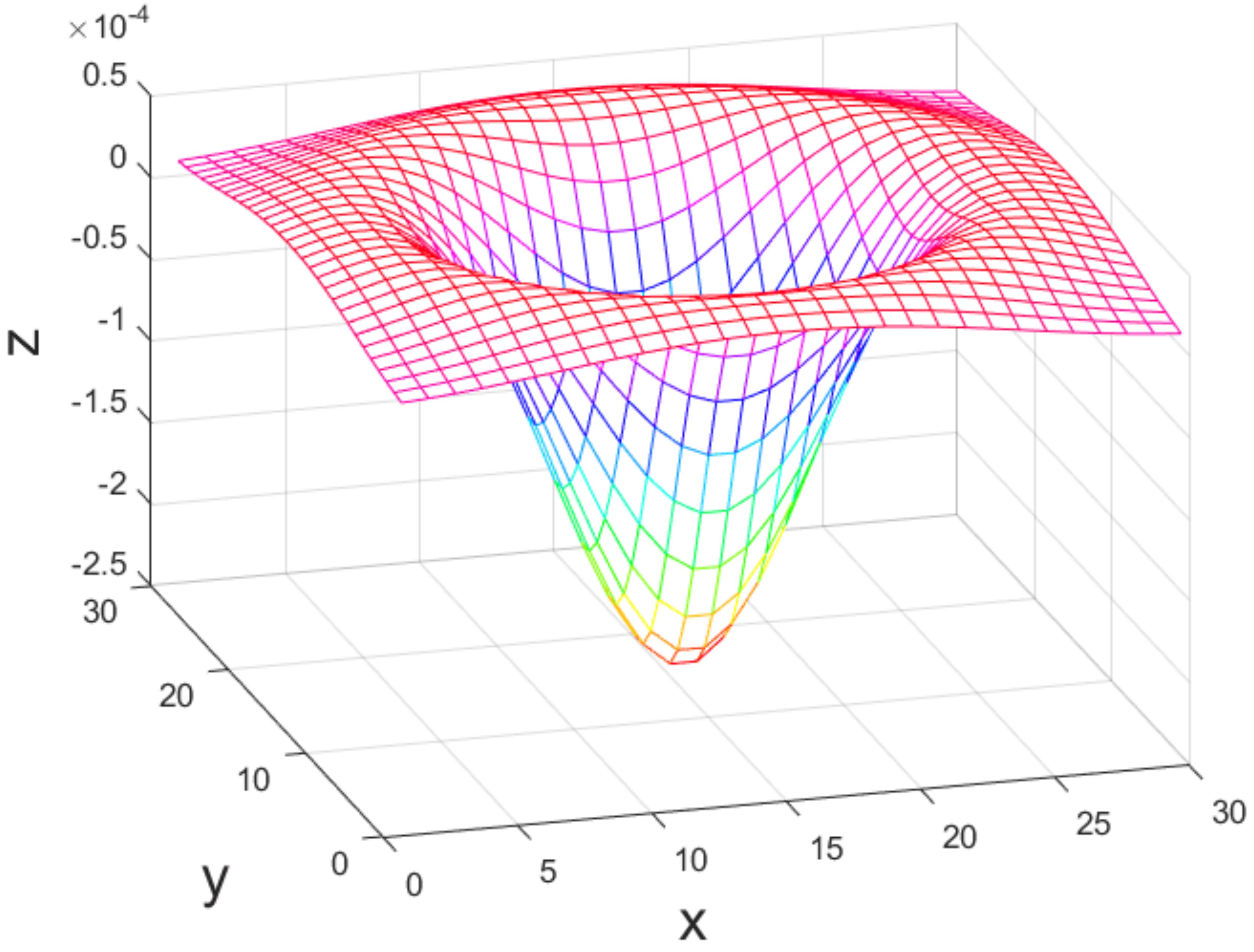
σ

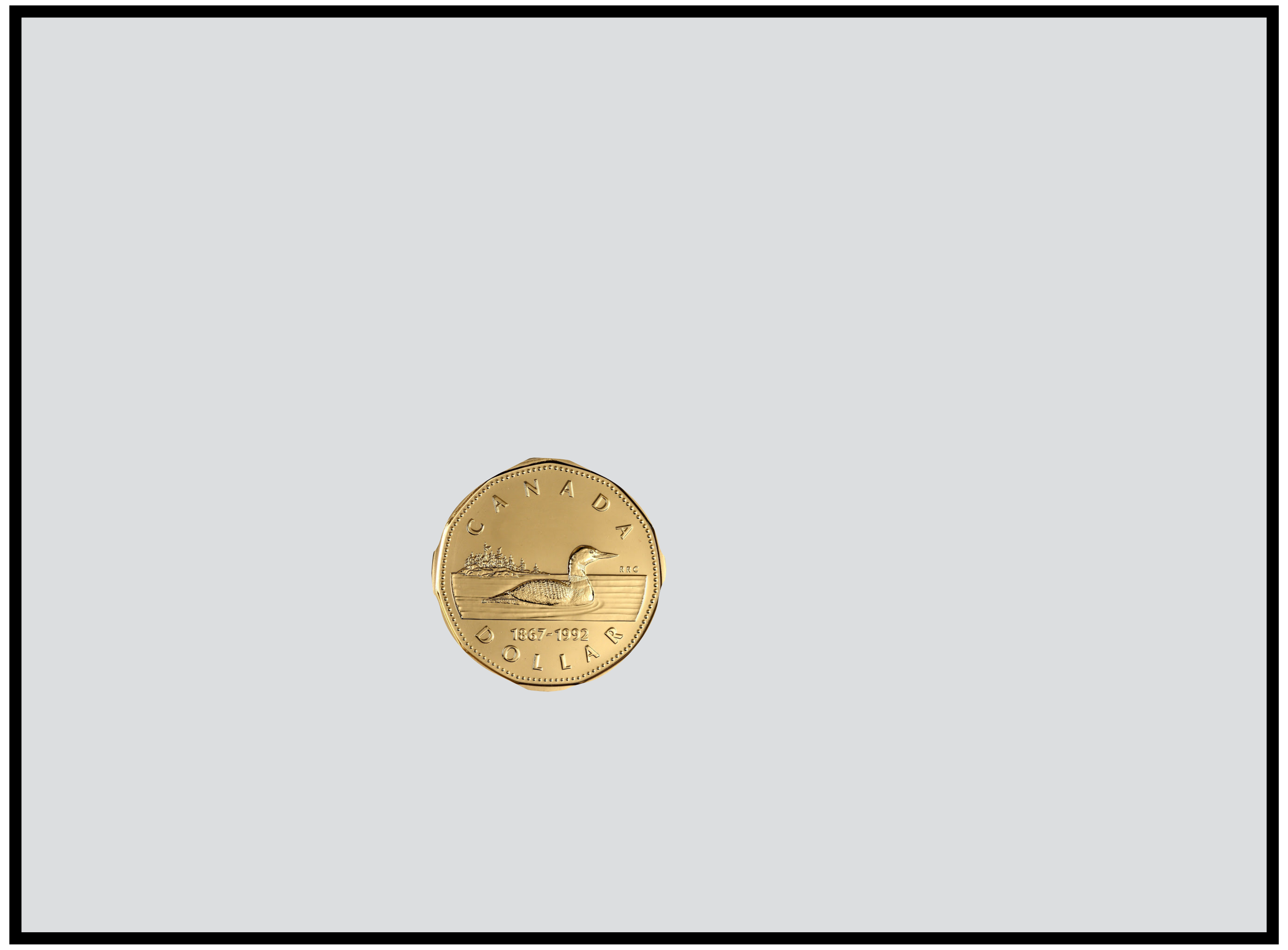


$\sigma' = 2\sigma$



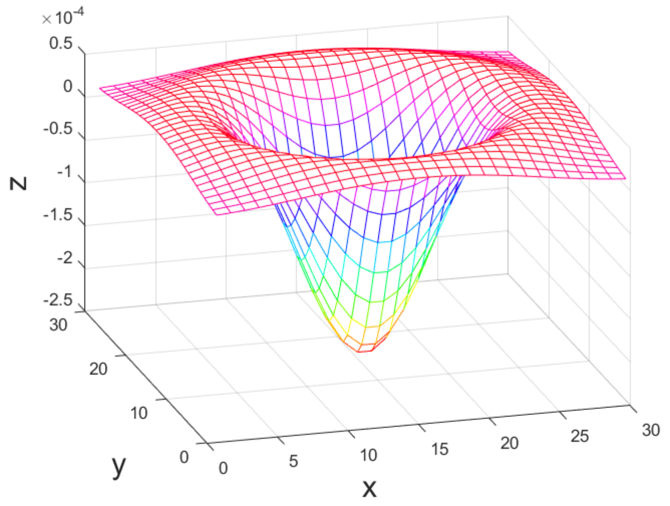
$\sigma' = 3\sigma$



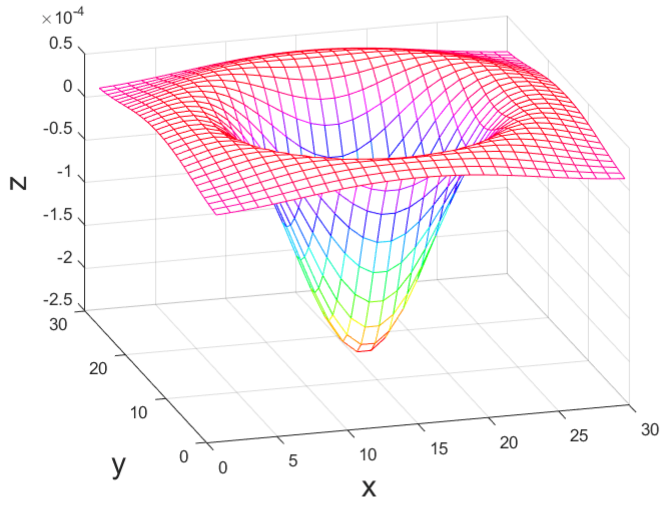


Searching over **Scale**-space

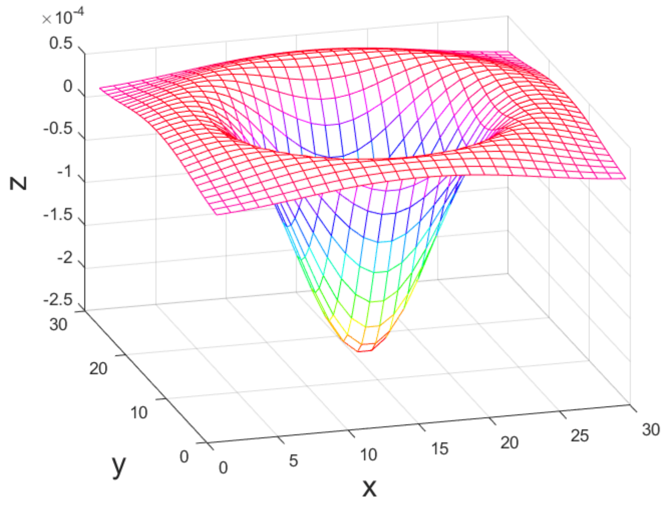
σ

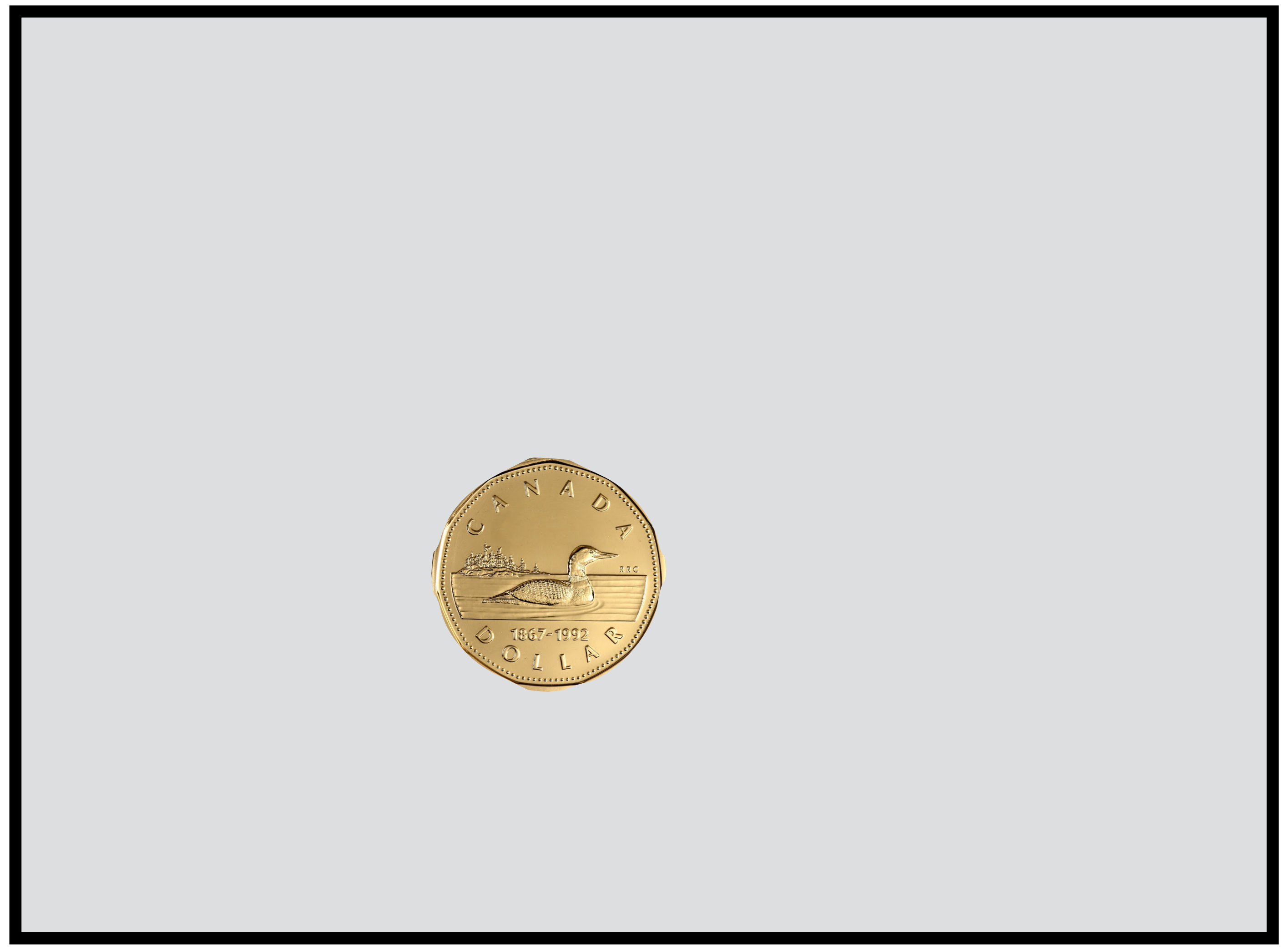


σ



σ

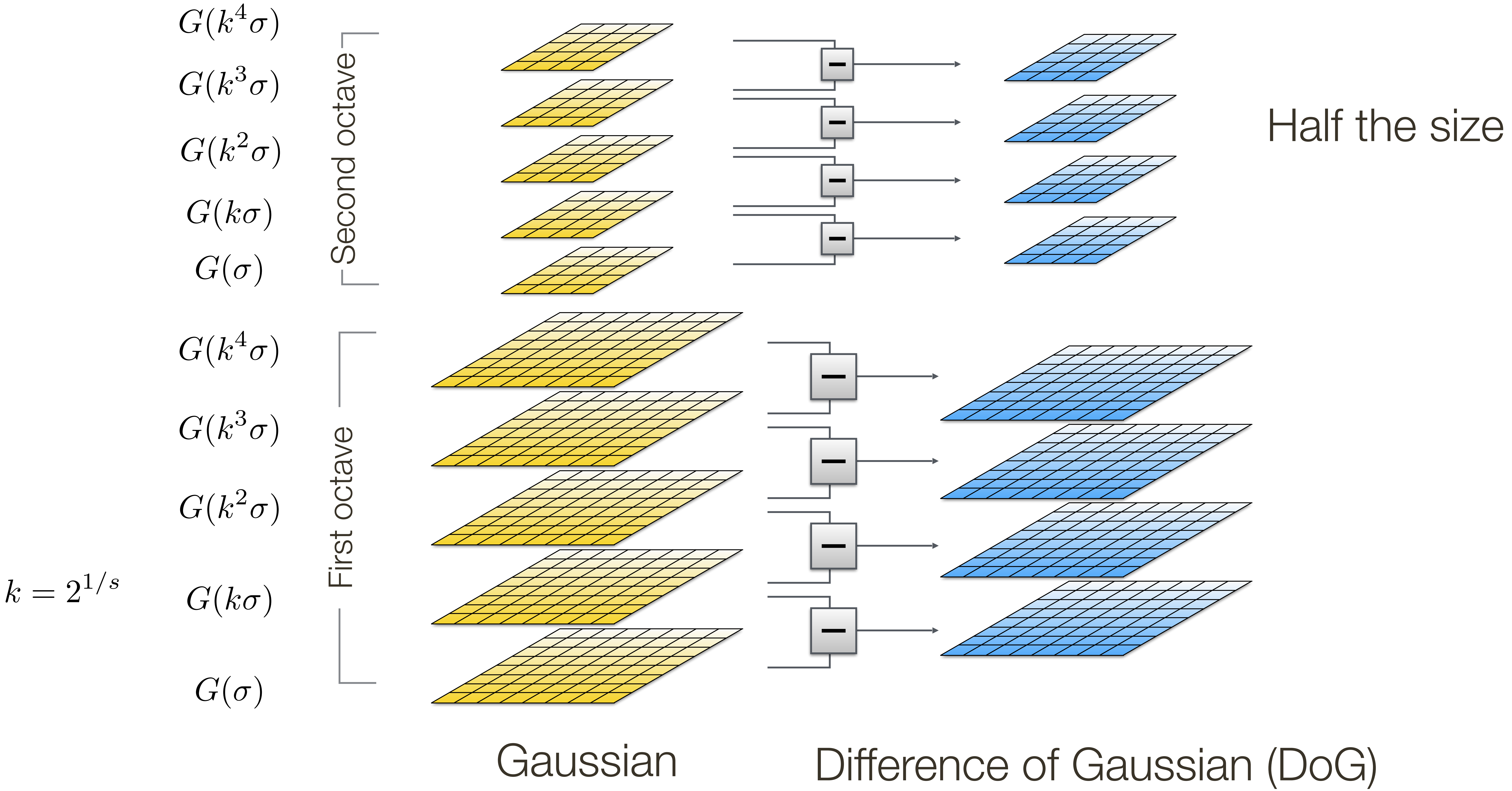




$s = 0.5$

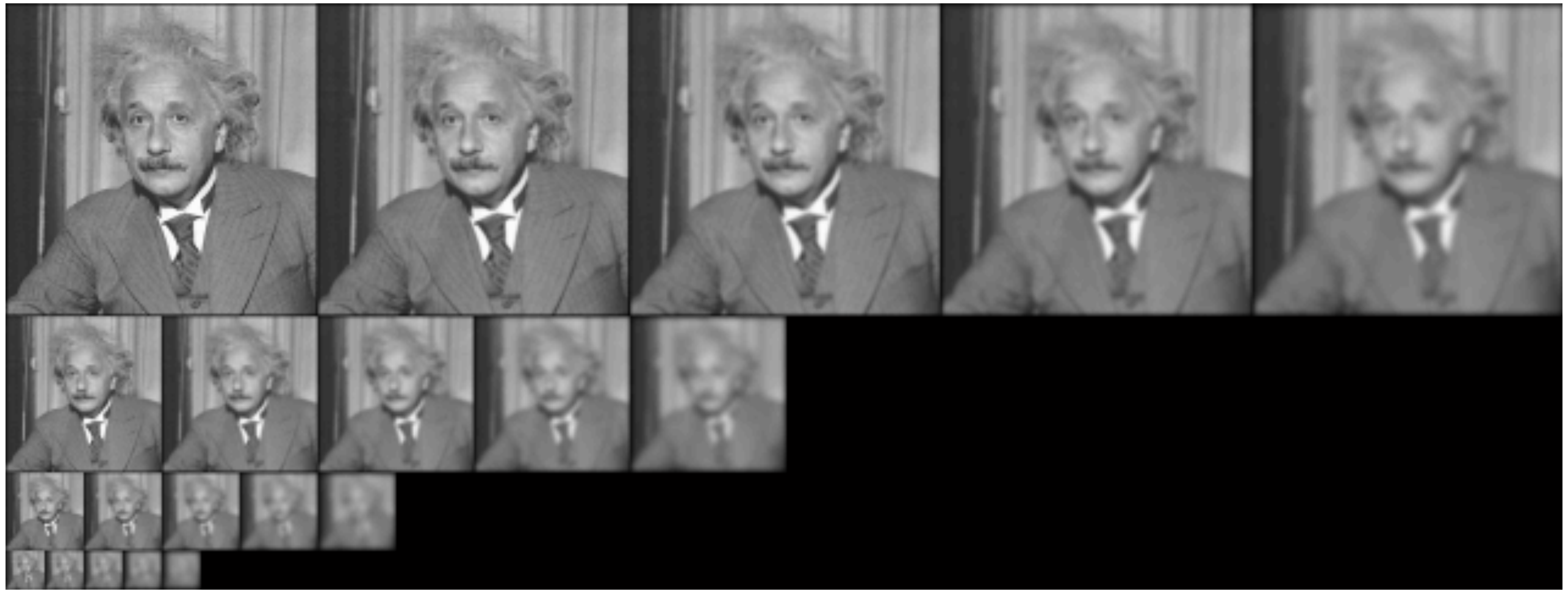
$s = 0.33$

1. Multi-scale Extrema Detection



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

1. Multi-scale Extrema Detection



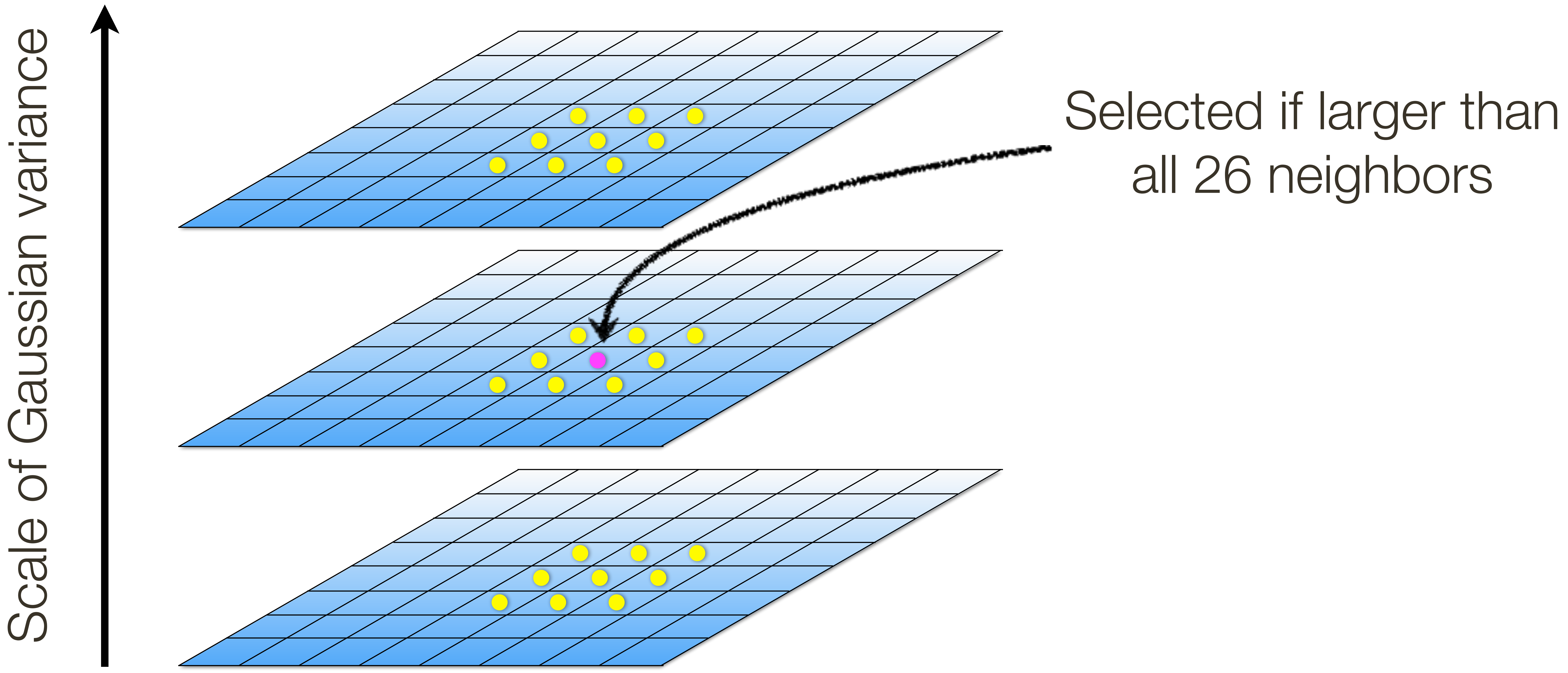
Gaussian

Laplacian

1. Multi-scale Extrema Detection

Detect maxima and minima of Difference of Gaussian in scale space

$$\sigma = 2^{1/s}$$

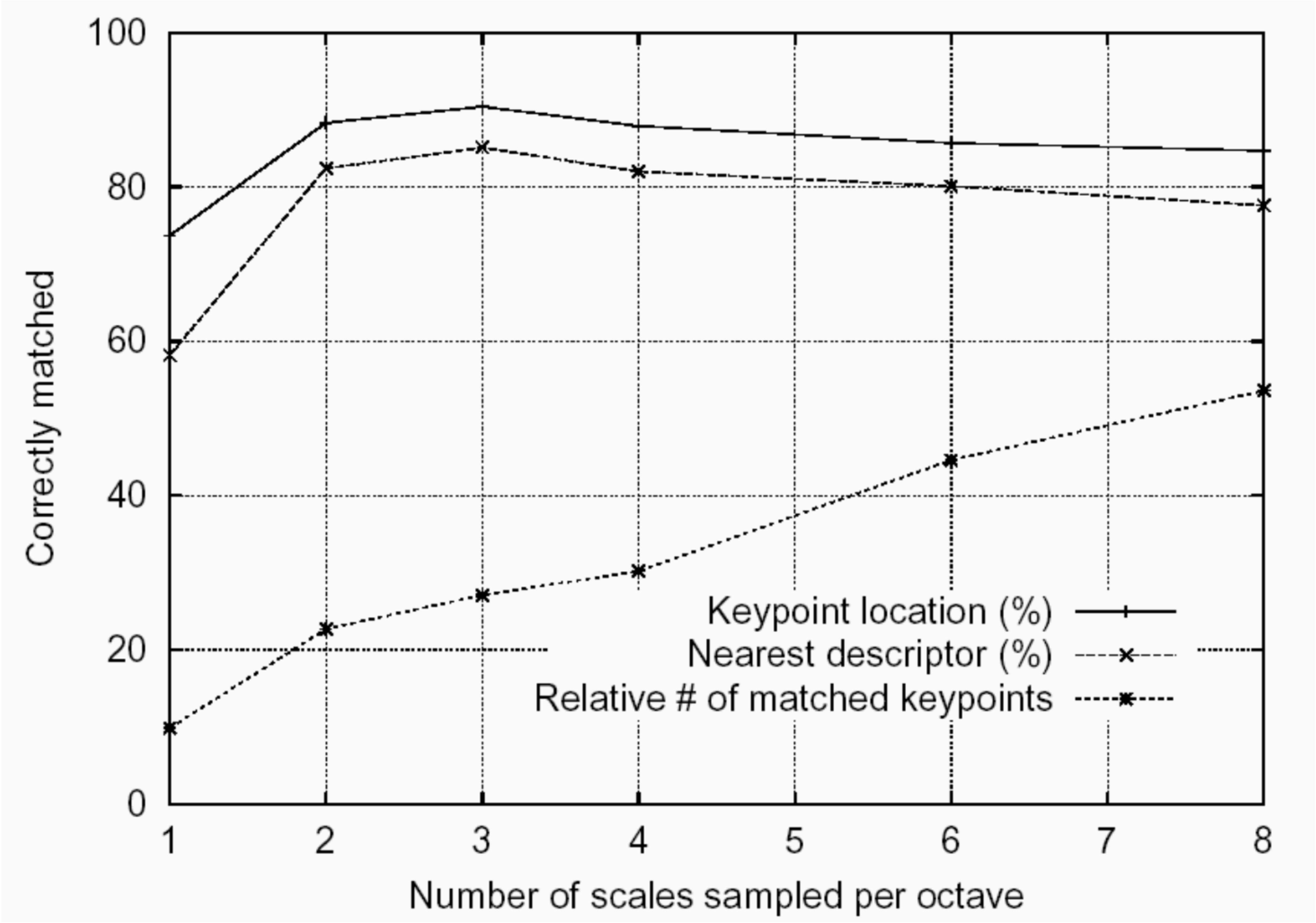


Difference of Gaussian (DoG)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

1. Multi-scale Extrema Detection — Sampling Frequency

More points are found as sampling frequency increases, but accuracy of matching decreases after 3 scales/octave



2. Keypoint Localization

— After keypoints are detected, we remove those that have **low contrast** or are **poorly localized** along an edge

2. Keypoint Localization

— After keypoints are detected, we remove those that have **low contrast** or are **poorly localized** along an edge

How do we decide whether a keypoint is poorly localized, say along an edge, vs. well-localized?

2. Keypoint Localization

— After keypoints are detected, we remove those that have **low contrast** or are **poorly localized** along an edge

How do we decide whether a keypoint is poorly localized, say along an edge, vs. well-localized?

$$C = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

2. Keypoint Localization

— After keypoints are detected, we remove those that have **low contrast** or are **poorly localized** along an edge

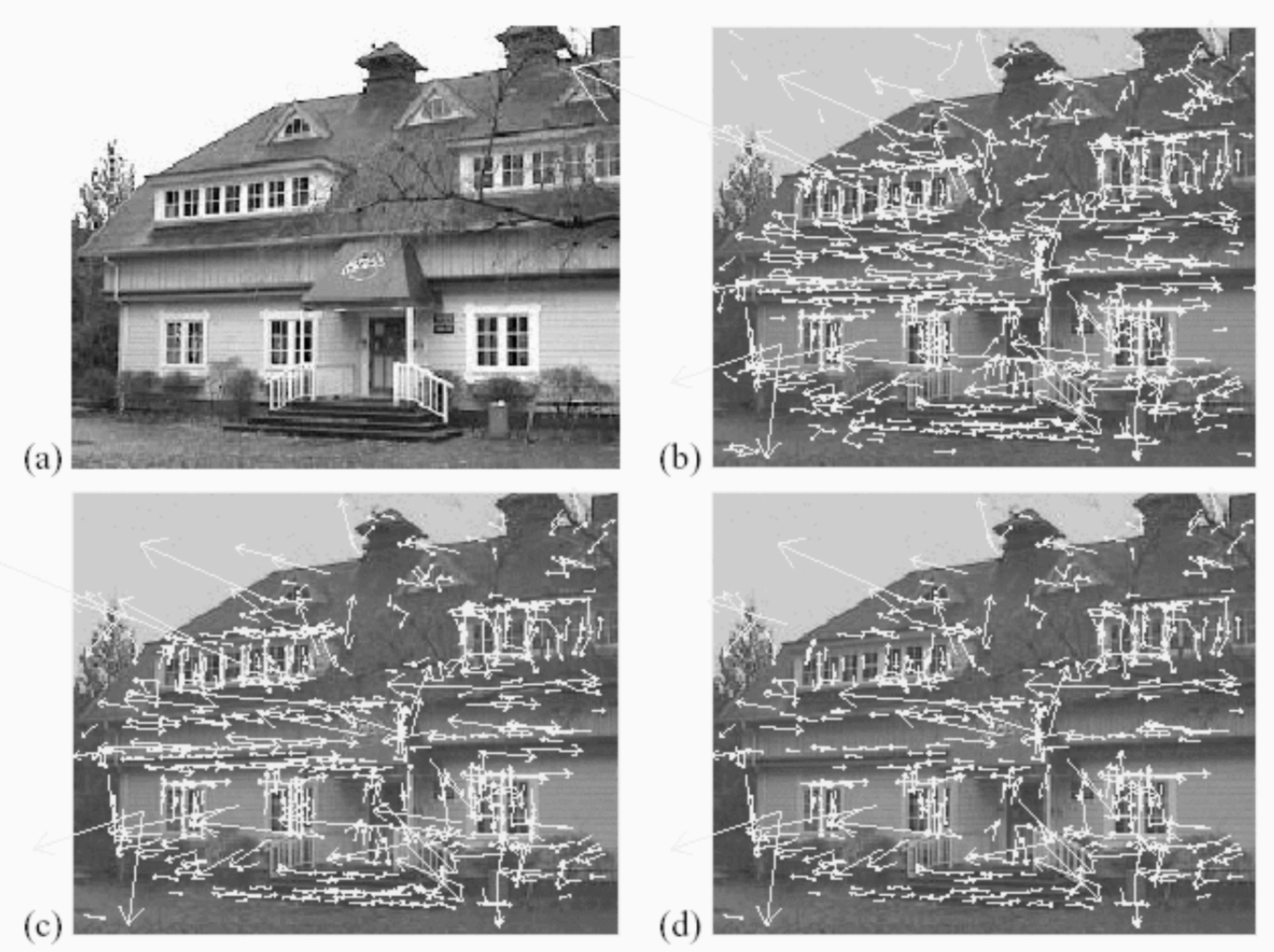
How do we decide whether a keypoint is poorly localized, say along an edge, vs. well-localized?

— Lowe suggests computing the ratio of the eigenvalues of **C** (recall Harris corners) and checking if it is greater than a threshold

— Aside: The ratio can be computed efficiently in fewer than 20 floating point operations, using a trick involving the trace and determinant of **C** - no need to explicitly compute the eigenvalues

2. Keypoint Localization

Example:



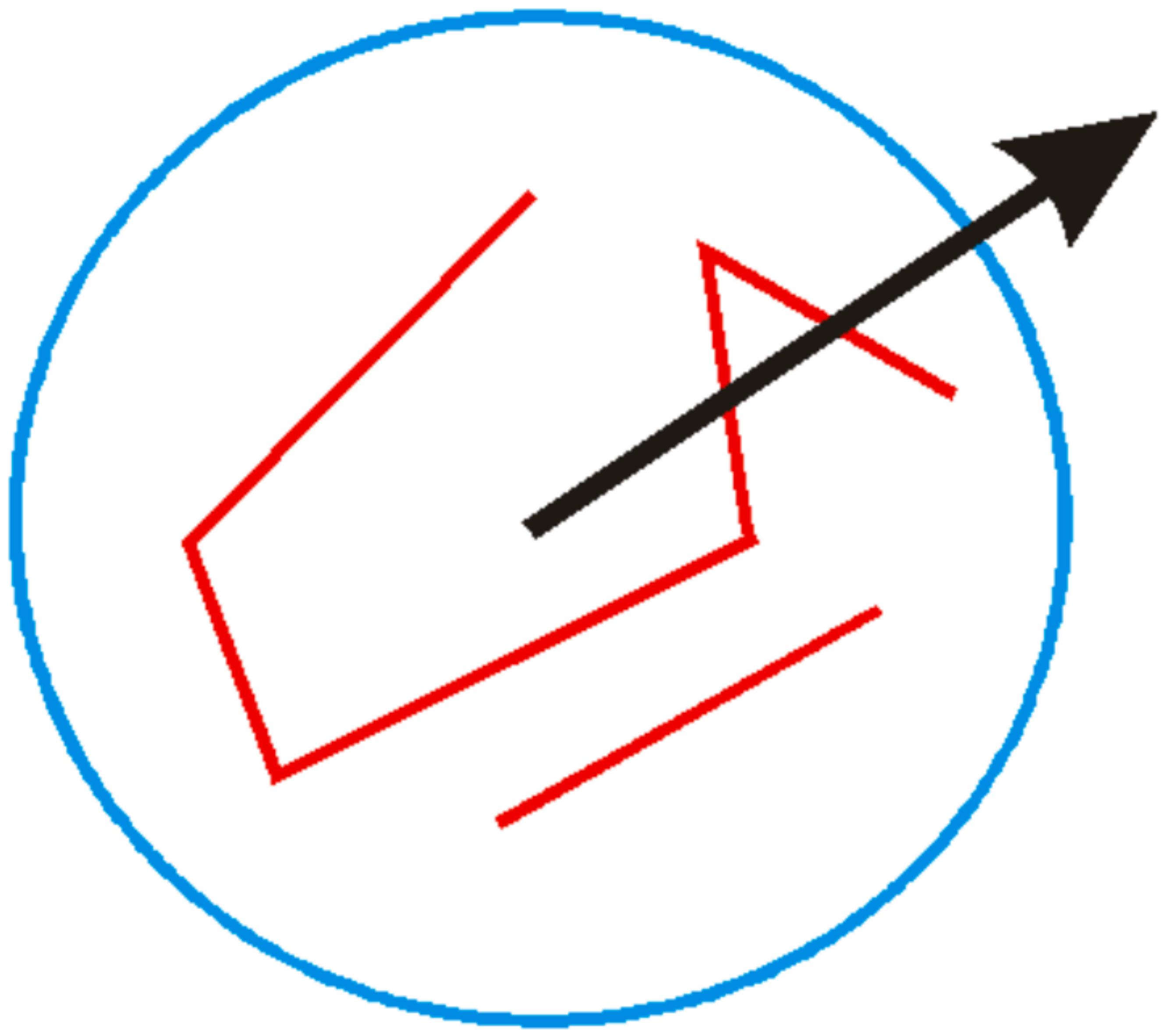
- (a) 233×189 image
- (b) 832 DOG extrema
- (c) 729 left after peak value threshold
- (d) 536 left after testing ratio of principal curvatures

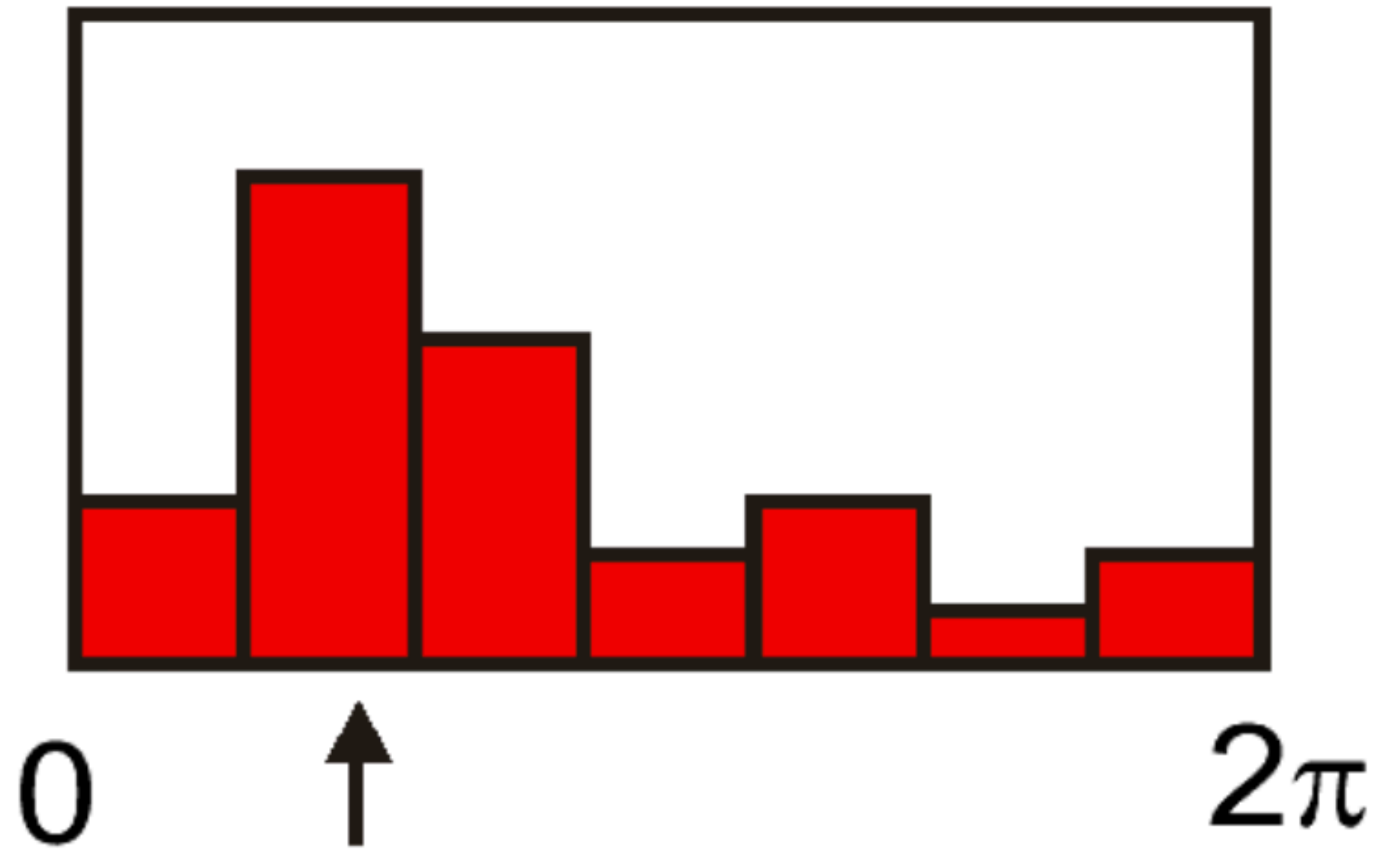
3. Orientation Assignment

— Create **histogram** of local gradient directions computed at selected scale

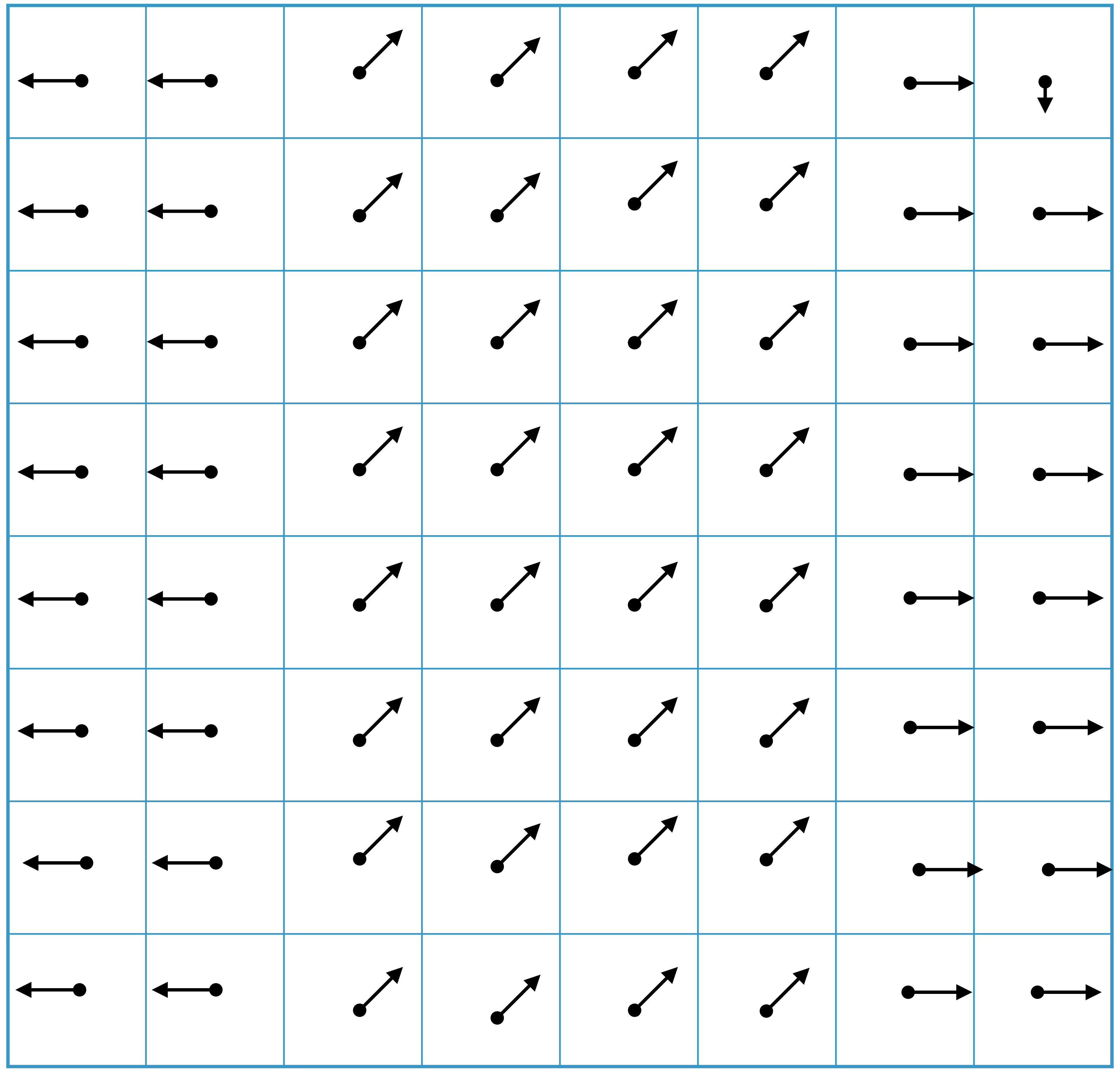
— Assign **canonical orientation** at peak of smoothed histogram

— Each key specifies stable 2D coordinates (x , y , scale, orientation)



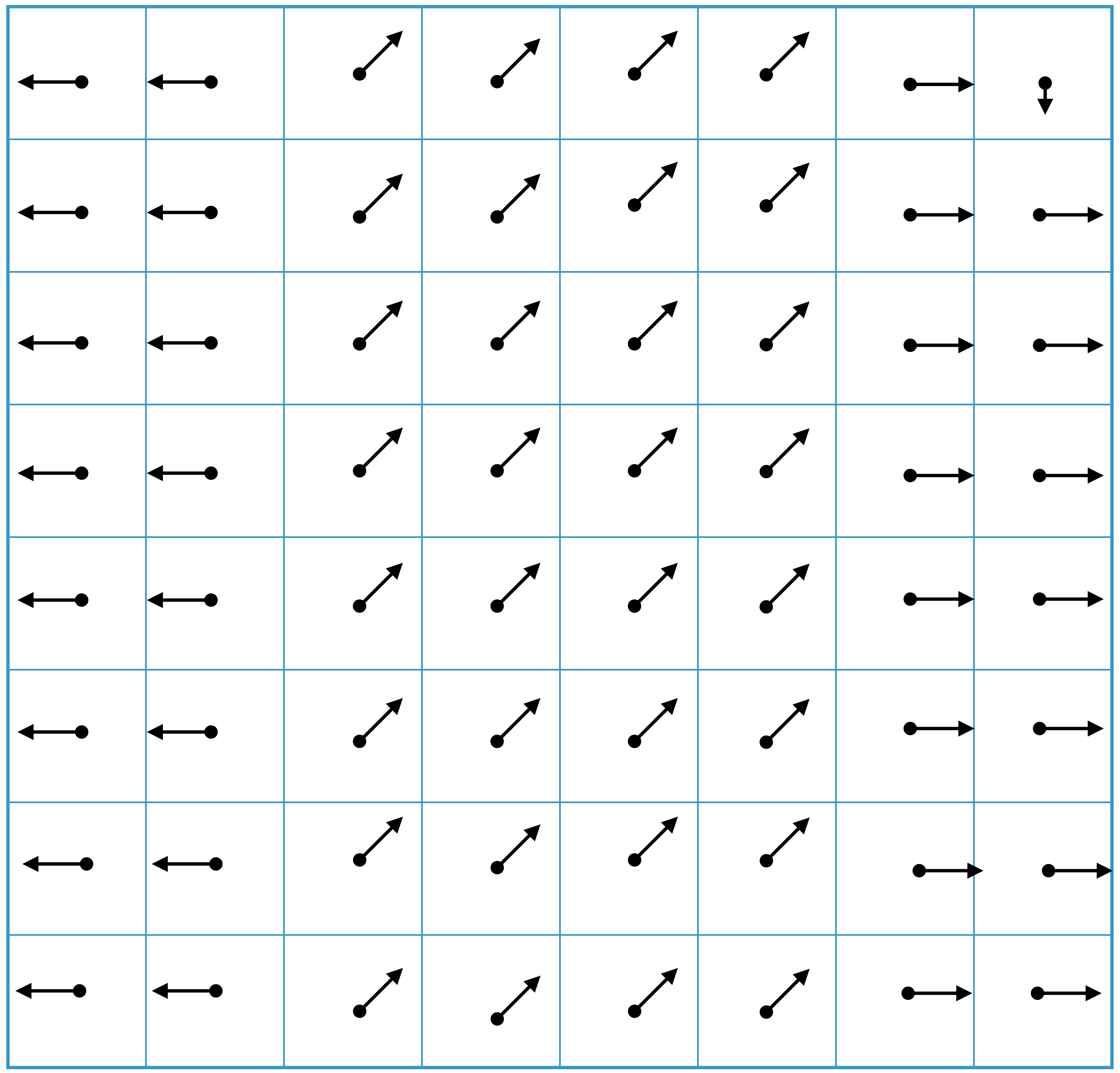


3. Orientation Assignment

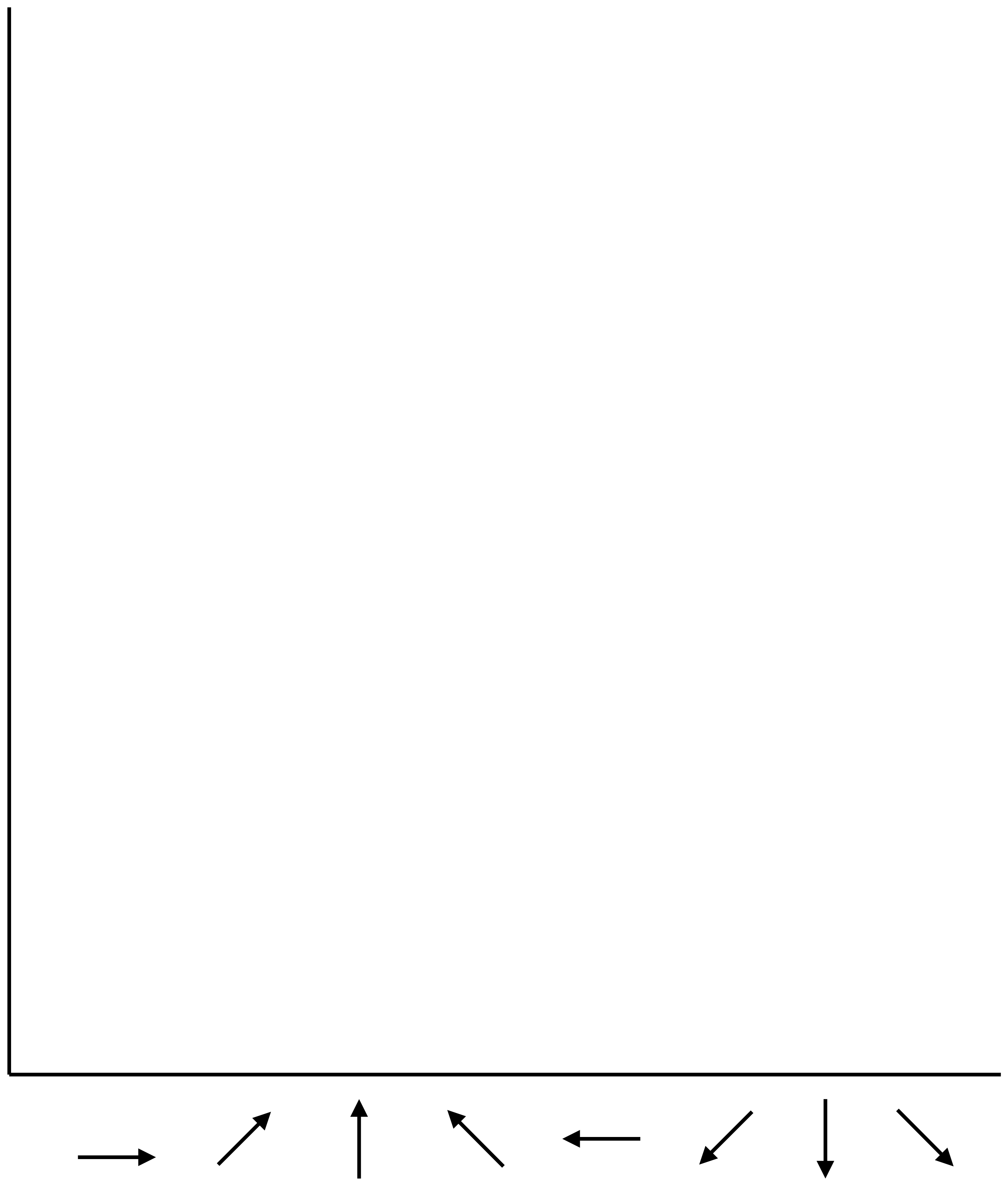


Arrows illustrate **gradient orientation** (direction) and **gradient magnitude** (arrow length)

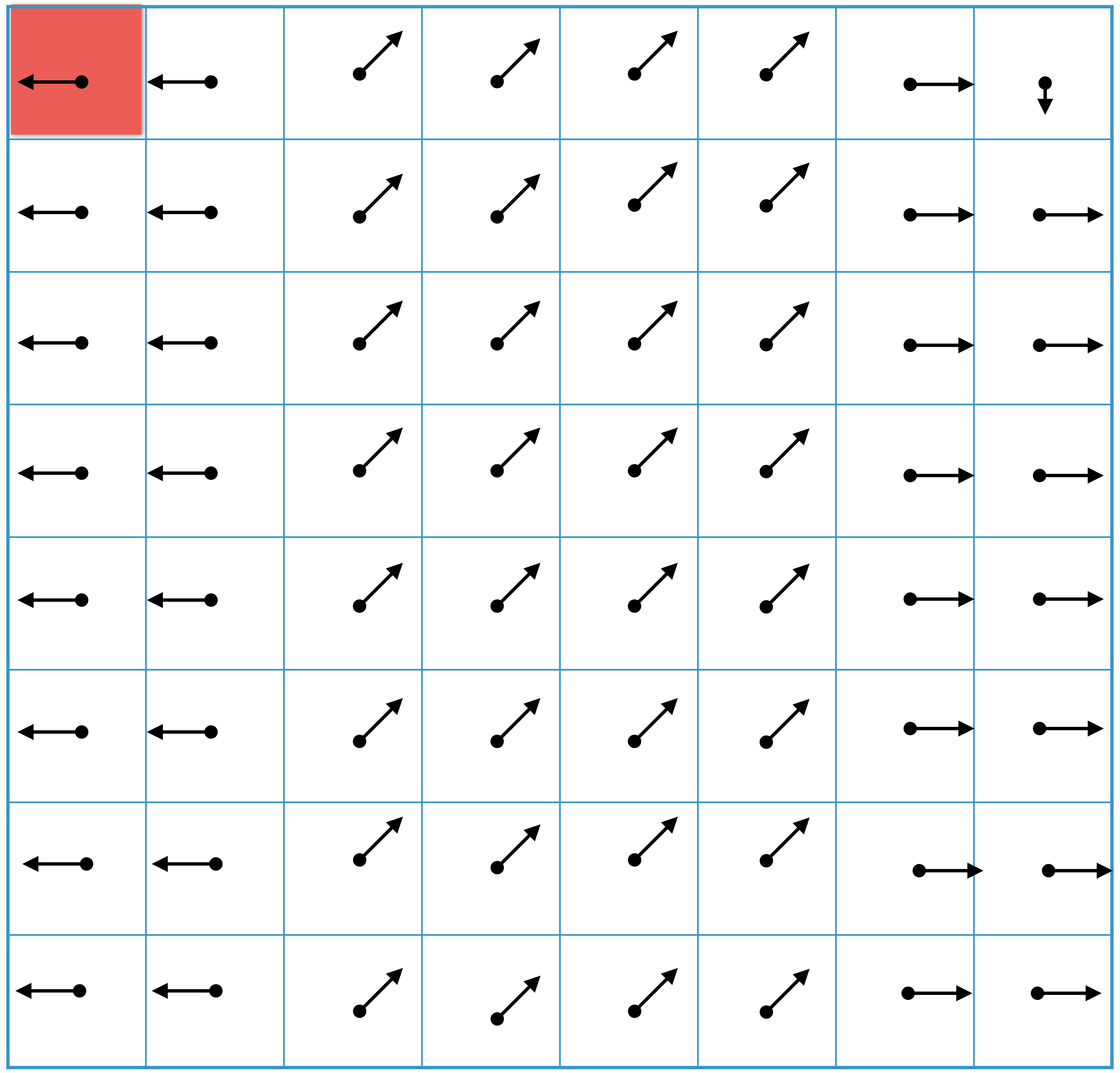
3. Orientation Assignment



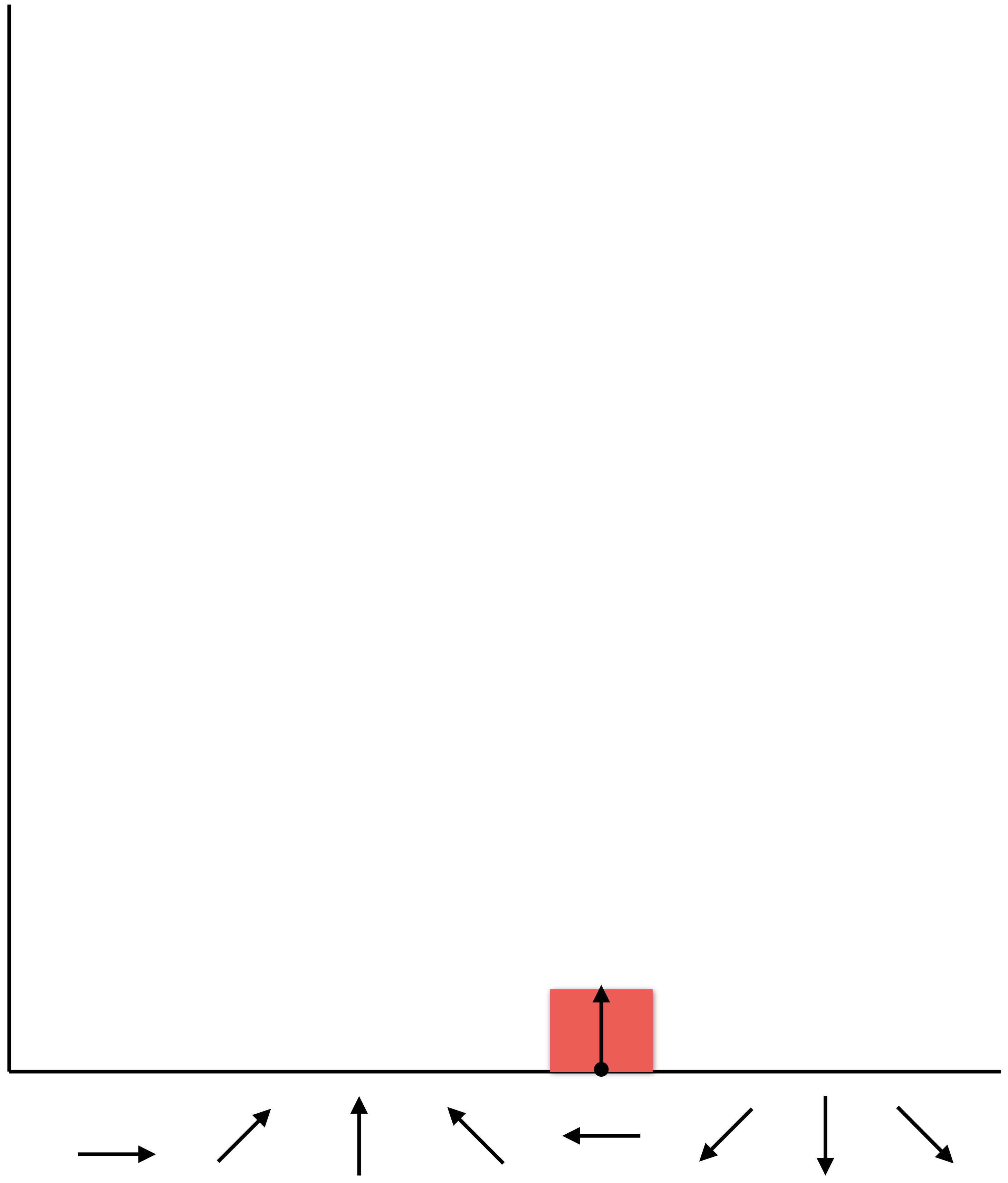
Arrows illustrate **gradient orientation** (direction) and **gradient magnitude** (arrow length)



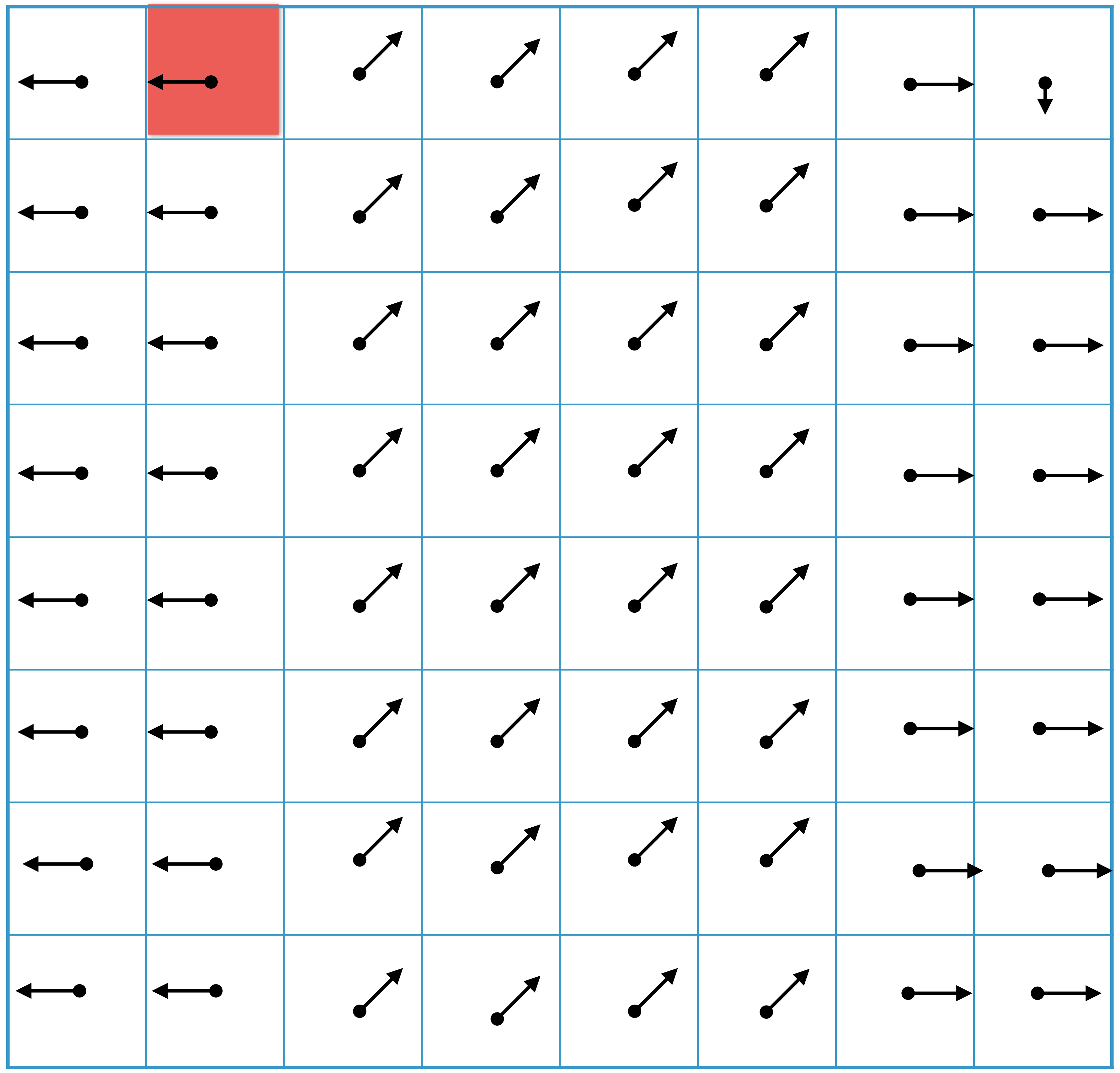
3. Orientation Assignment

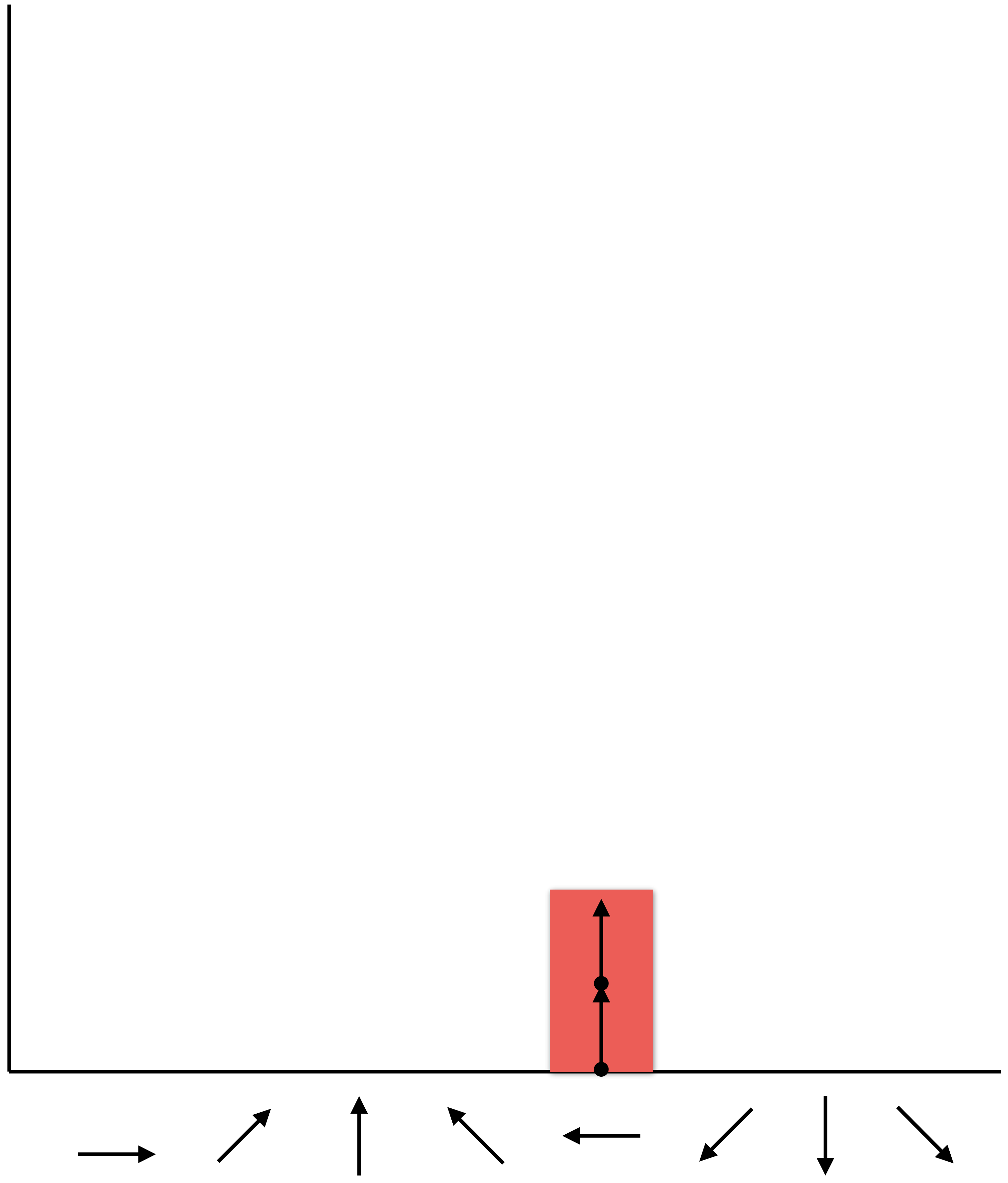


Arrows illustrate **gradient orientation** (direction) and **gradient magnitude** (arrow length)



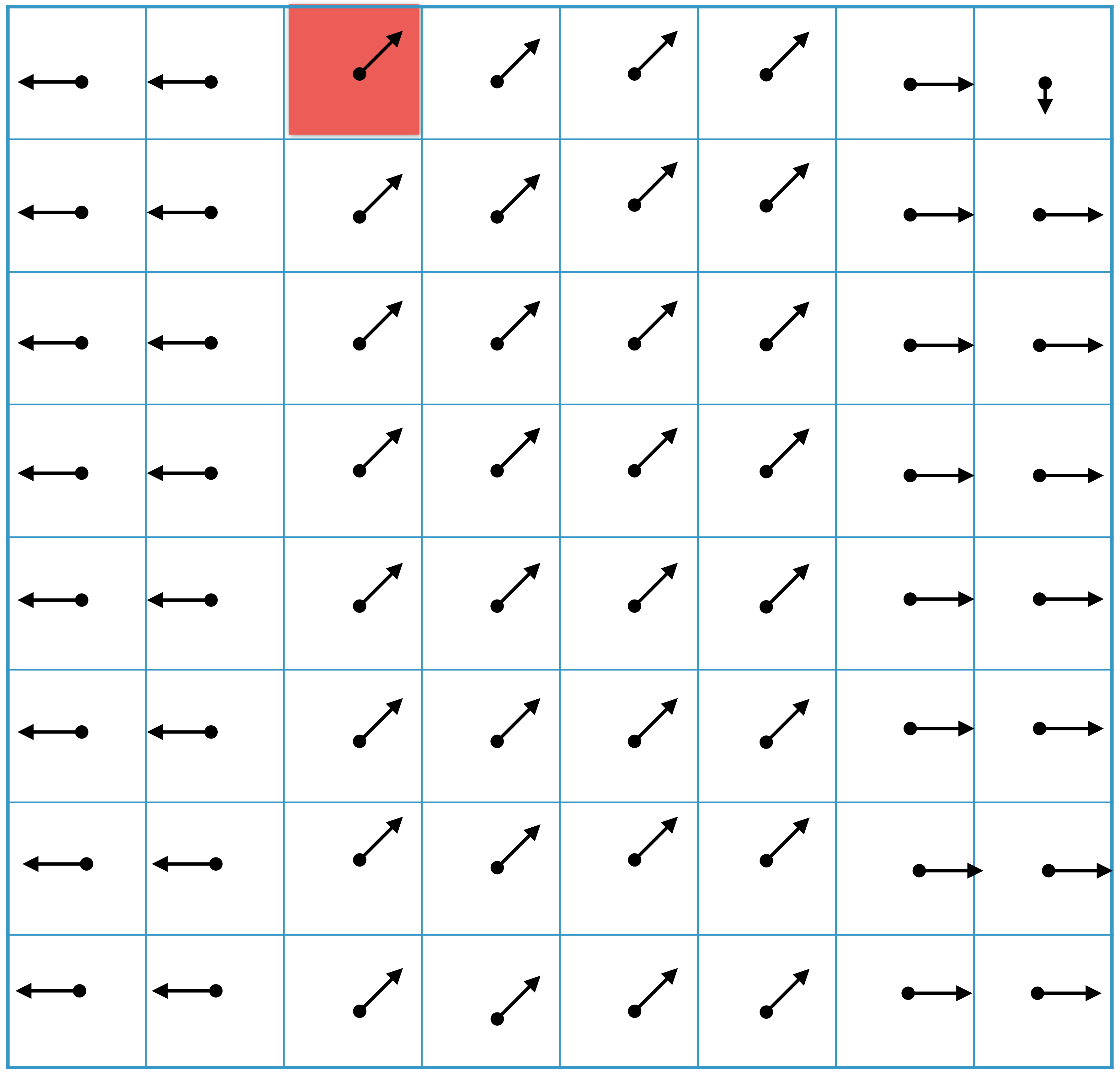
3. Orientation Assignment

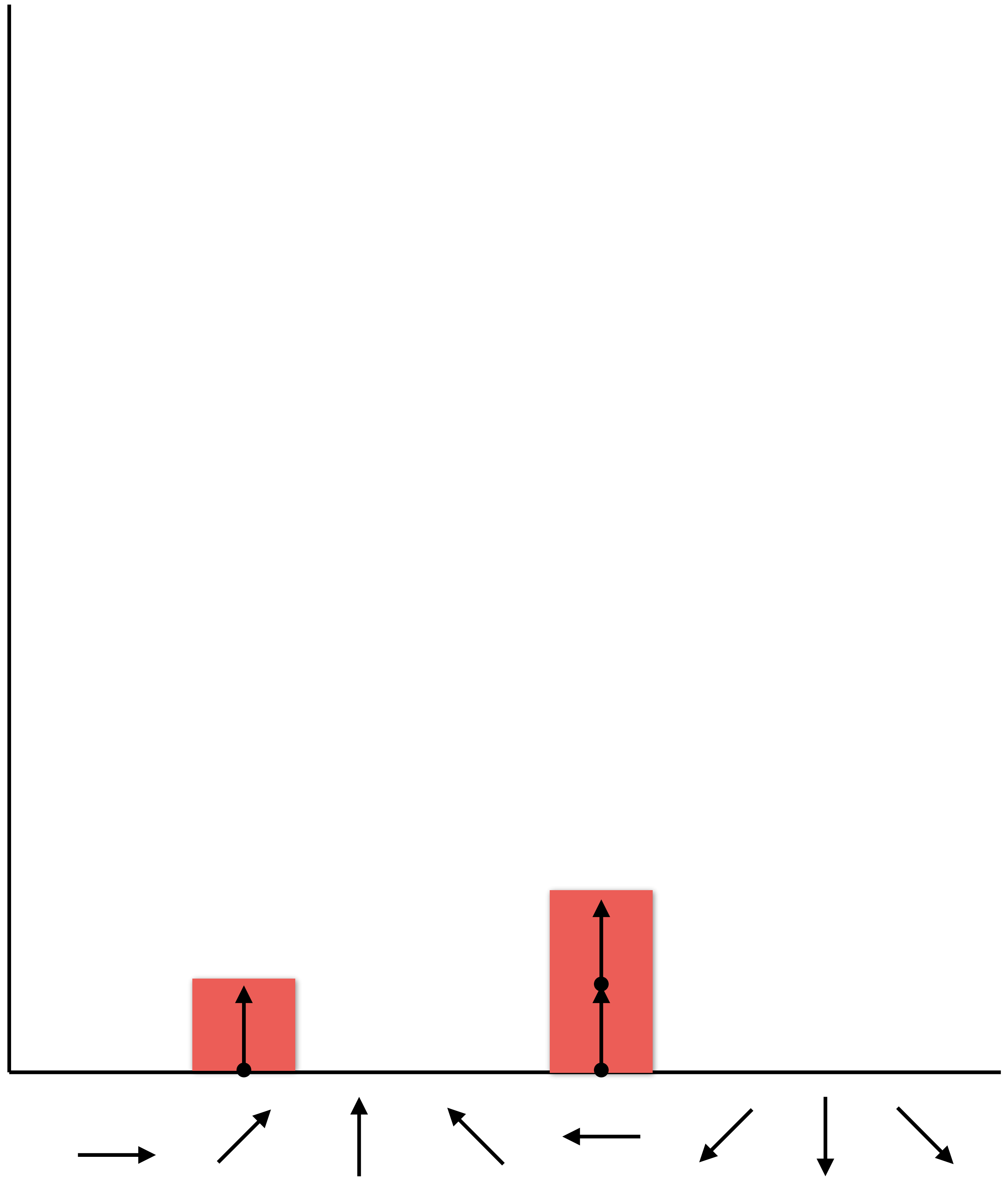




Arrows illustrate **gradient orientation** (direction) and **gradient magnitude** (arrow length)

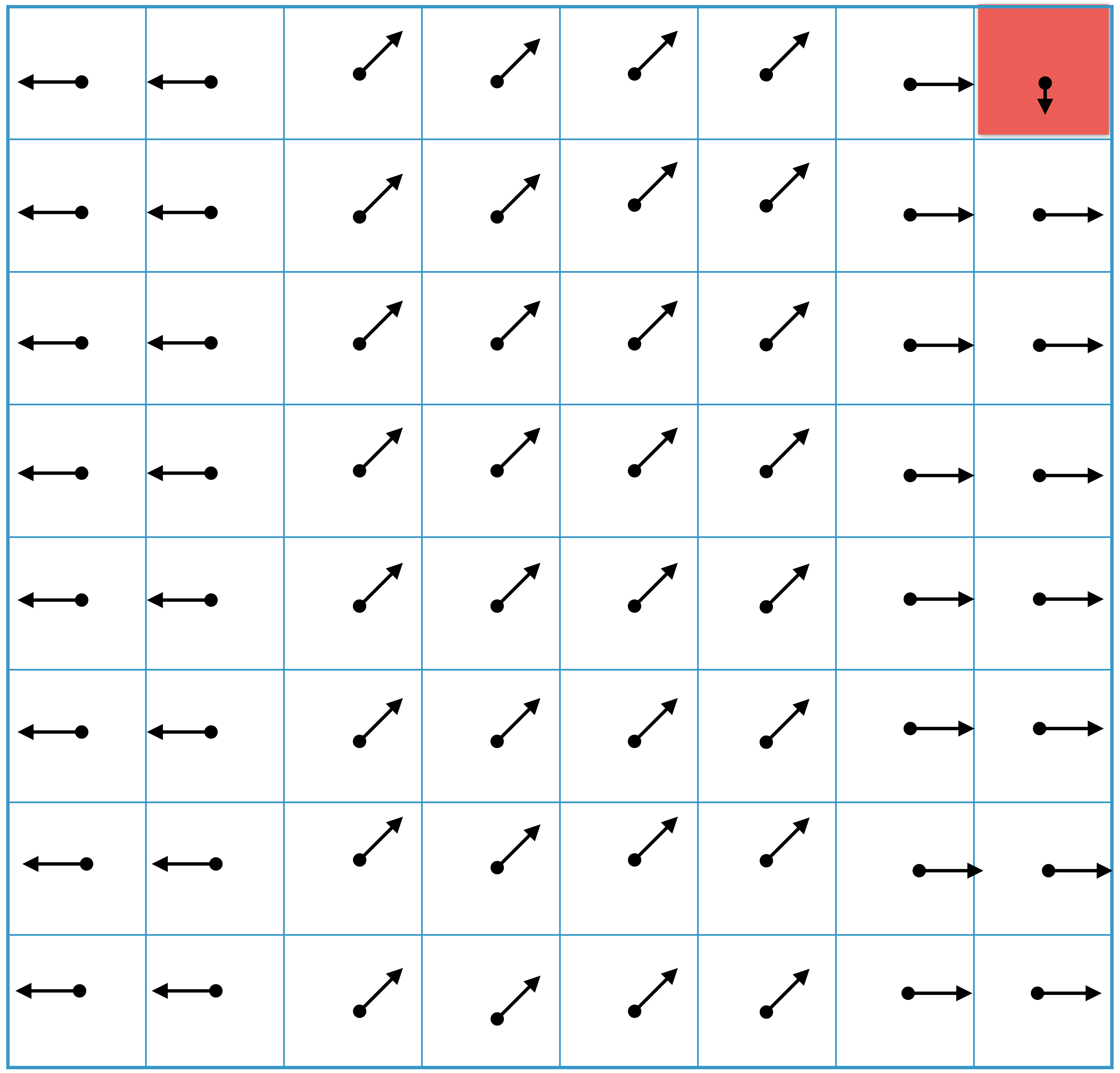
3. Orientation Assignment

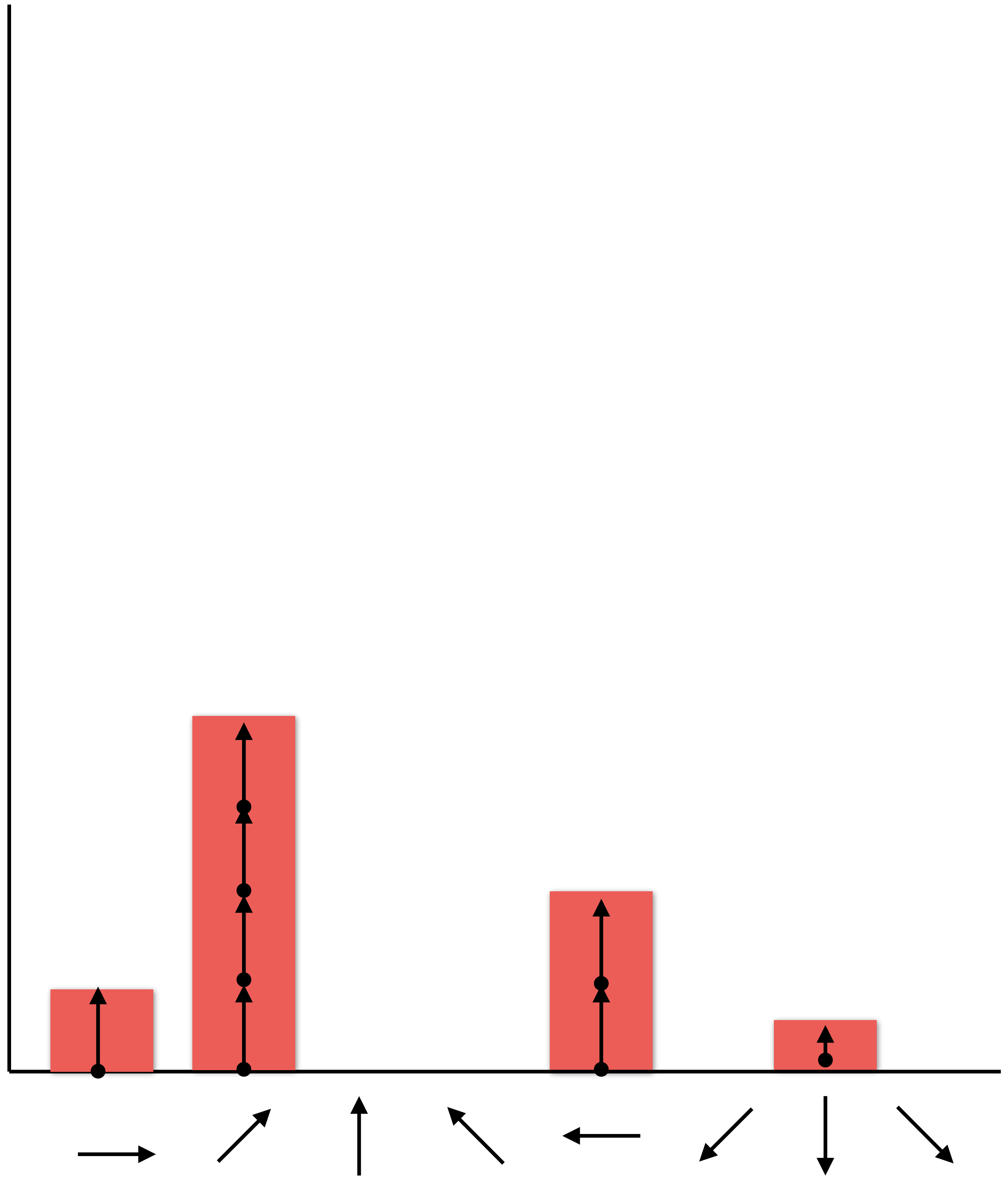




Arrows illustrate **gradient orientation** (direction)
and **gradient magnitude** (arrow length)

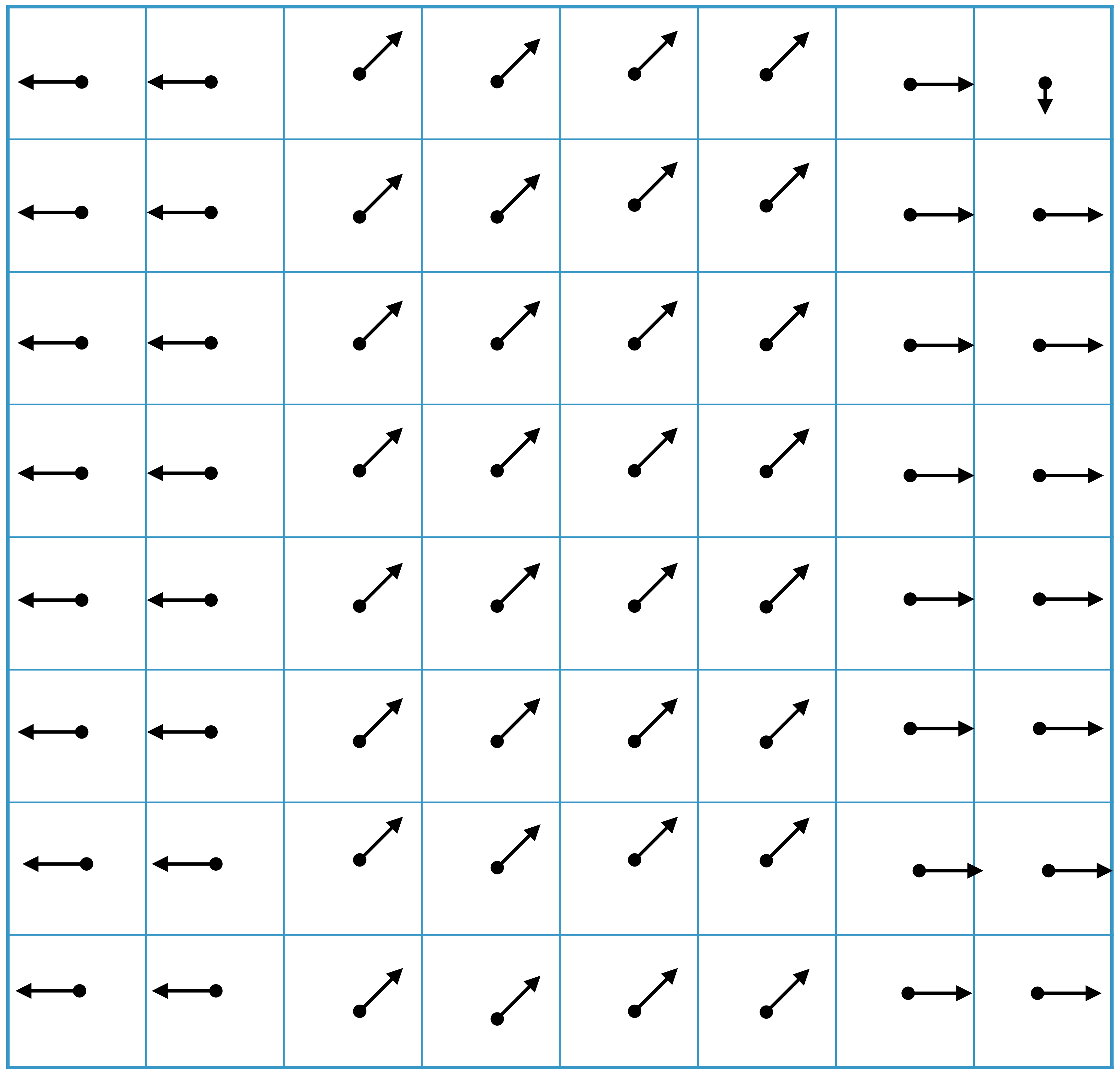
3. Orientation Assignment



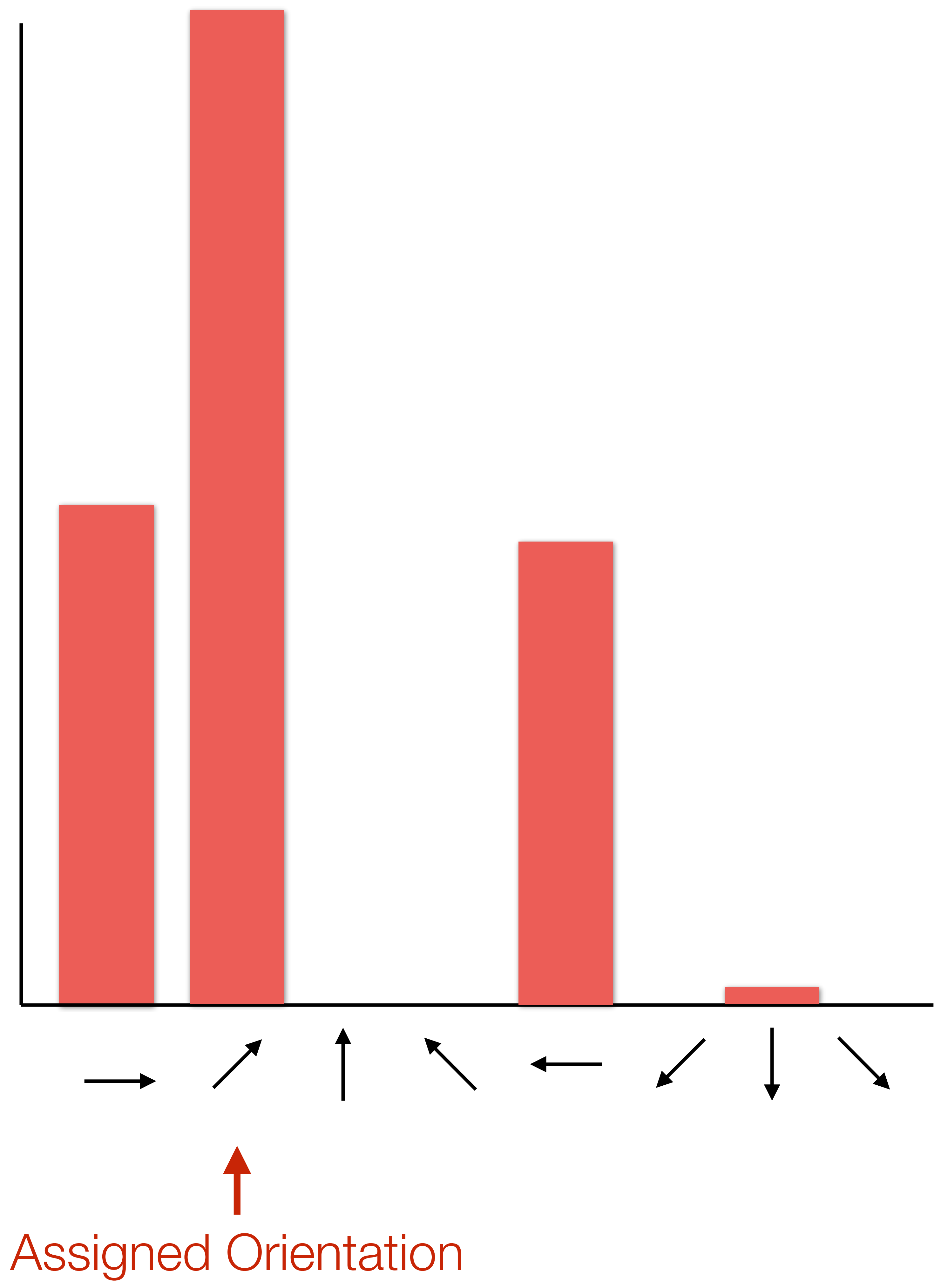


Arrows illustrate **gradient orientation** (direction) and **gradient magnitude** (arrow length)

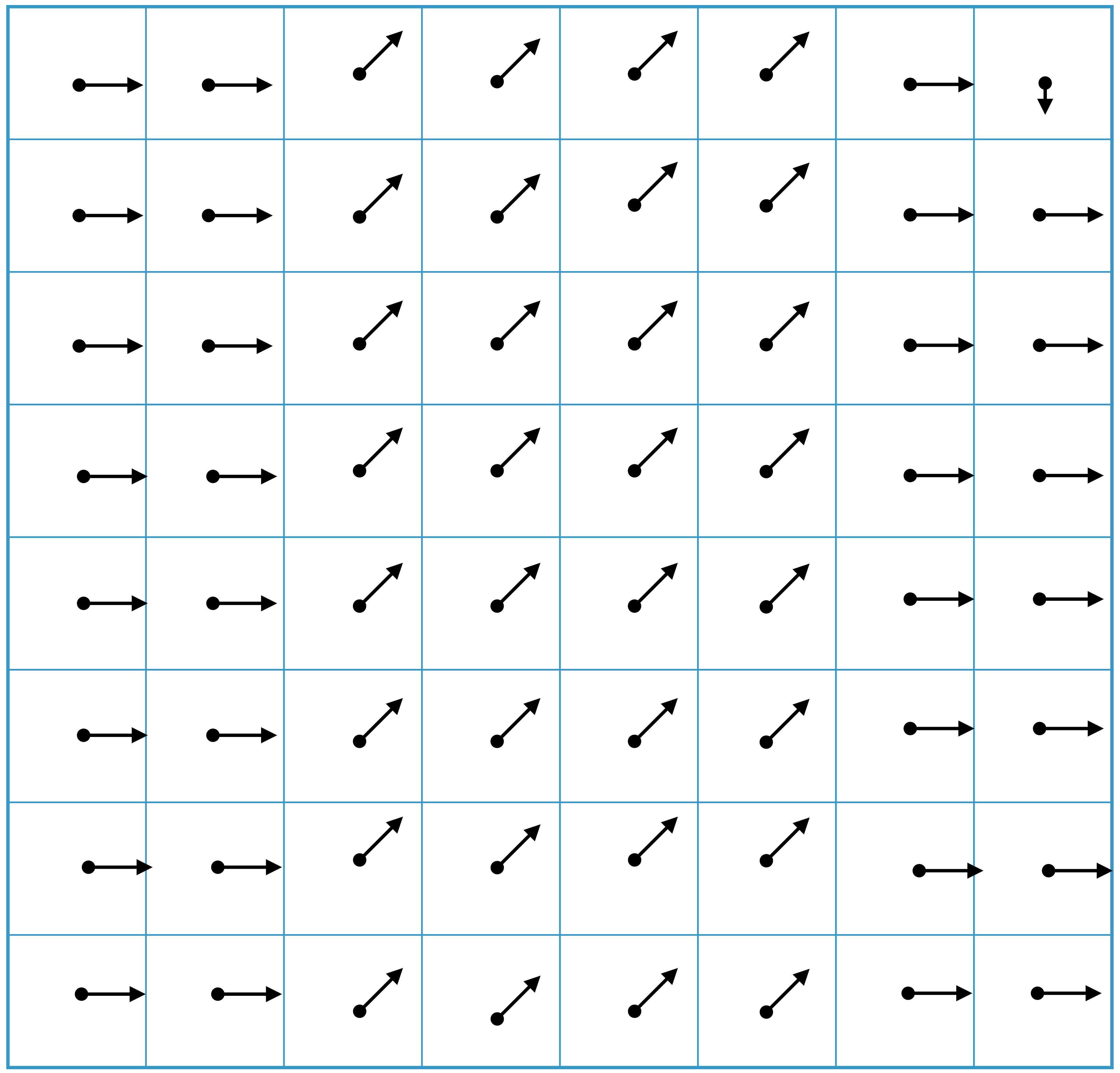
3. Orientation Assignment



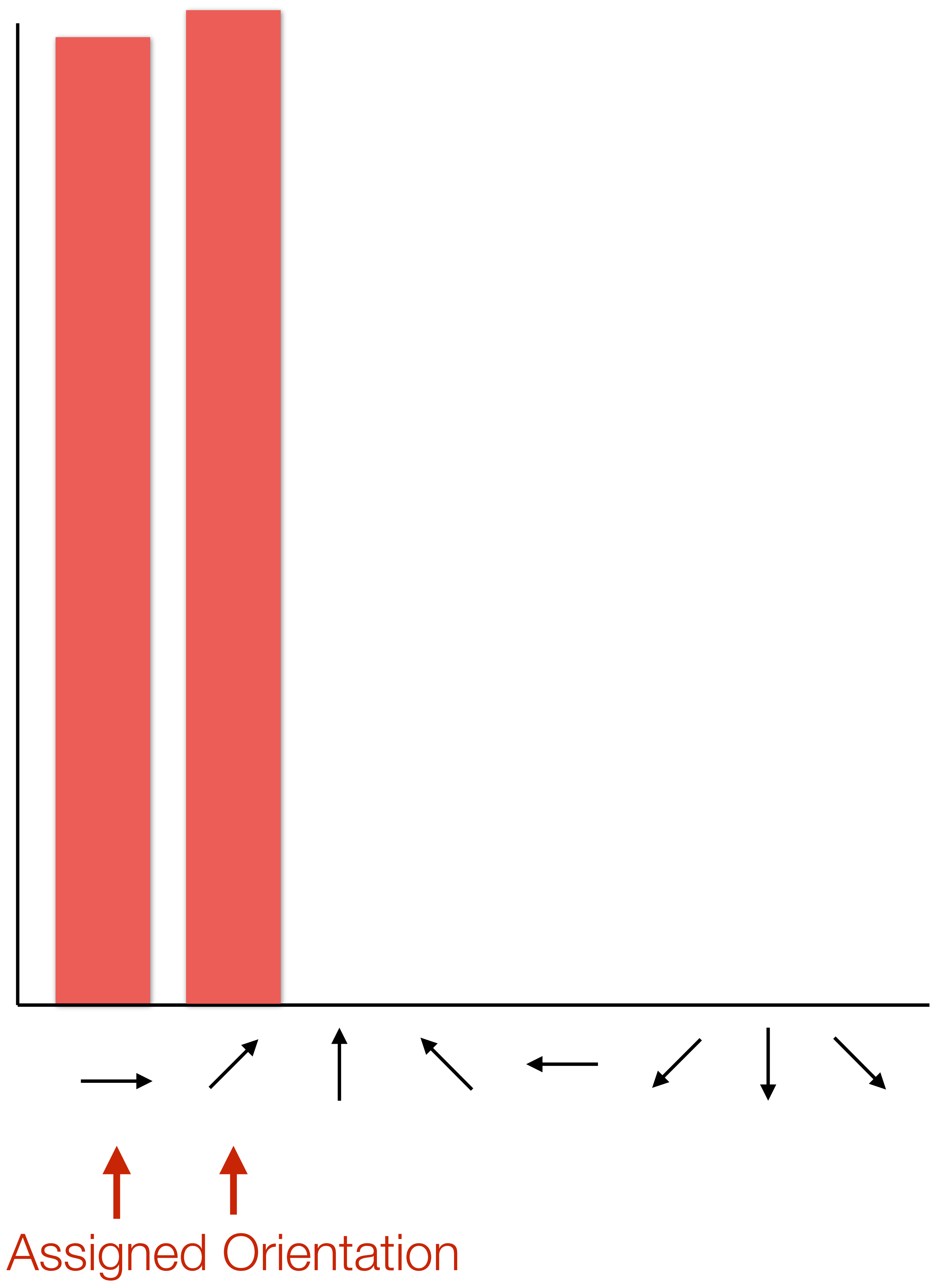
Arrows illustrate **gradient orientation** (direction) and **gradient magnitude** (arrow length)



3. Orientation Assignment

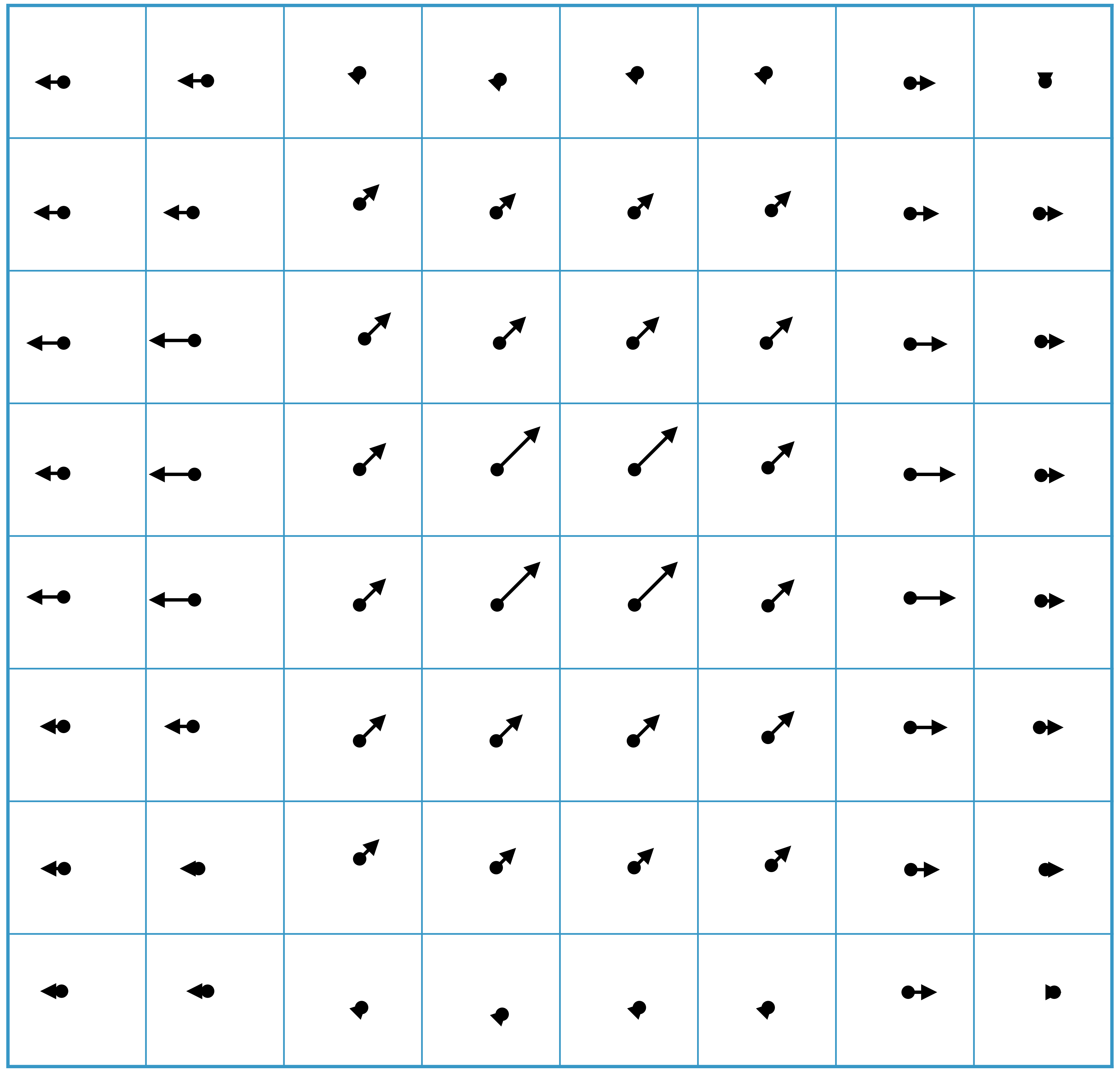


Arrows illustrate **gradient orientation** (direction)
and **gradient magnitude** (arrow length)

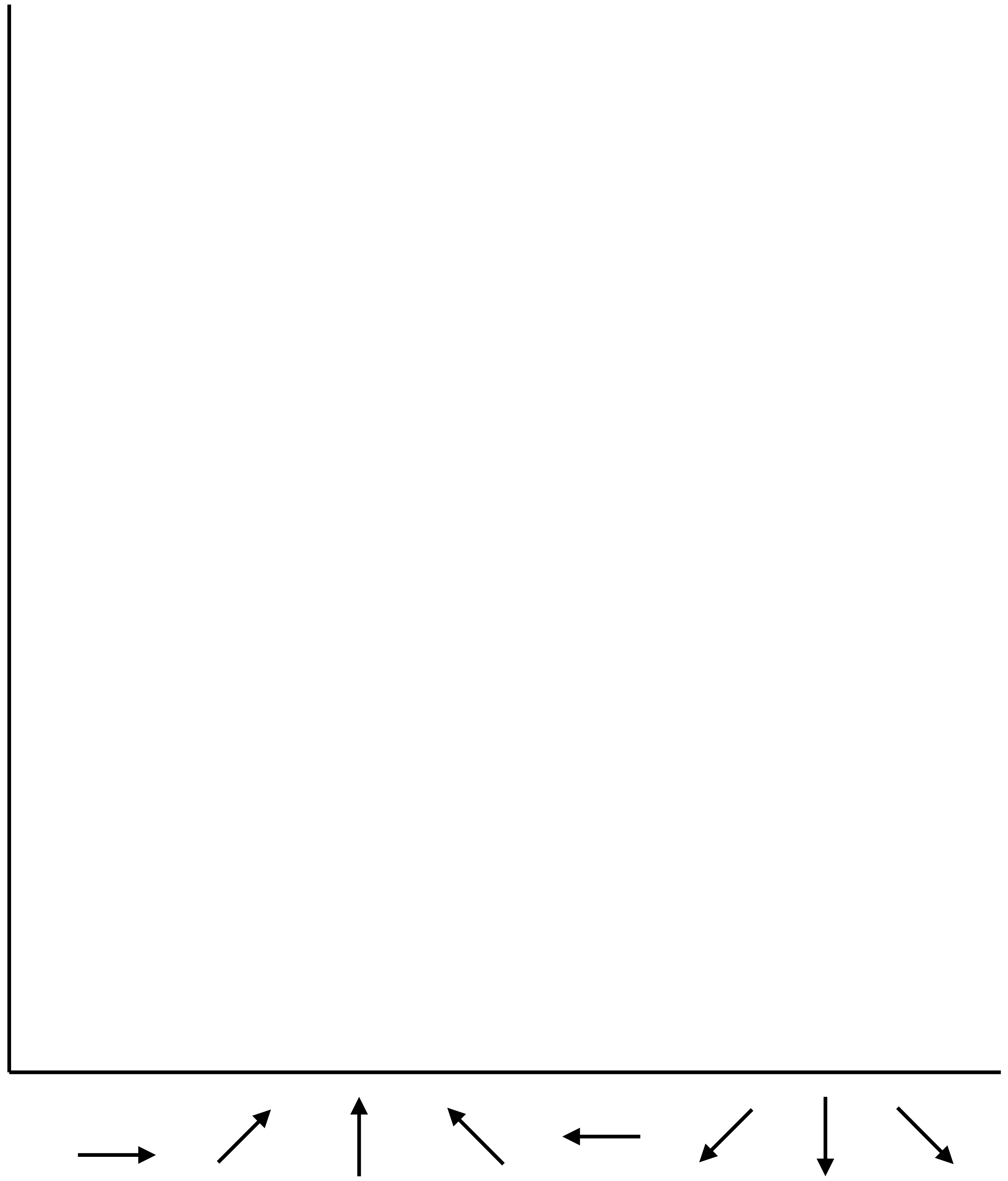


3. Orientation Assignment

Multiply **gradient magnitude** by a **Gaussian** kernel



Arrows illustrate **gradient orientation** (direction) and **gradient magnitude** (arrow length)



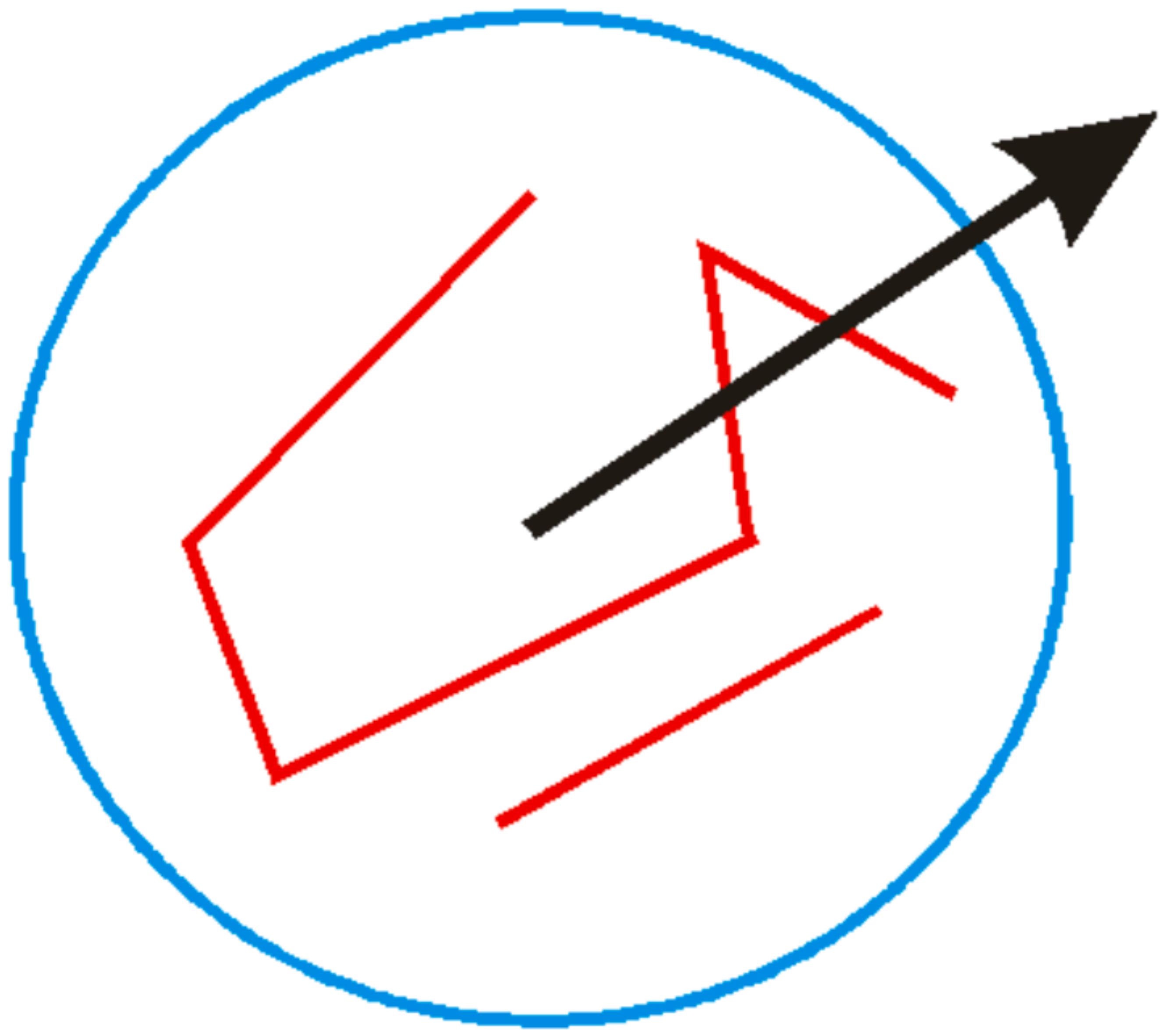
3. Orientation Assignment

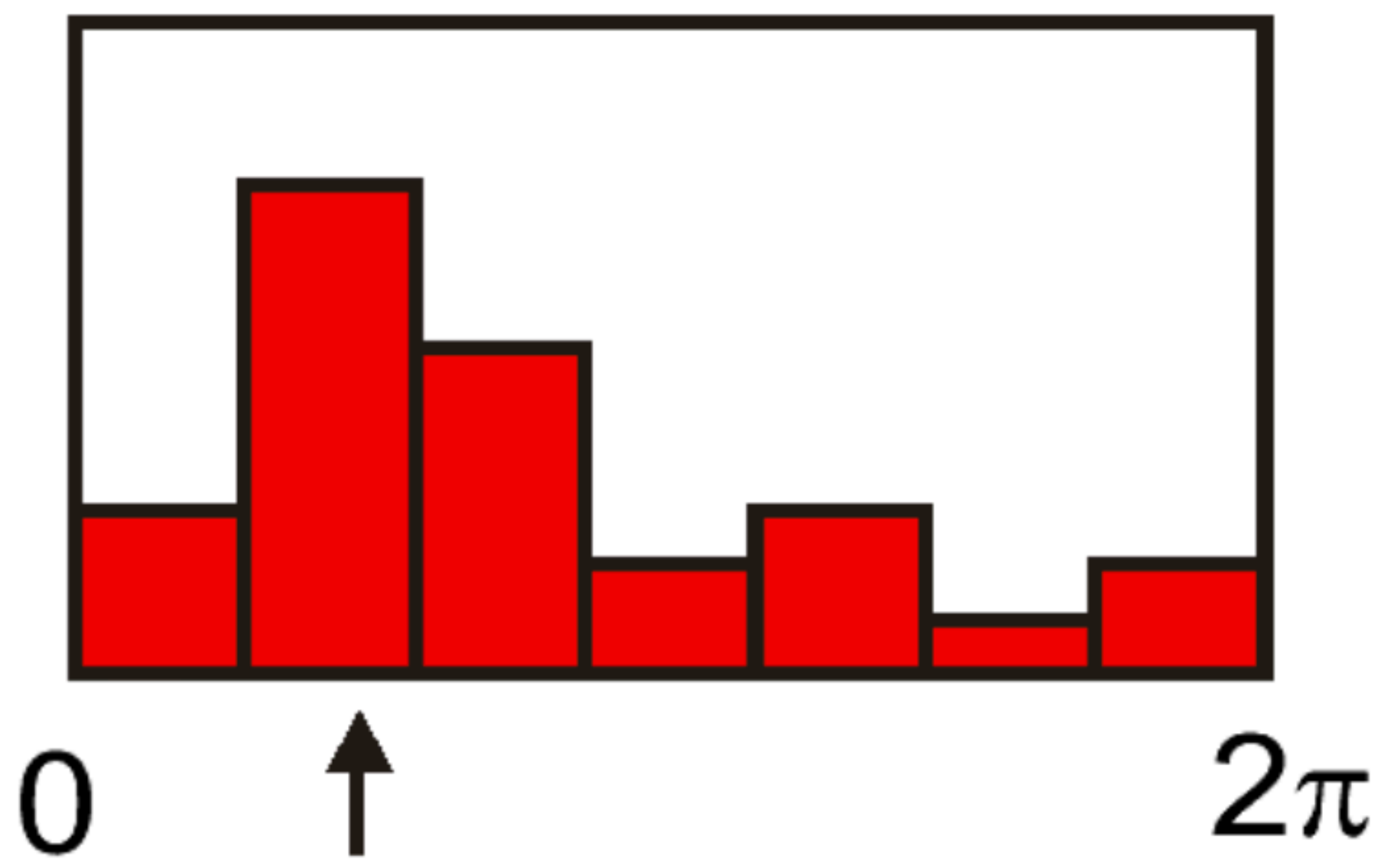
— **Histogram** of 36 bins (10 degree increments)

— Size of the **window** is 1.5 scale (recall the Gaussian filter)

— Gaussian-weighted **voting**

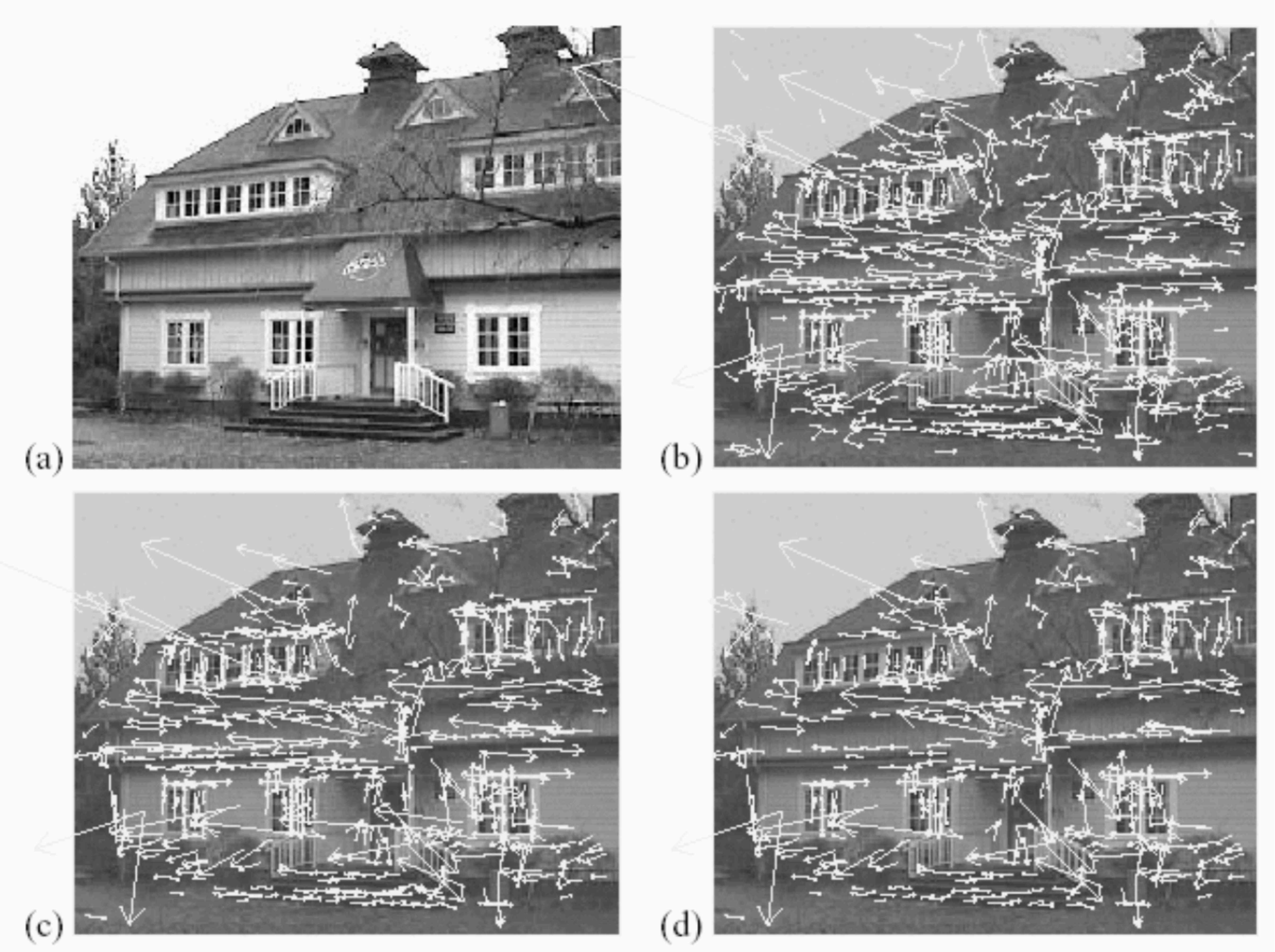
— Highest **peak** and peaks above 80% of highest also considered for calculating dominant orientations





3. Keypoint Localization

Example:



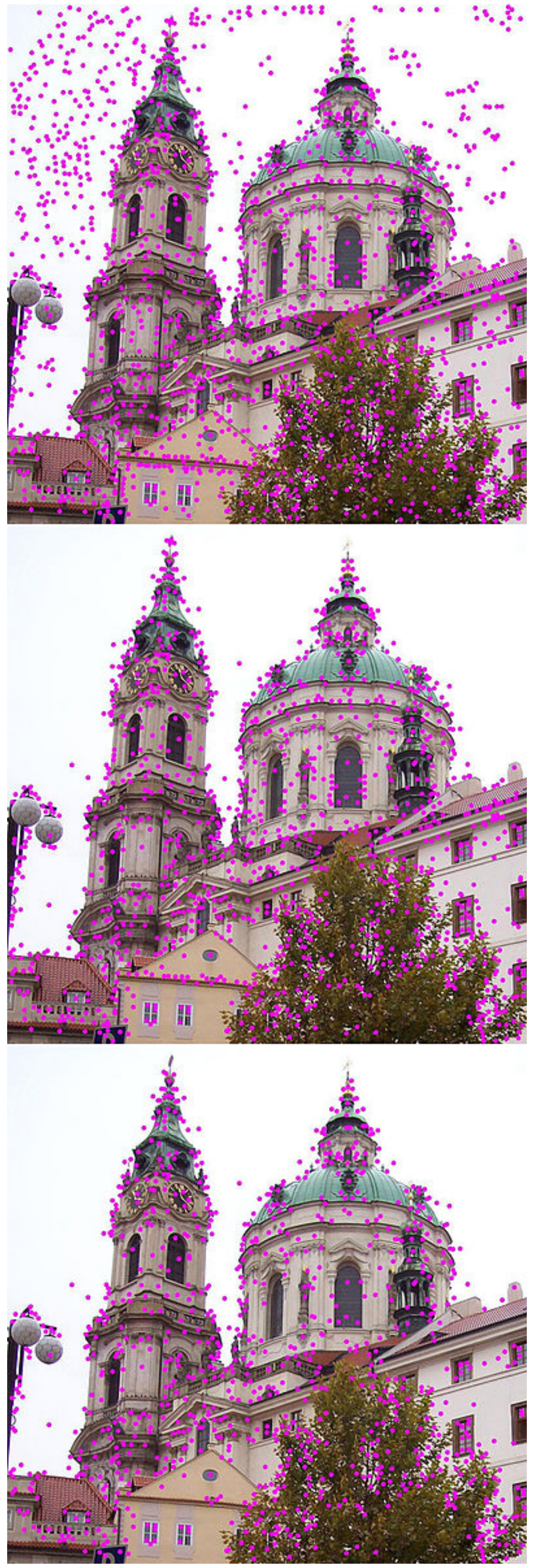
(a) 233 × 189 image

(b) 832 DOG extrema

(c) 729 left after peak value threshold

(d) 536 left after testing ratio of principal curvatures

Scale Invariant Feature Transform (**SIFT**)



SIFT describes both a **detector** and **descriptor**

1. Multi-scale extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

4. Keypoint Description

We have seen how to assign a location, scale, and orientation to each key point — **keypoint detection**

— The next step is to compute a **keypoint descriptor**: should be robust to local shape distortions, changes in illumination or 3D viewpoint

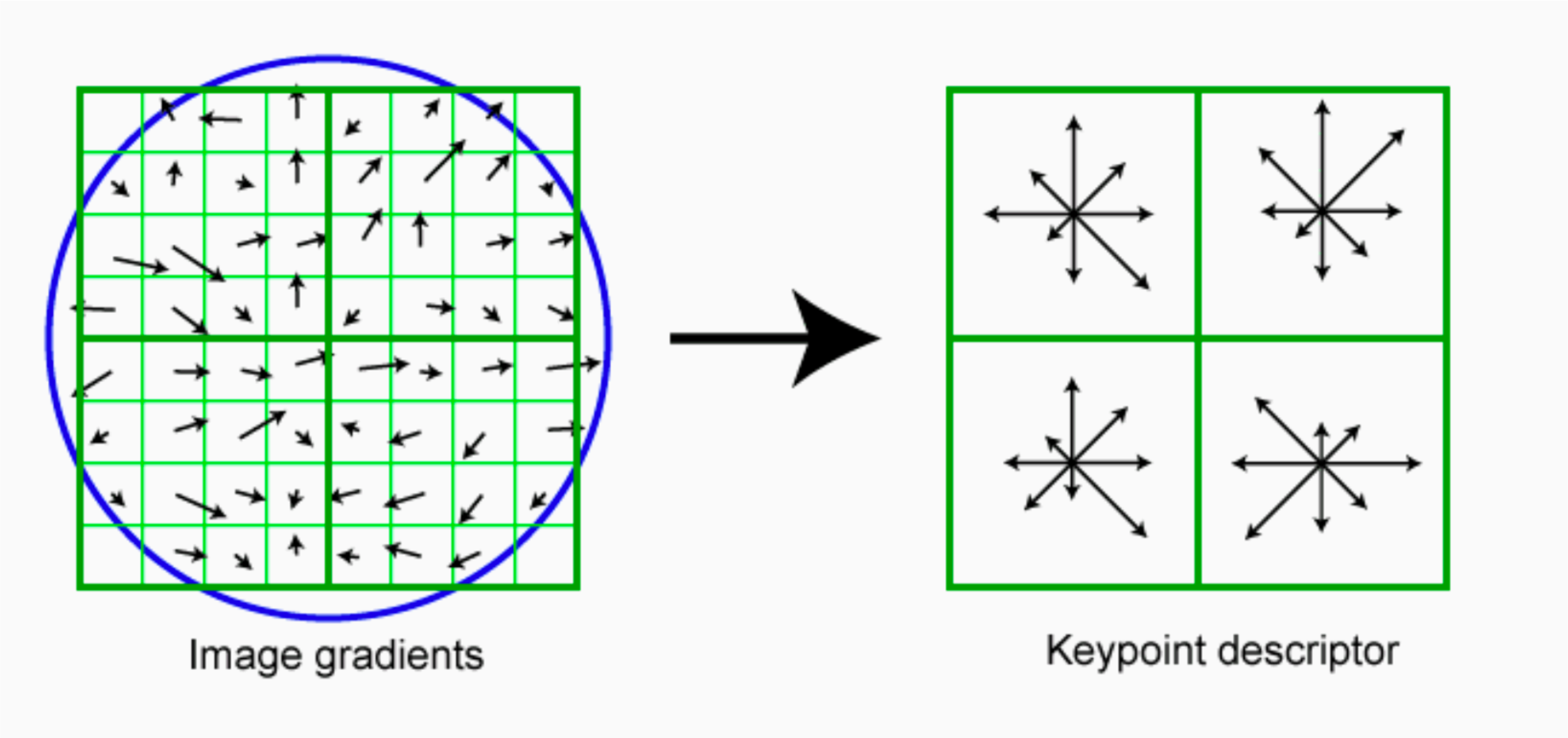
— Keypoint detection is not the same as keypoint description, e.g. some applications skip keypoint detection and extract SIFT descriptors on a regularly spaced grid

4. SIFT Descriptor

— Image gradients are sampled over 16 × 16 array of locations in scale space (weighted by a Gaussian with sigma half the size of the window)

— Create array of orientation histograms

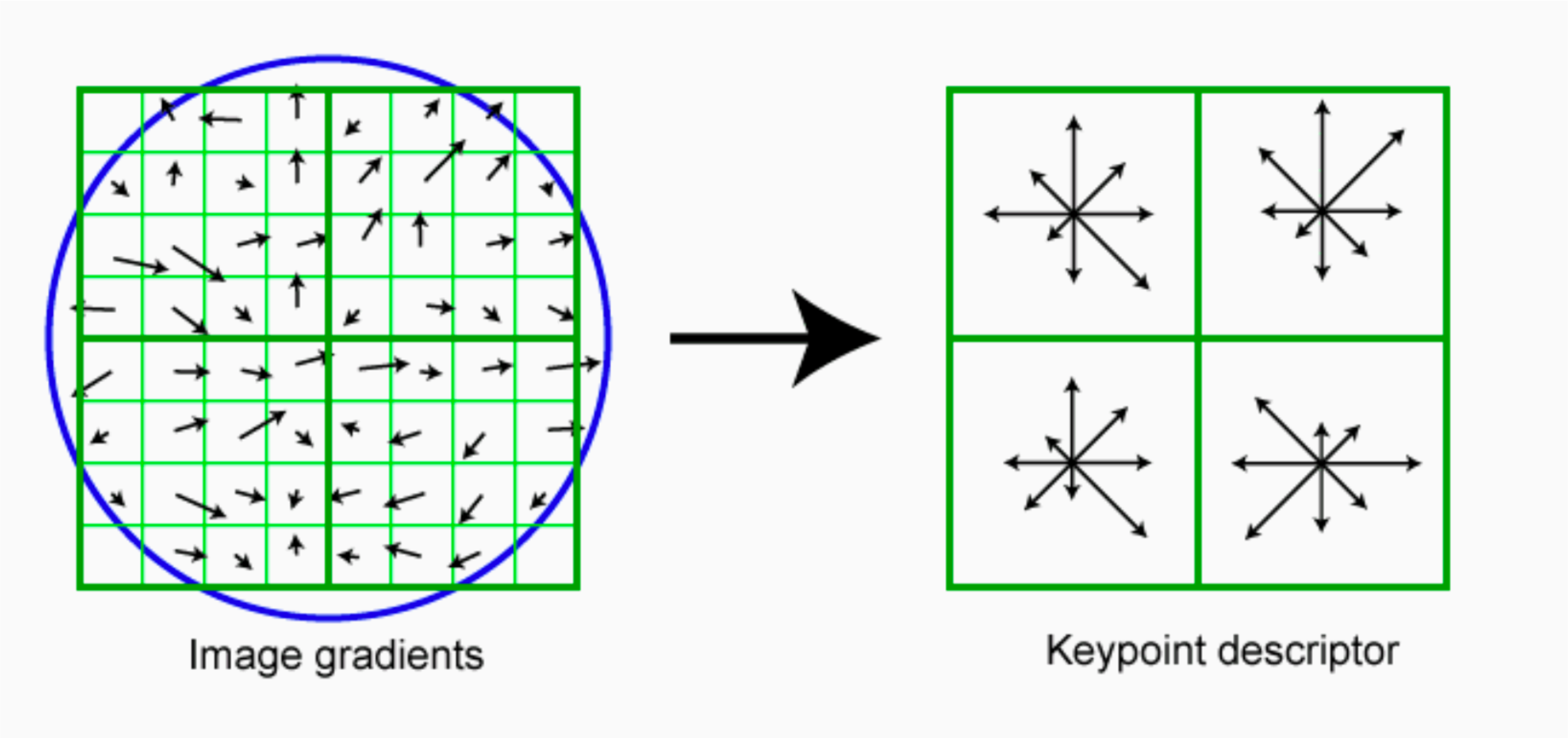
— 8 orientations × 4 × 4 histogram array



4. SIFT Descriptor

How many dimensions are there in a SIFT descriptor?

(**Hint**: This diagram shows a 2 x 2 histogram array but the actual descriptor uses a 4 x 4 histogram array)



4. SIFT Descriptor — Photometric Invariance

Descriptor is **normalized** to unit length (i.e. magnitude of 1) to reduce the effects of illumination change

— if brightness values are **scaled (multiplied)** by a constant, the gradients are scaled by the same constant, and the normalization cancels the change

— if brightness values are **increased/decreased** by a constant **(additive)**, the gradients do not change

SIFT Recap

Detector:

— Find points that are maxima in a DOG pyramid

— Compute local orientation from gradient histogram

— This establishes a local coordinate frame with scale/orientation

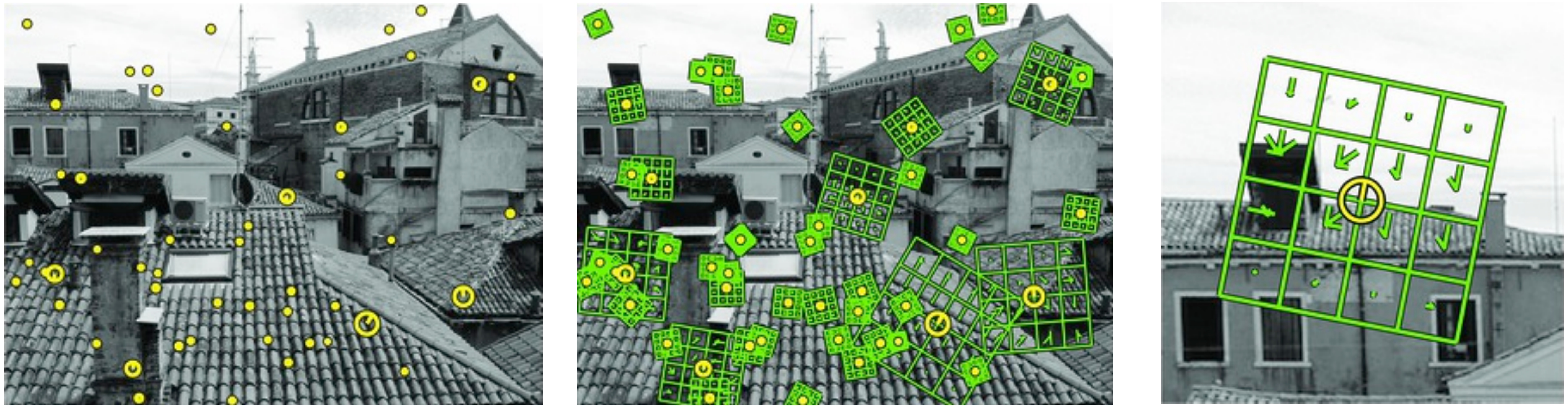
Descriptor:

— Build histograms over gradient orientations (8 orientations, 4x4 grid)

— Normalise the final descriptor to reduce the effects of illumination change

SIFT Matching

Extract features from the image …

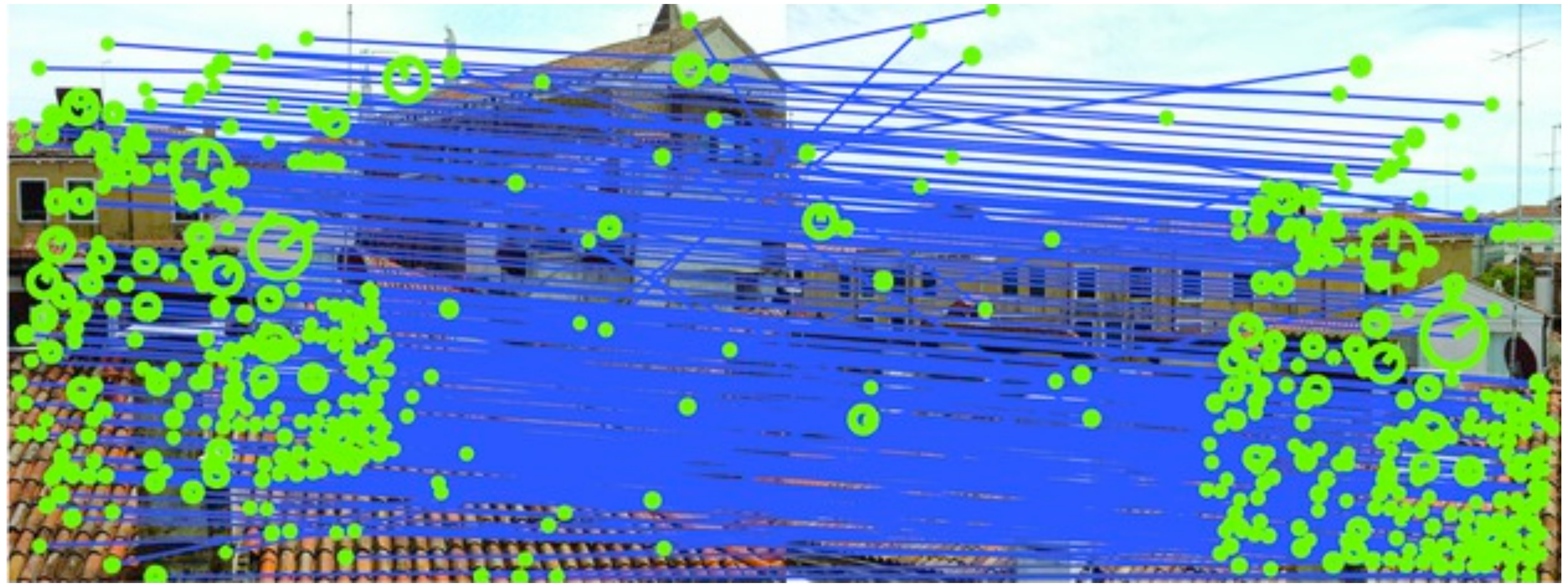


Each image might generate 100’s or 1000’s of SIFT descriptors

SIFT Matching

Goal: Find all correspondences between a pair of images

Means: extract and match all SIFT descriptors from both images



SIFT Matching

— Each SIFT feature is represented by 128-D vector (numbers)

— Feature matching becomes the task of finding the closest 128-D vector

— Nearest-neighbor matching:

$$NN(j) = \arg\min_i |\mathbf{x}_i - \mathbf{x}_j|, \; i \neq j$$

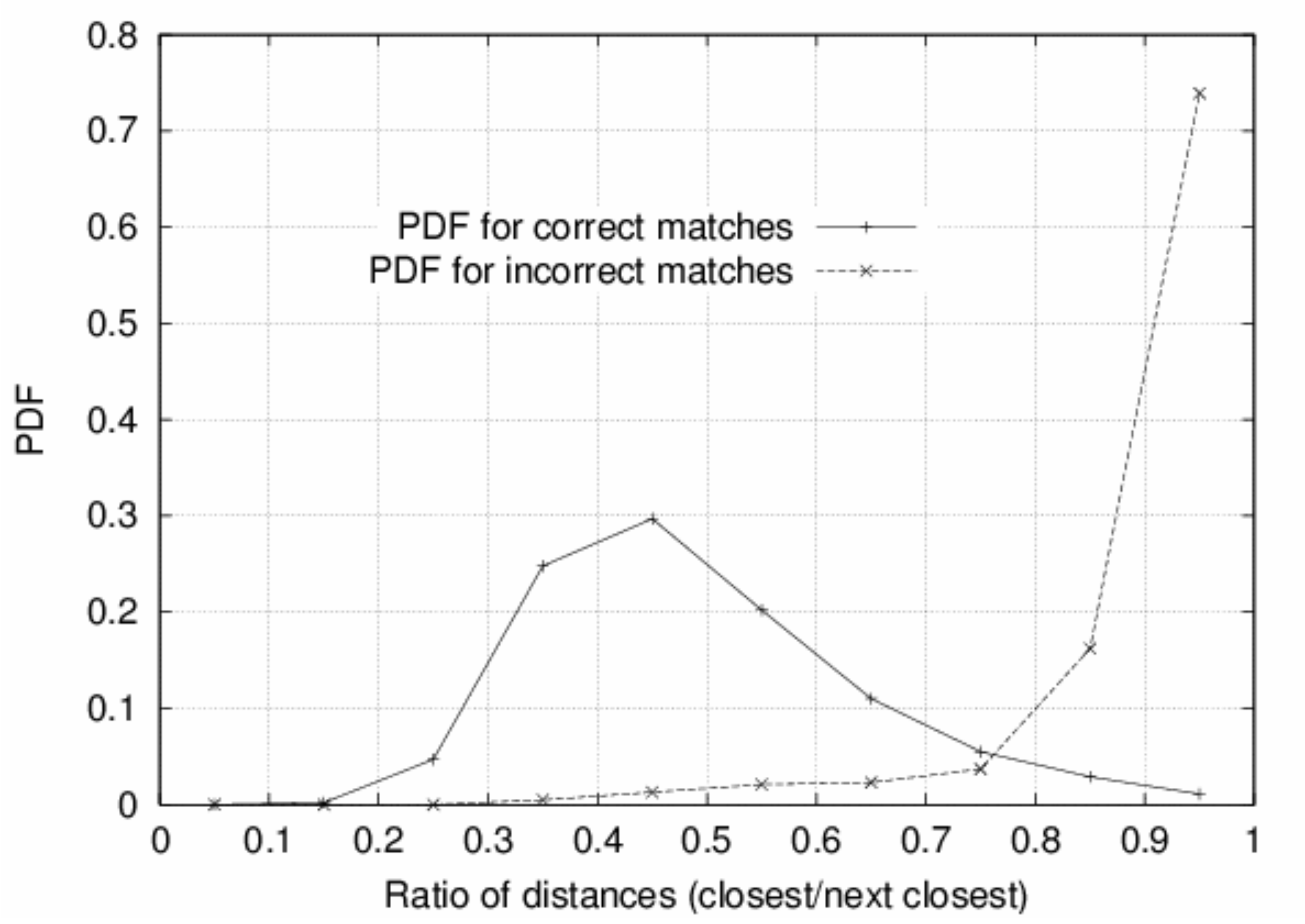
— This is expensive (linear time), but good approximation algorithms exist

e.g., Best Bin First K-d Tree [Beis Lowe 1997], FLANN (Fast Library for Approximate Nearest Neighbours) [Muja Lowe 2009]

Match **Ratio Test**

Compare ratio of distance of **nearest** neighbour (1NN) to **second** nearest (2NN) neighbour — this will be a non-matching point

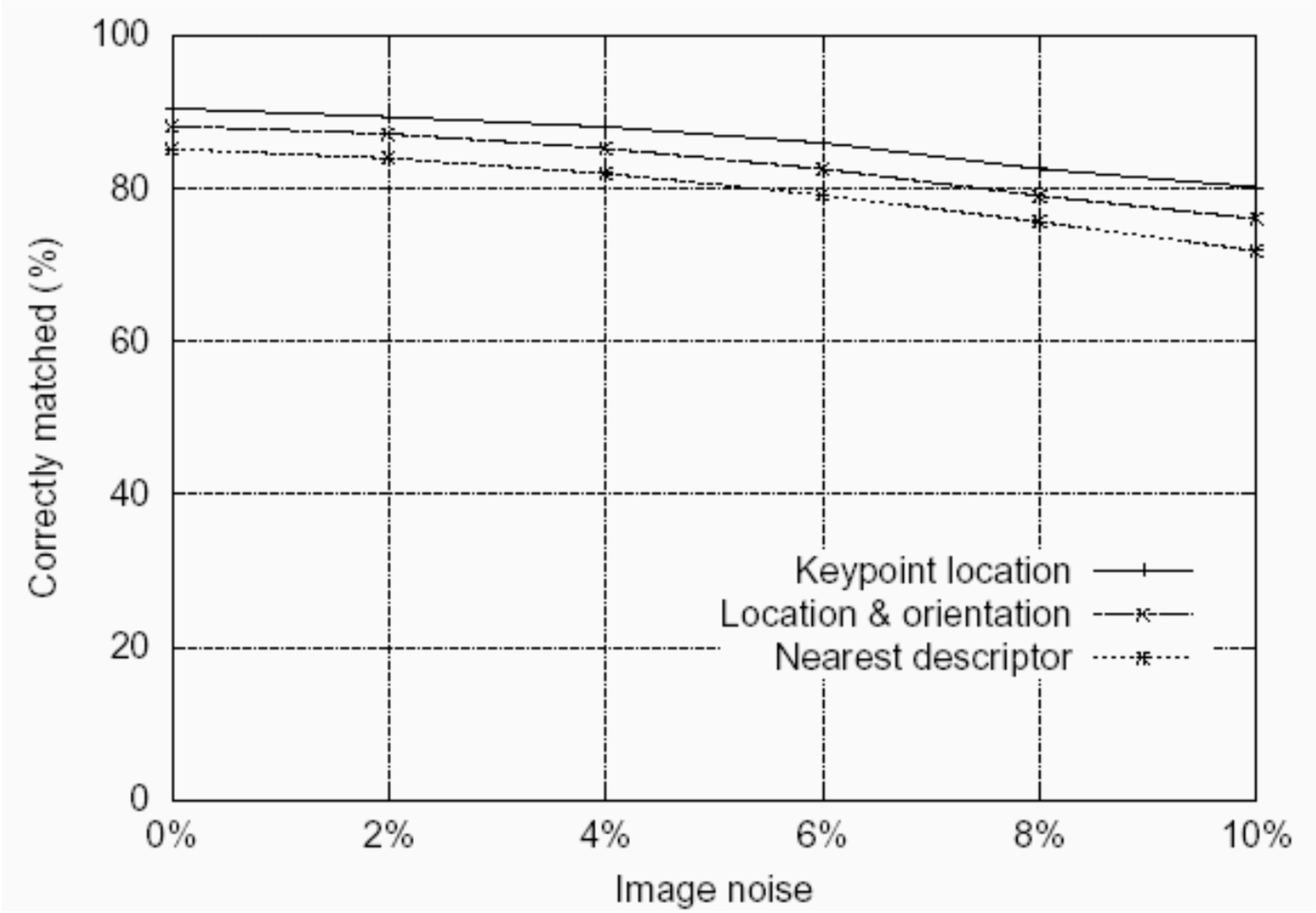
Rule of thumb: d(1NN) < 0.8 * d(2NN) for good match



Feature Stability to **Noise**

Match features after random change in image scale & orientation, with differing levels of image noise

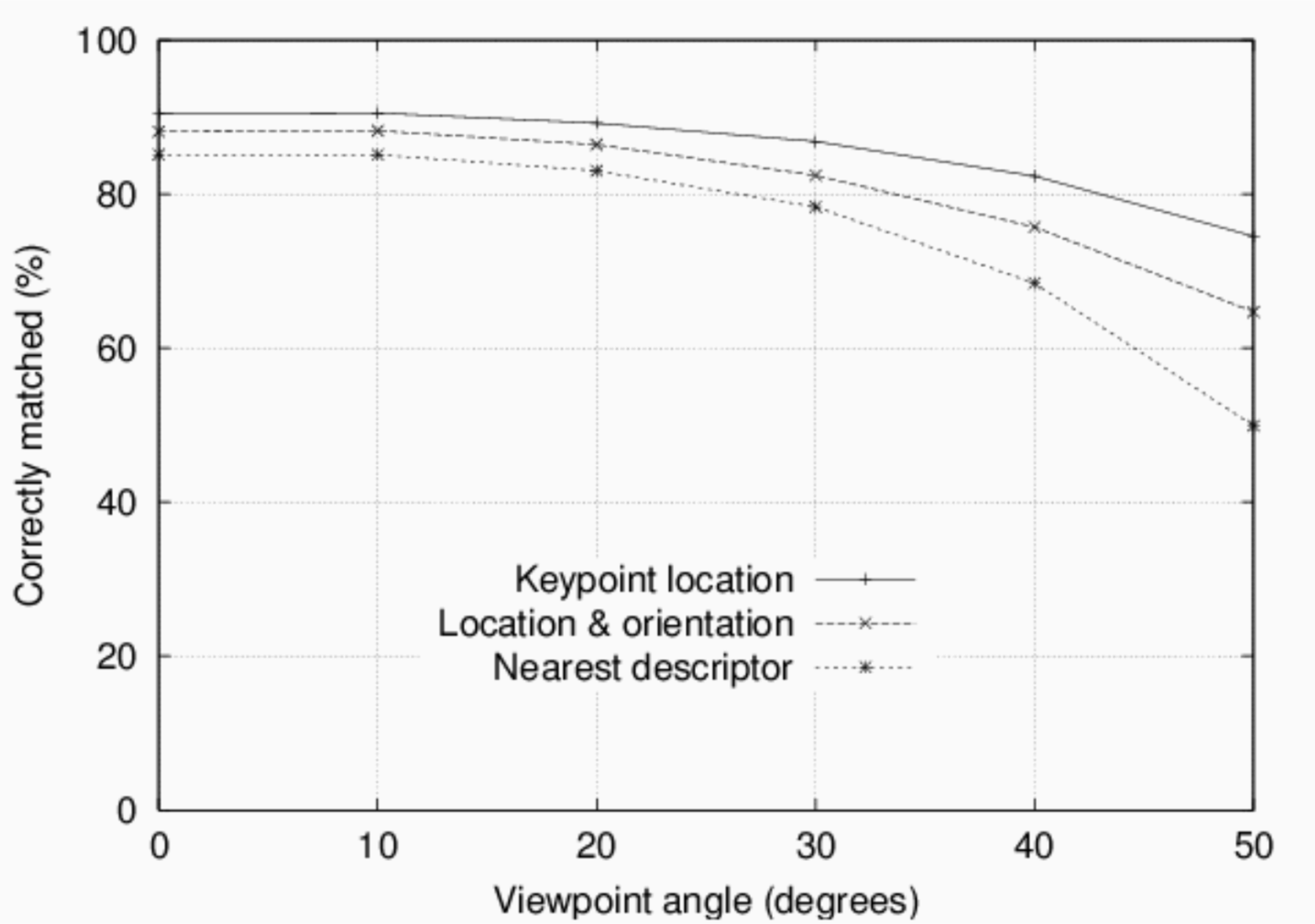
Find nearest neighbour in database of 30,000 features



Feature Stability to **Affine Change**

Match features after random change in image scale & orientation, with differing levels of image noise

Find nearest neighbour in database of 30,000 features



Summary

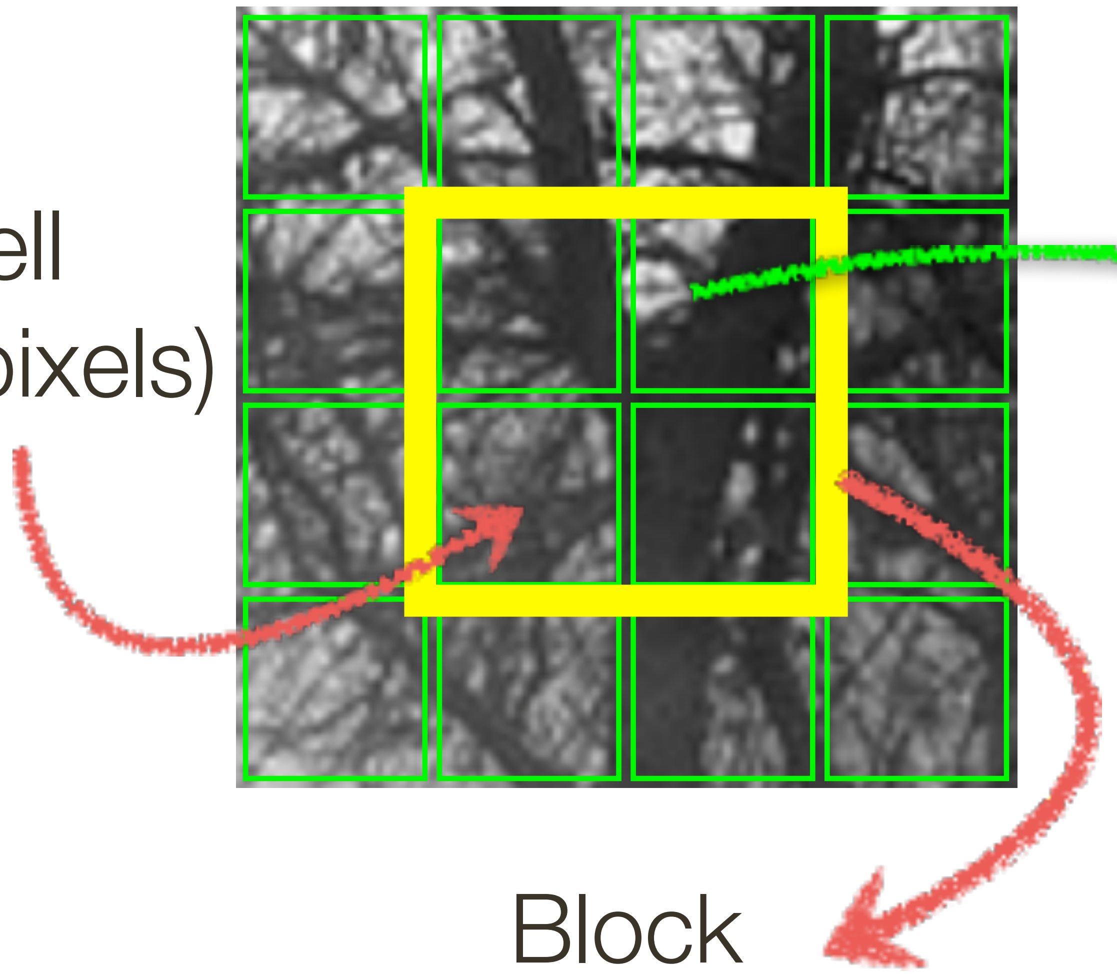
Four steps to SIFT feature generation:

1. **Scale-space representation and local extrema detection**
   — use DoG pyramid
   — 3 scales/octave, down-sample by factor of 2 each octave
2. **Keypoint localization**
   — select stable keypoints (threshold on magnitude of extremum, ratio of principal curvatures)
3. **Keypoint orientation assignment**
   — based on histogram of local image gradient directions
4. **Keypoint descriptor**
   — histogram of local gradient directions — vector with $8 \times (4 \times 4) = 128$ dim
   — vector normalized (to unit length)

Histogram of Oriented Gradients (**HOG**) Features

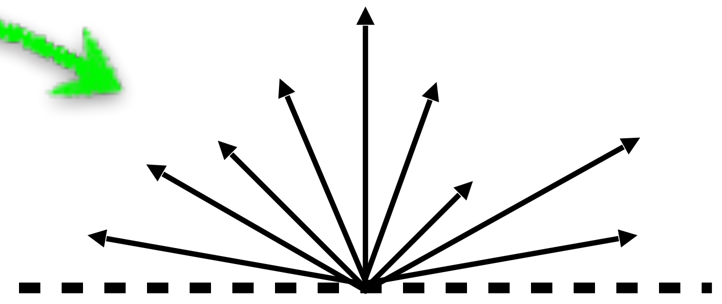
Dalal, Triggs. Histograms of Oriented Gradients for Human Detection. CVPR, 2005

Cell
(8x8 pixels)



Block
(2x2 cells)

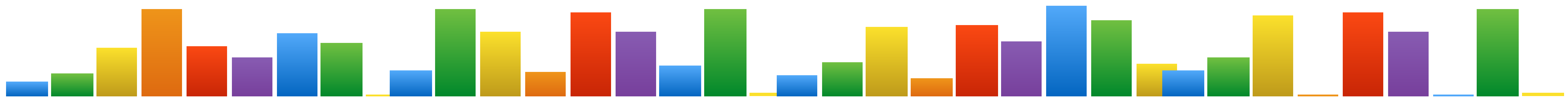
histogram of
'unsigned'
gradients



soft binning

gradient magnitude histogram
(one for each cell)

Concatenate and L-2 normalization



Single scale, no dominant orientation

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Histogram of Oriented Gradients (**HOG**) Features

Pedestrian detection

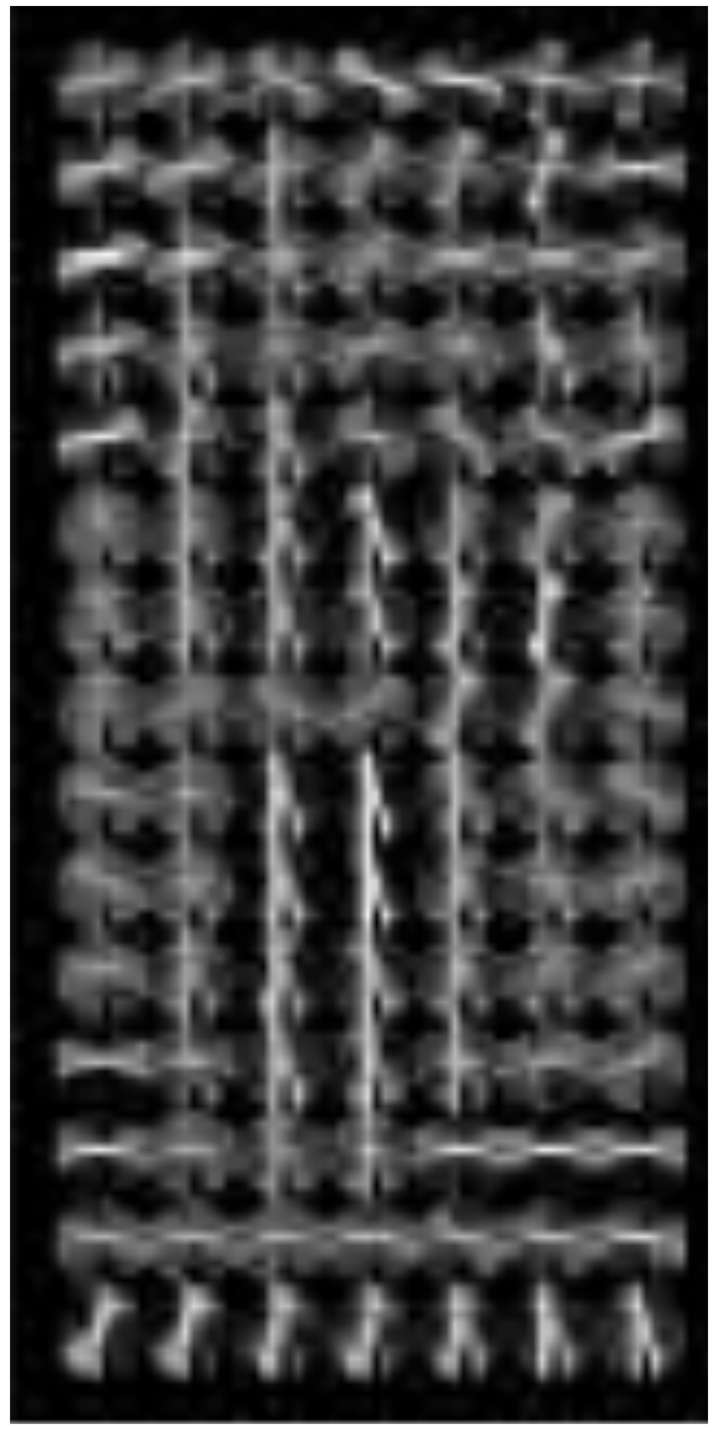
1 cell step size

128 pixels
16 cells
15 blocks

64 pixels
8 cells
7 blocks

15 x 7 x 4 x 9 =
3780

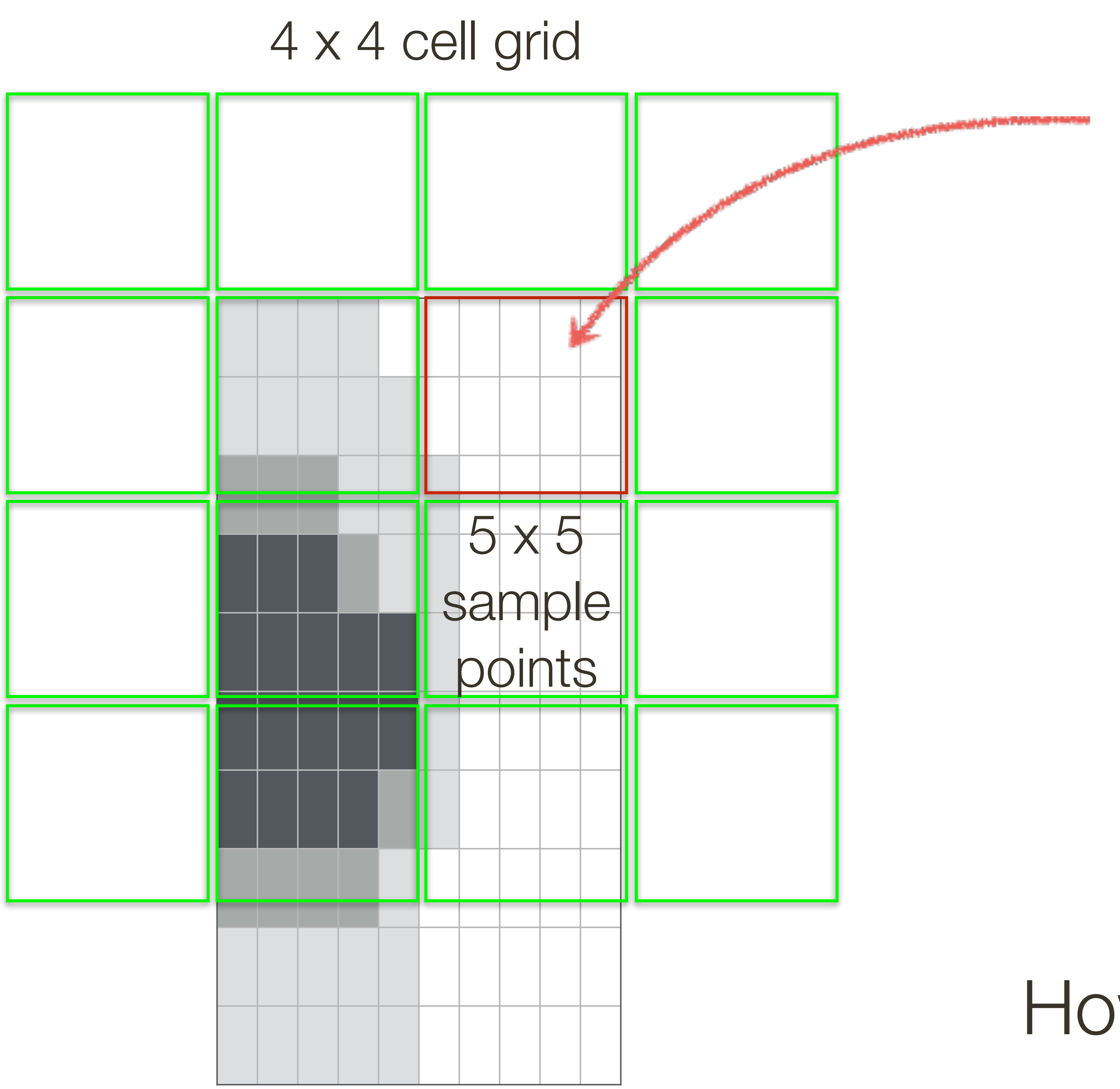
visualization



Redundant representation due to overlapping blocks

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

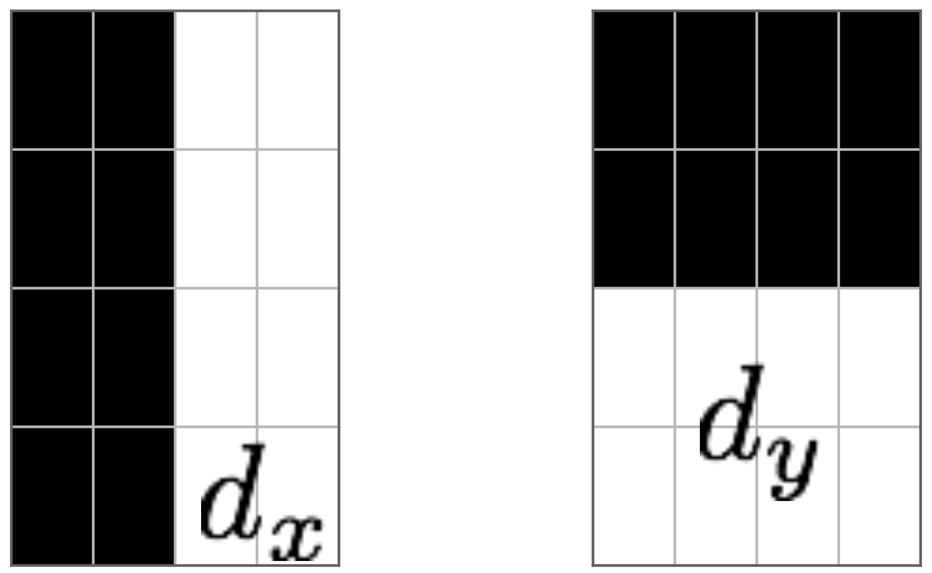
‘Speeded’ Up Robust Features (**SURF**)



Each cell is represented by 4 values:

$$\left[\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right]$$

Haar wavelets filters
(Gaussian weighted from center)

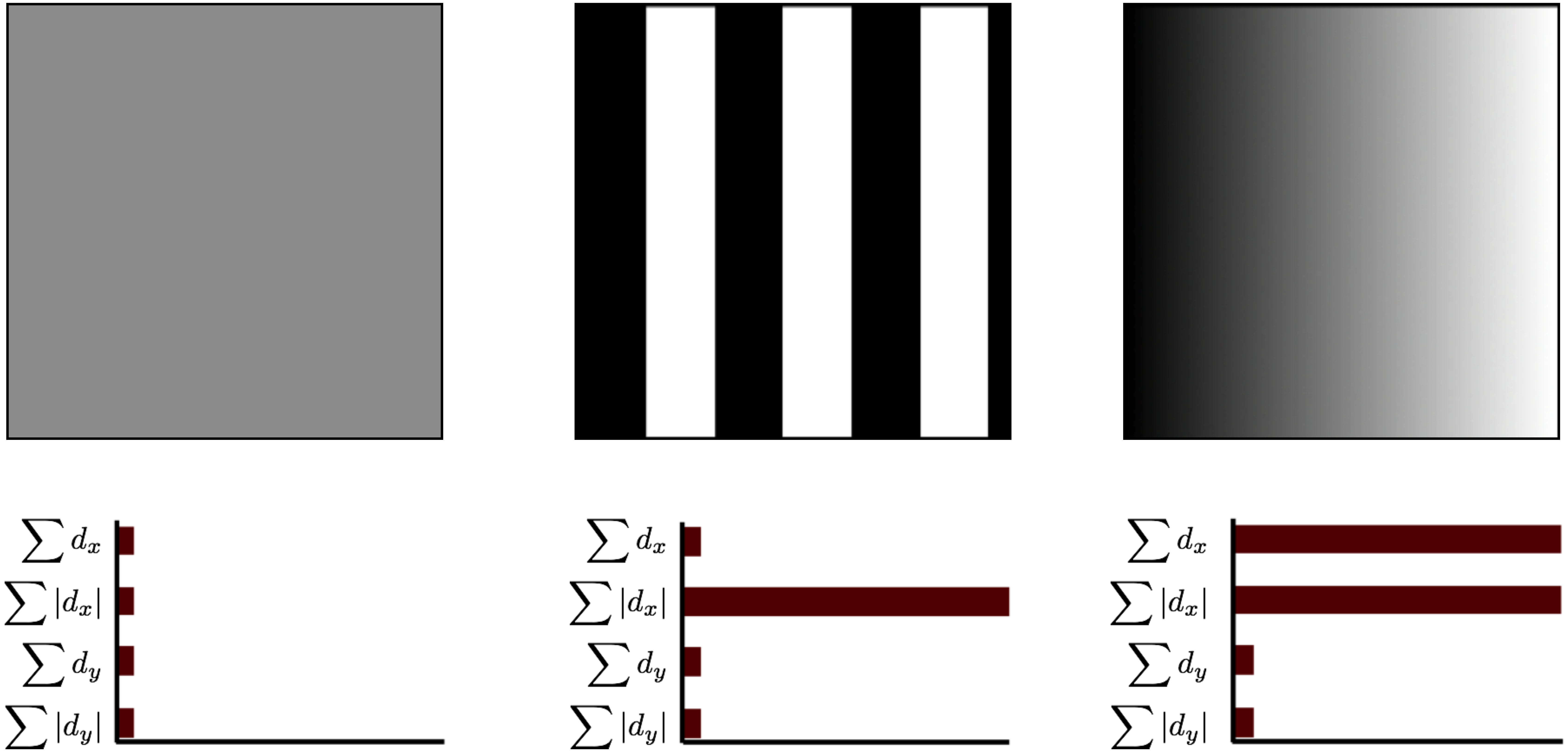


How big is the SURF descriptor?

64 dimensions

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

‘Speeded’ Up Robust Features (**SURF**)



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

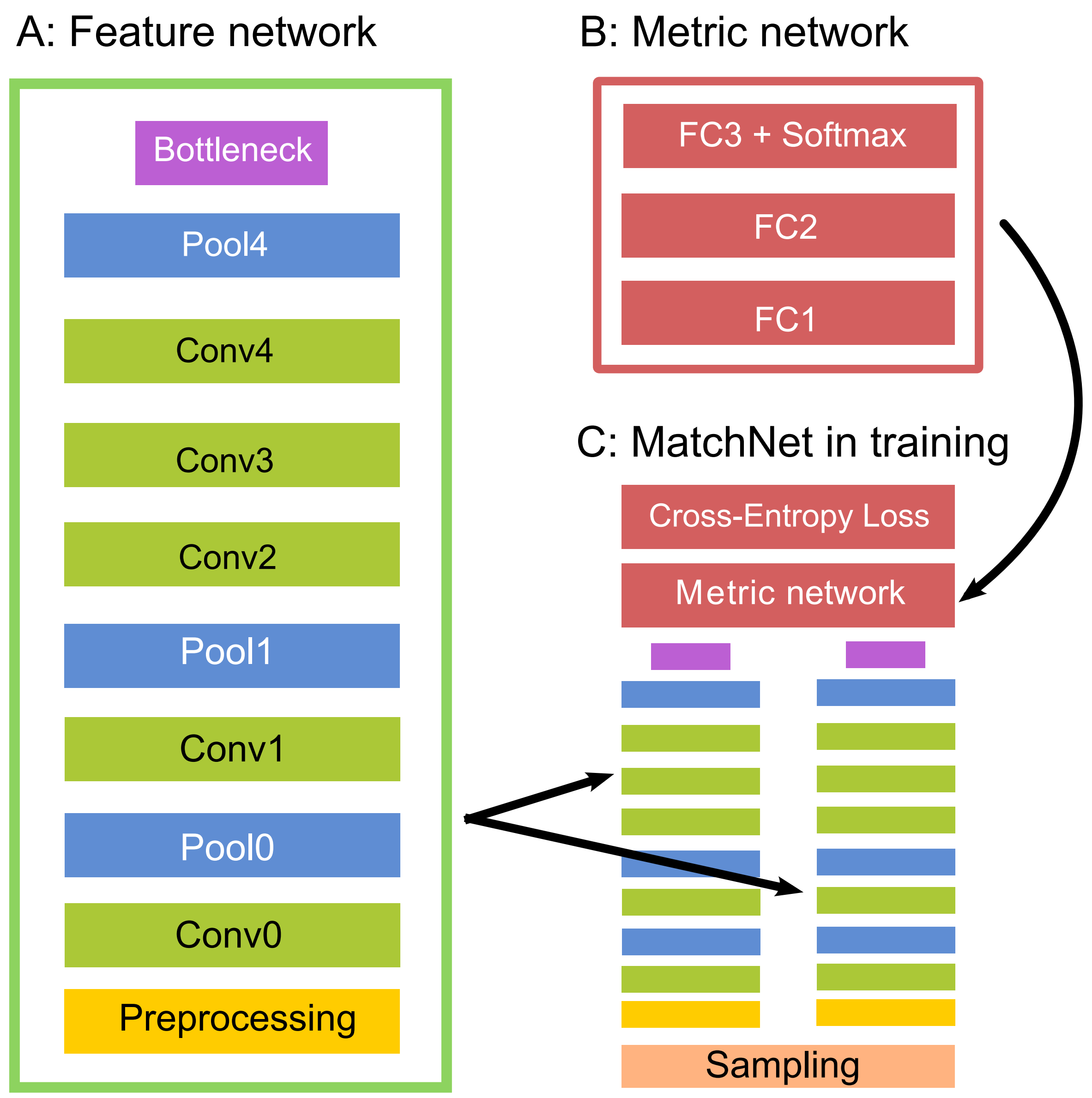
Keypoint **Detectors** vs. **Descriptors**

— Harris	— SIFT
— Blob (Laplacian)	— HoG
— SIFT	— SURF

Learning Descriptors

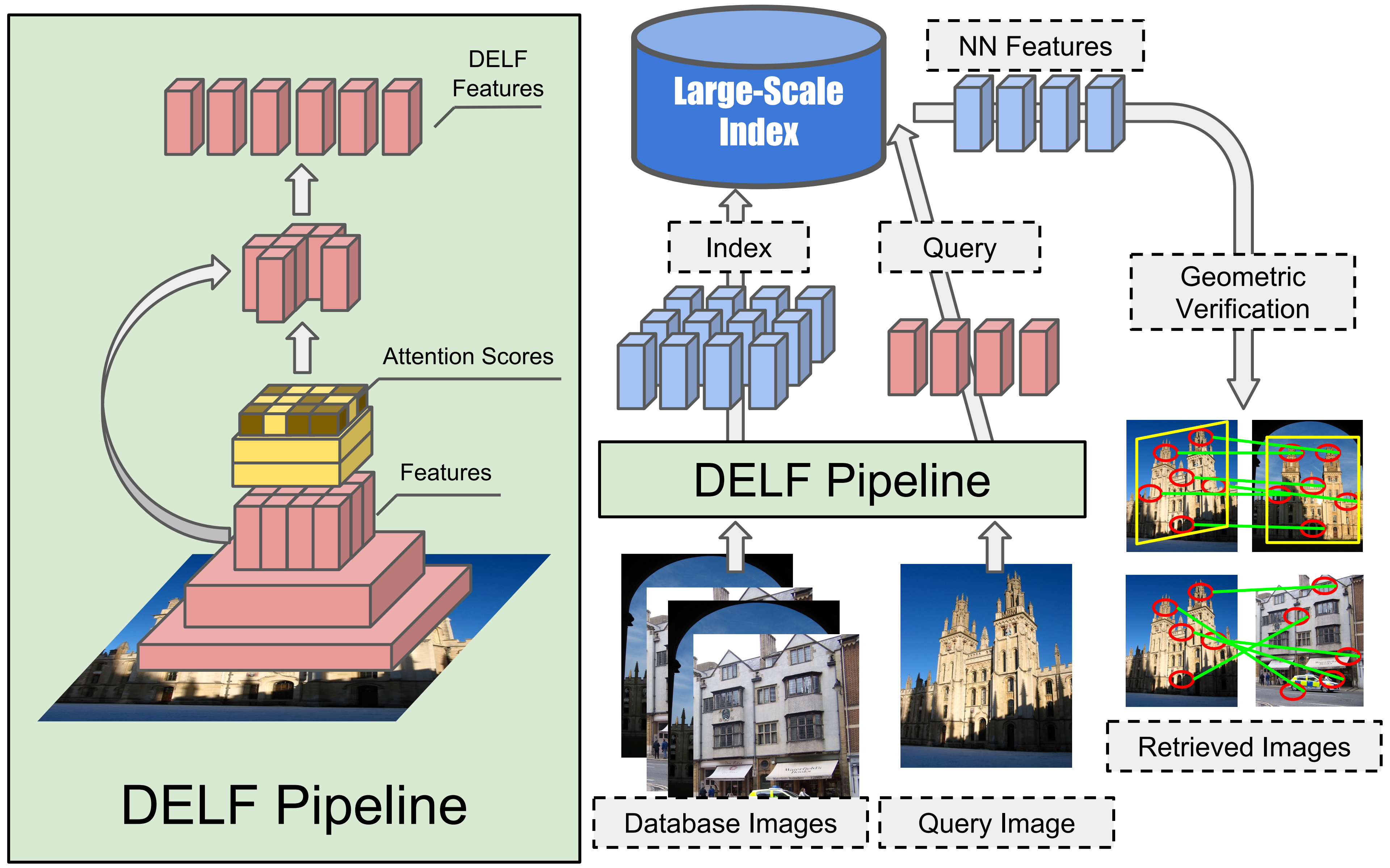
- Deep networks for descriptor learning

Patch labels



[MatchNet
Han et al 2015]

Image labels, also learns interest function



[DELF
Noh et al 2017]

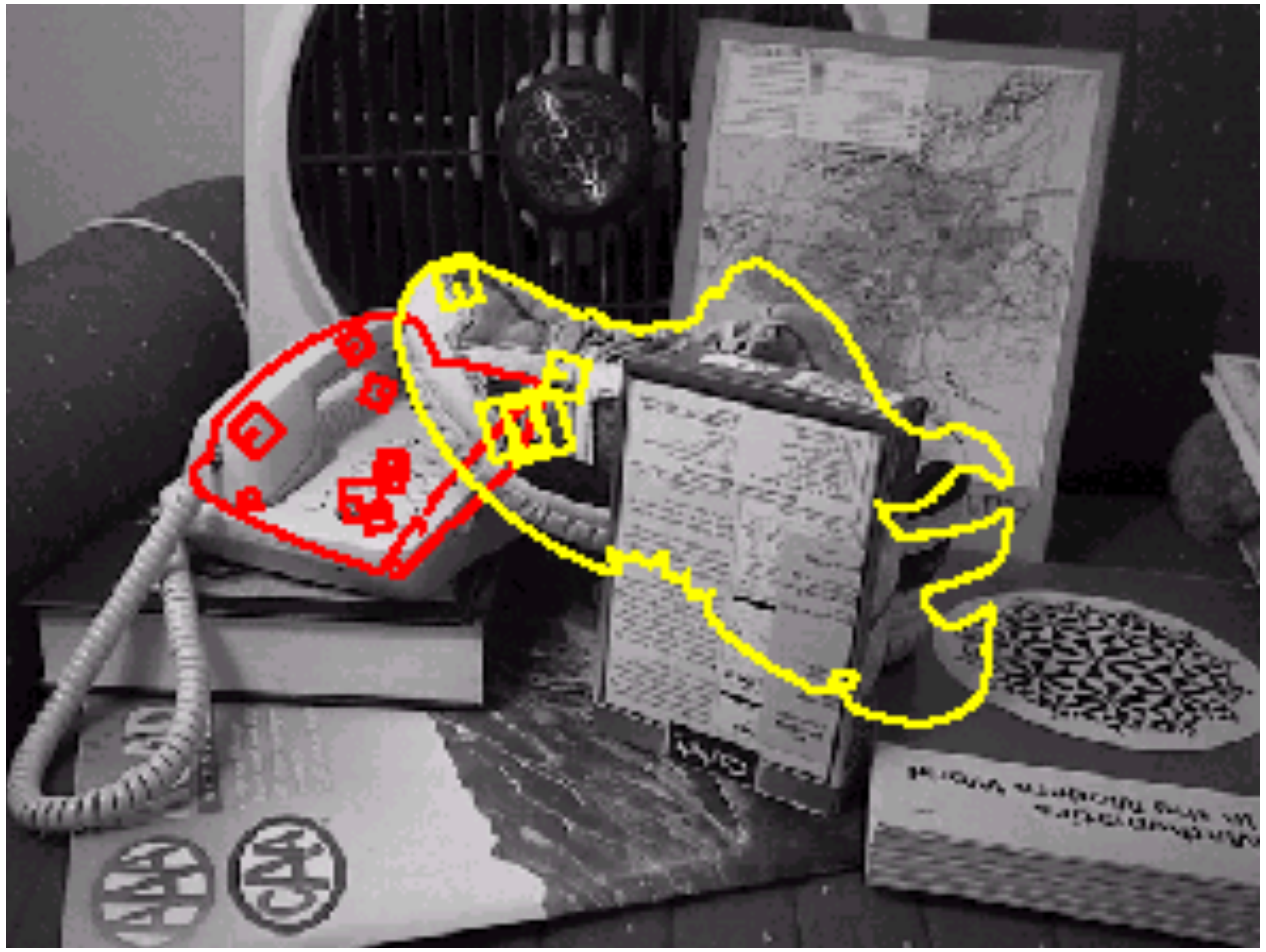
Planar Object **Instance Recognition**

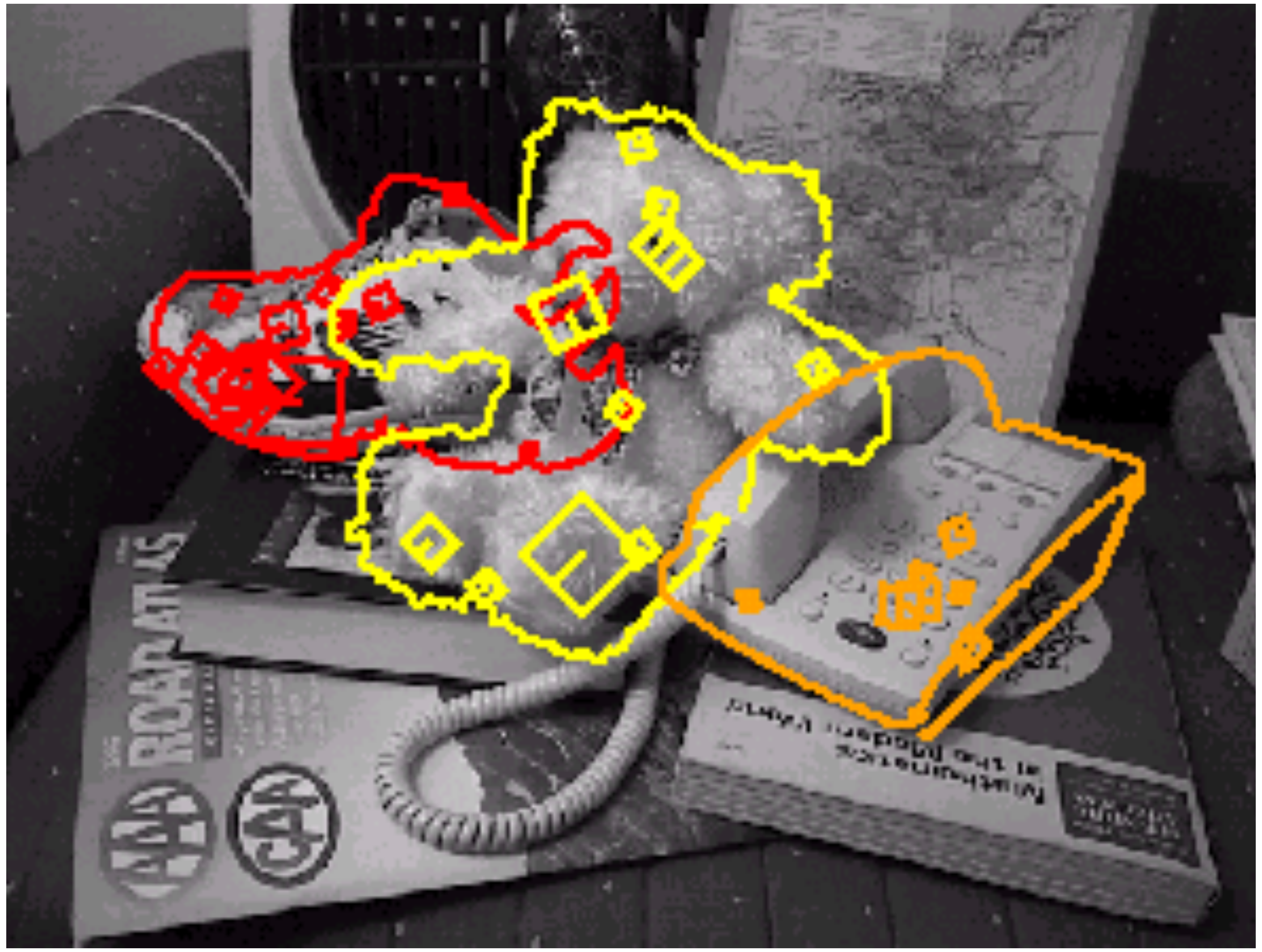
Database of planar objects

Instance recognition

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Recognition under **Occlusion**





Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)