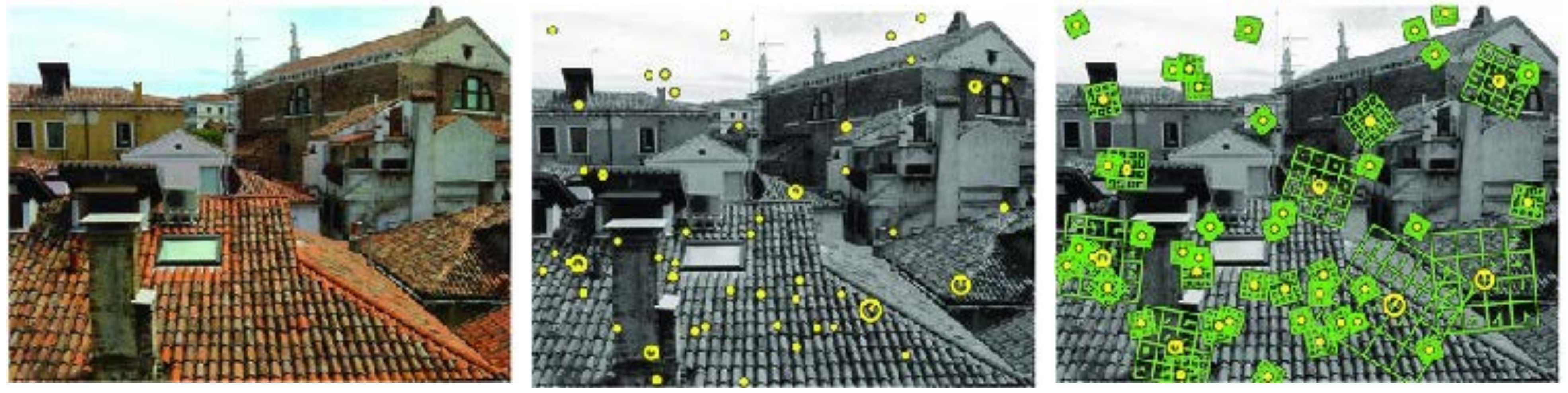
THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision



Lecture 12: 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:

- **Assignment 3:** due **October 29th!**

Learning **Goals**

1. The design philosophy behind SIFT

Scale Invariant Feature Transform = SIFT

Distinctive Image Features
from Scale-Invariant Keypoints

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

January 5, 2004

Abstract

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

Accepted for publication in the *International Journal of Computer Vision*, 2004.

1

The SIFT paper (David Lowe) was rejected twice (and eventually published only as a Poster). Became one of the most **influential** and **widely cited papers** in the field ~ 107,000 citations.

Building a panorama

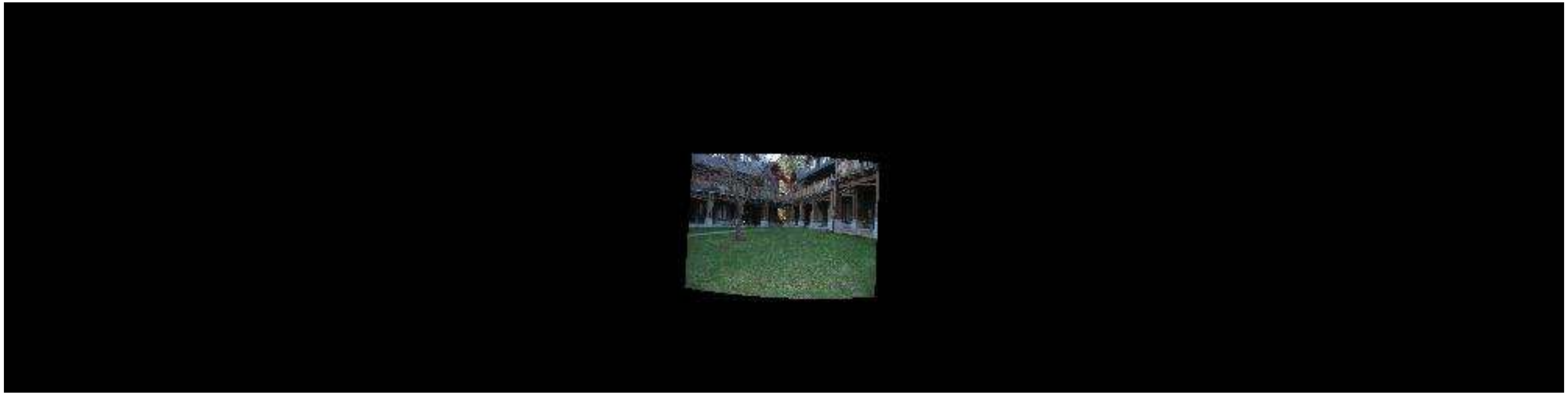


Figure Credit: Matthew Brown and David Lowe

Building a panorama

Figure Credit: Matthew Brown and David Lowe

Building a panorama

Figure Credit: Matthew Brown and David Lowe

Building a panorama

Figure Credit: Matthew Brown and David Lowe

Building a panorama

Figure Credit: Matthew Brown and David Lowe

Building a panorama

Figure Credit: Matthew Brown and David Lowe

Building a panorama

Figure Credit: Matthew Brown and David Lowe

Building a panorama

Figure Credit: Matthew Brown and David Lowe

Building a panorama

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...

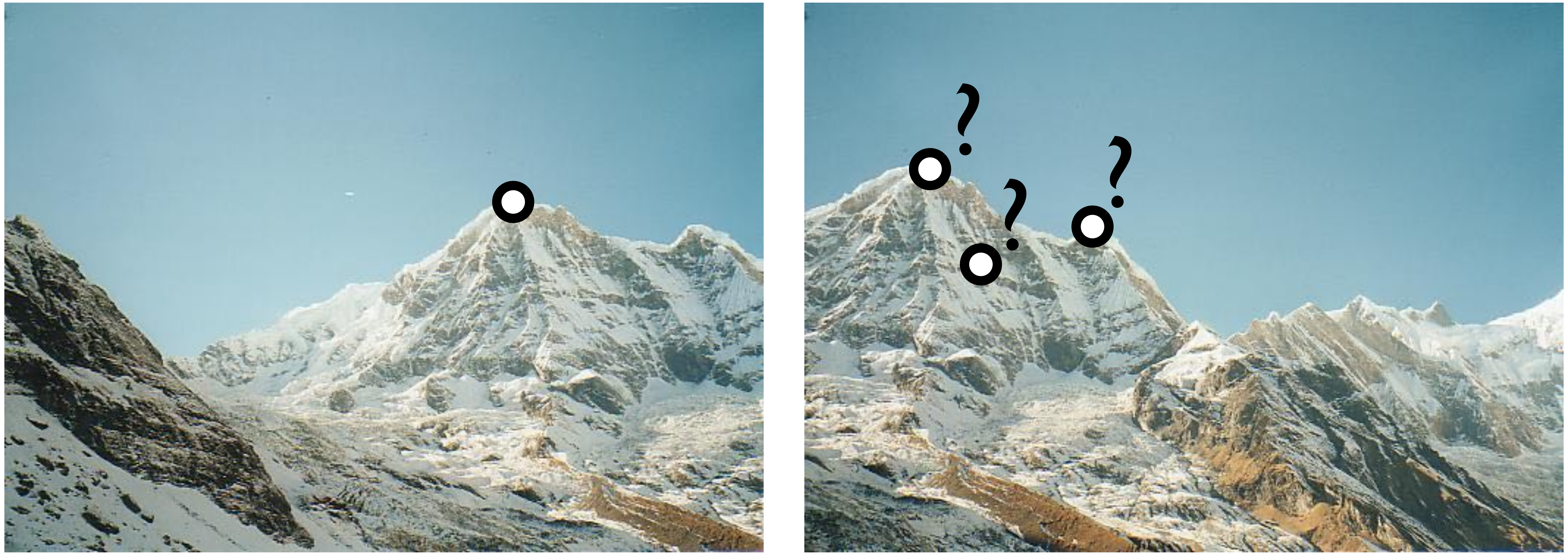


Image **Panoramas**

Building Rome in a Day

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

Still in 2025

NVIDIA 2025-10-20

CuSfM: CUDA-Accelerated Structure-from-Motion

Jingrui Yu[1], Jun Liu[1], Kefei Ren[1], Joydeep Biswas[1], Rurui Ye[1], Keqiang Wu[1], Chirag Majithia[1], Di Zeng[1*]
[1]NVIDIA
*Project Lead
{jingruiy, junli, kefeir, jbiswas, ruruiye, keqiangw, cmajithia, dizeng}@nvidia.com

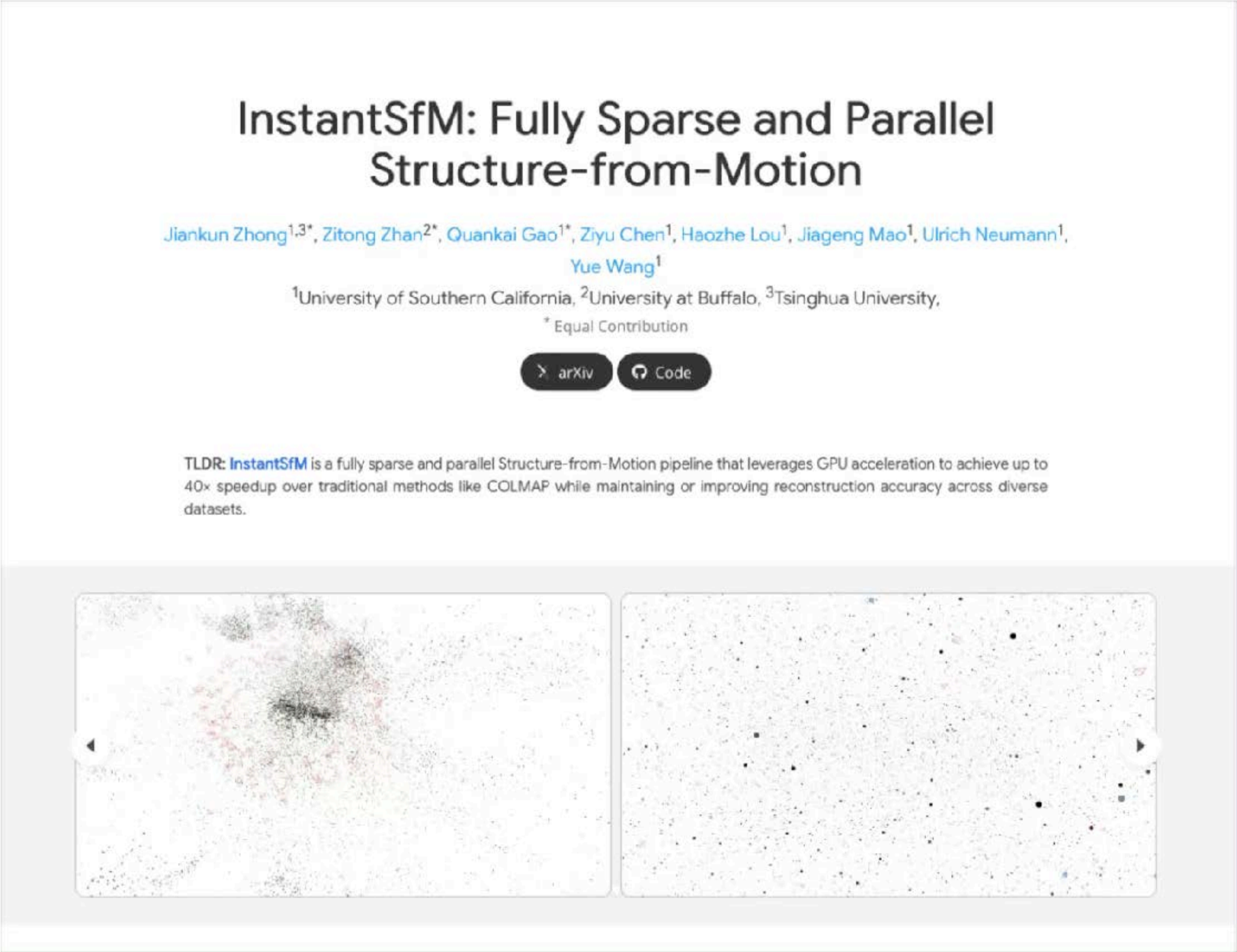
Abstract

Efficient and accurate camera pose estimation forms the foundational requirement for dense reconstruction in autonomous navigation, robotic perception, and virtual simulation systems. This paper addresses the challenge via cuSfM, a CUDA-accelerated offline Structure-from-Motion system that leverages GPU parallelization to efficiently employ computationally intensive yet highly accurate feature extractors, generating comprehensive and non-redundant data associations for precise camera pose estimation and globally consistent mapping. The system supports pose optimization, mapping, prior-map localization, and extrinsic refinement. It is designed for offline processing, where computational resources can be fully utilized to maximize accuracy. Experimental results demonstrate that cuSfM achieves significantly improved accuracy and processing speed compared to the widely used COLMAP method across various testing scenarios, while maintaining the high precision and global consistency essential for offline SfM applications. The system is released as an open-source Python wrapper implementation, PyCuSfM, available at https://github.com/nvidia-isaac/pyCuSFM, to facilitate research and applications in computer vision and robotics.

1. Introduction

Structure-from-Motion (SfM), a well-established technique in computer vision, refers to the process of re-constructing accurate three-dimensional structures by analyzing the motion and viewpoints of a sequence of

[cs.CV] 17 Oct 2025



InstantSfM: Fully Sparse and Parallel Structure-from-Motion

Jiankun Zhong[1,3*], Zitong Zhan[2*], Quankai Gao[1*], Ziyu Chen[1], Haozhe Lou[1], Jiageng Mao[1], Ulrich Neumann[1], Yue Wang[1]
[1]University of Southern California, [2]University at Buffalo, [3]Tsinghua University,
* Equal Contribution

arXiv Code

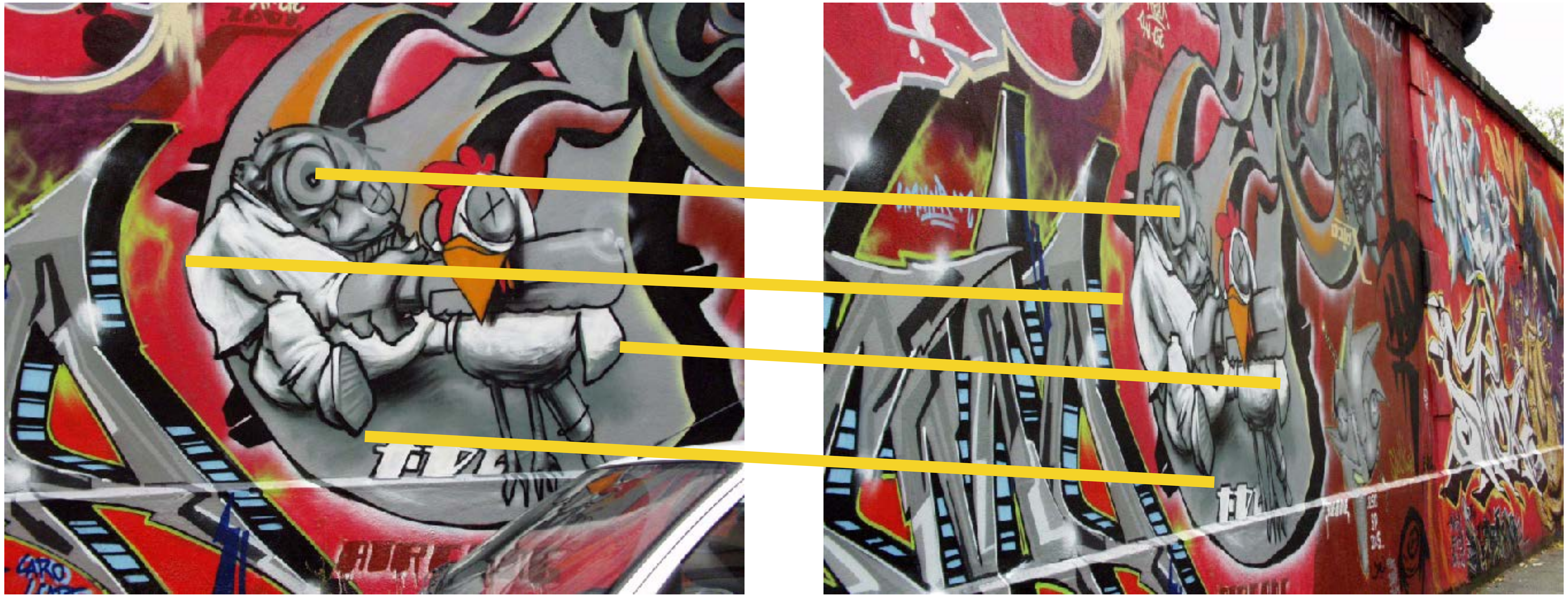
TLDR: InstantSfM is a fully sparse and parallel Structure-from-Motion pipeline that leverages GPU acceleration to achieve up to 40× speedup over traditional methods like COLMAP while maintaining or improving reconstruction accuracy across diverse datasets.

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)

Back to **Good Local Features**

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

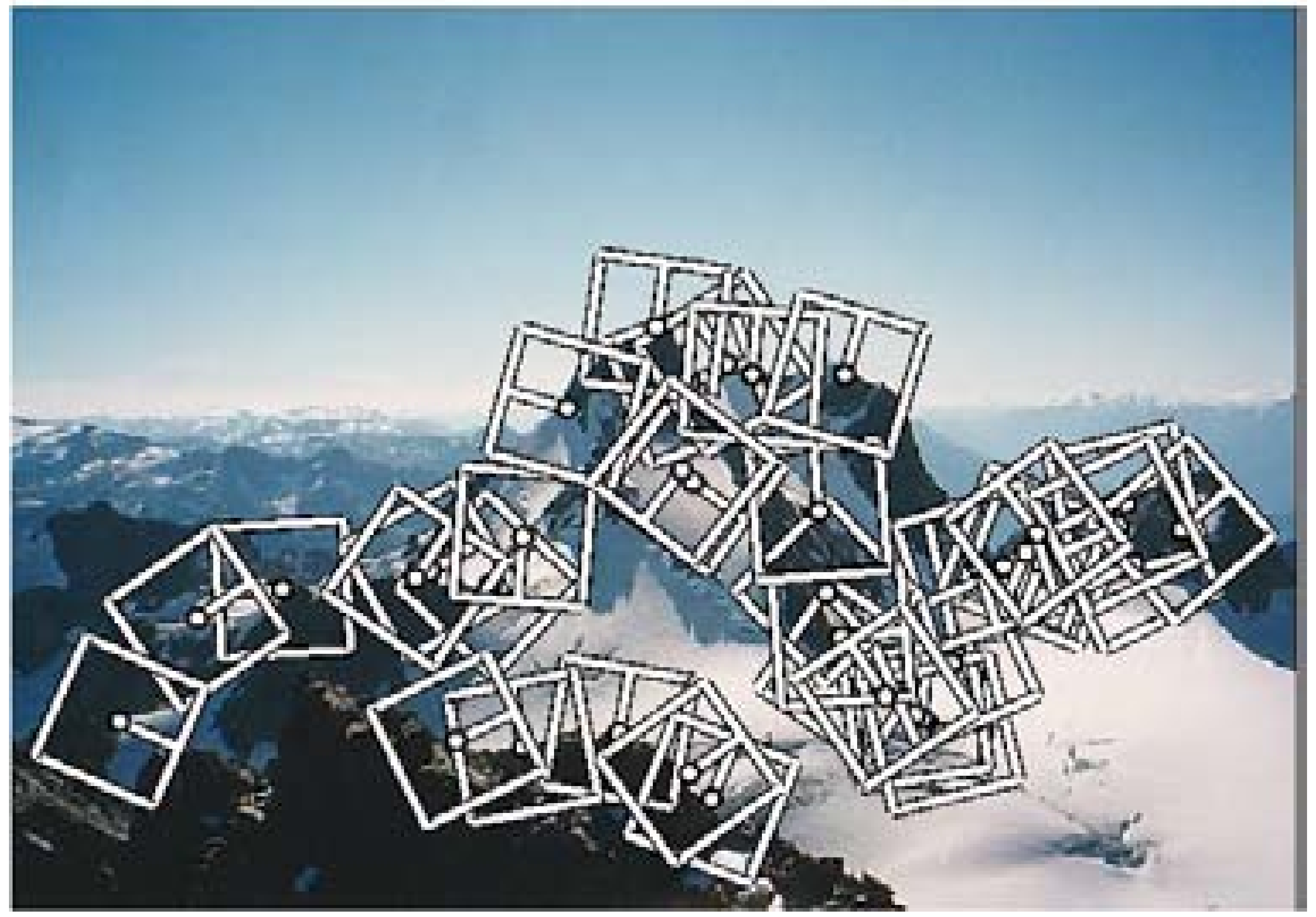
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Back to **Good Local Features**

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Recall: Feature **Detector**



Corners/Blobs

Regions

Edges

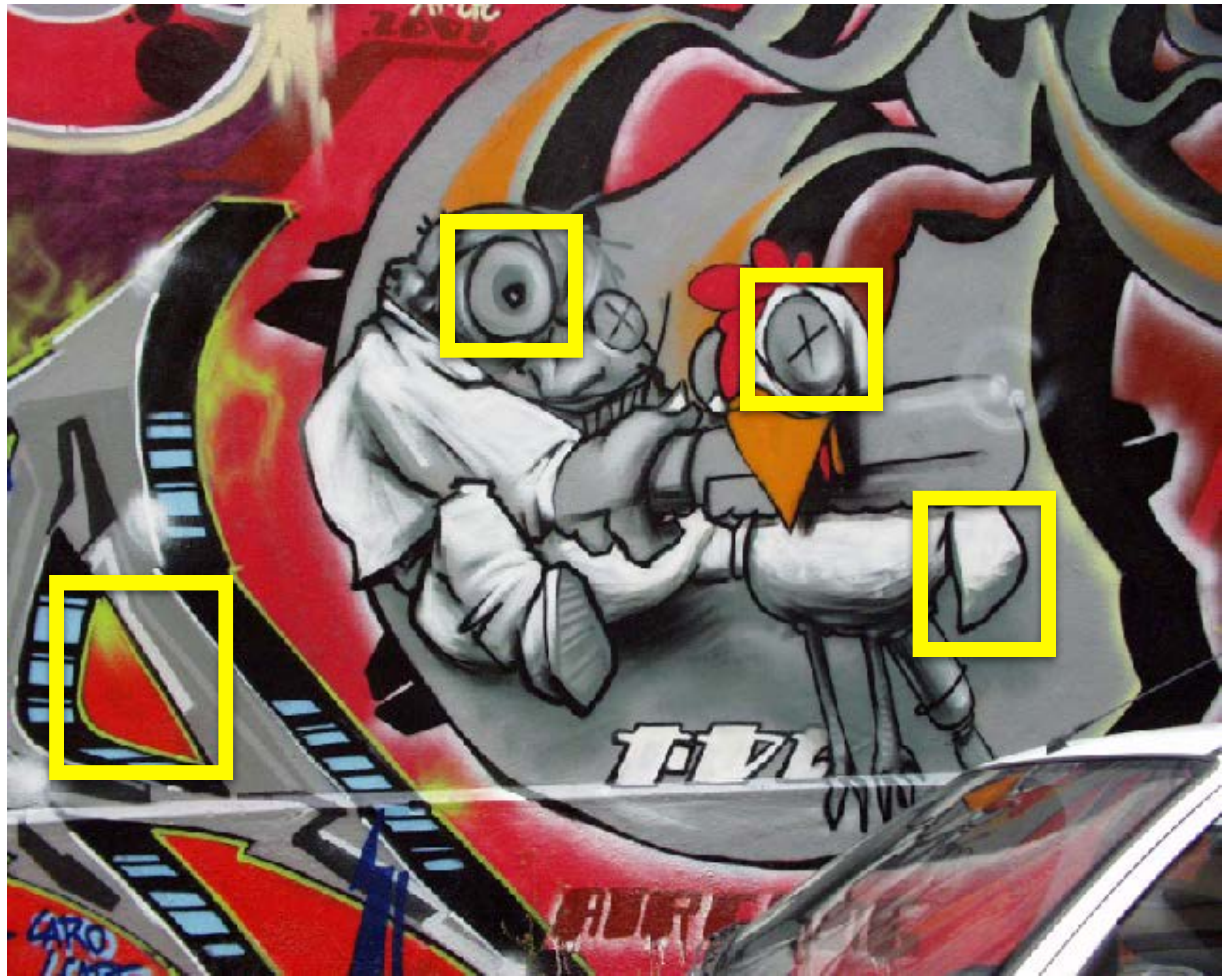
Straight Lines

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**



Patch around the local feature is very informative

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Recall: Feature **Descriptor**

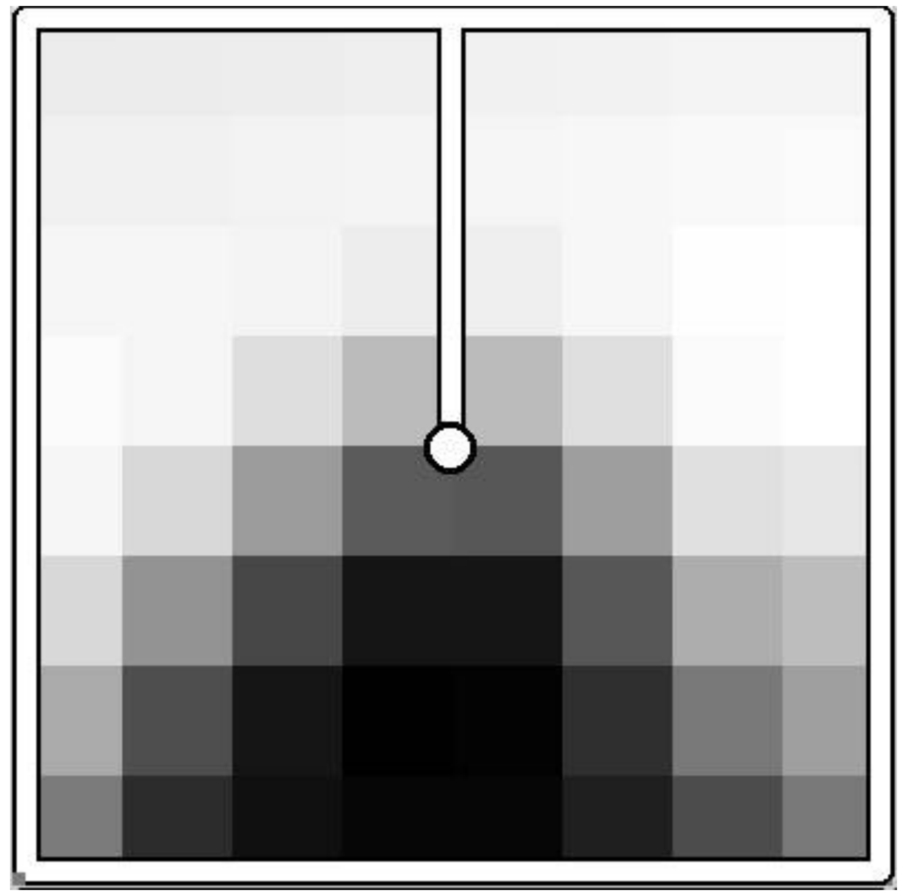
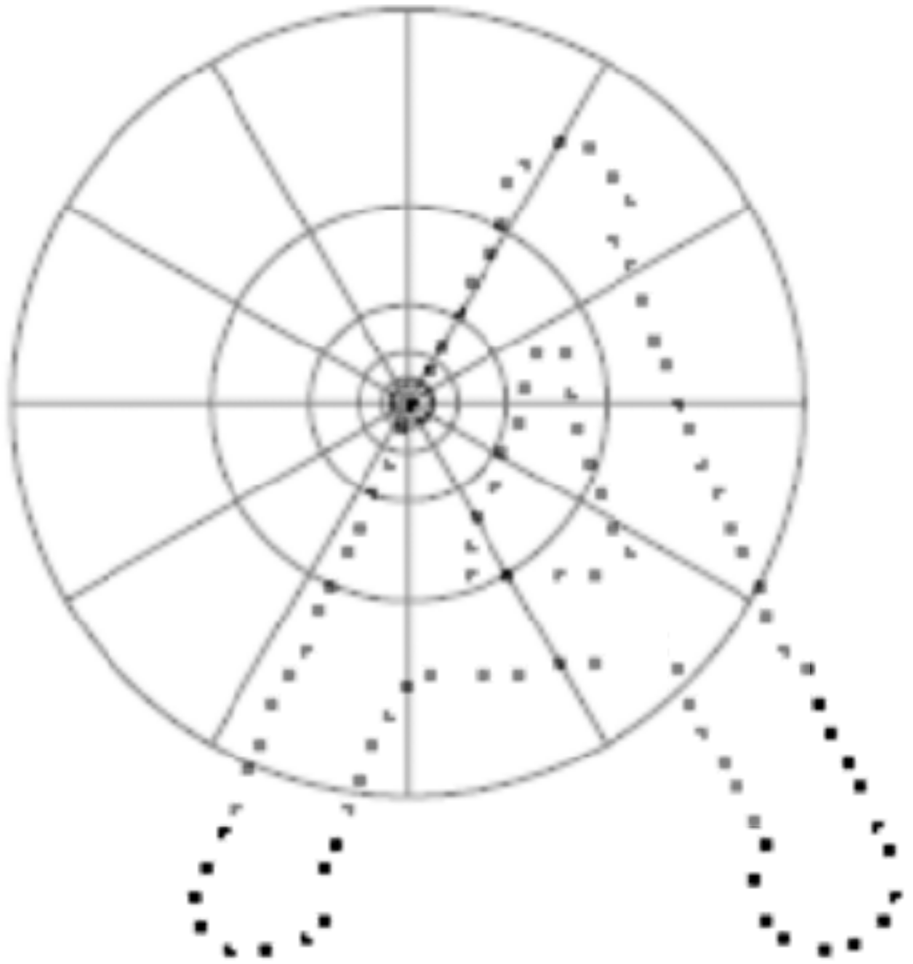
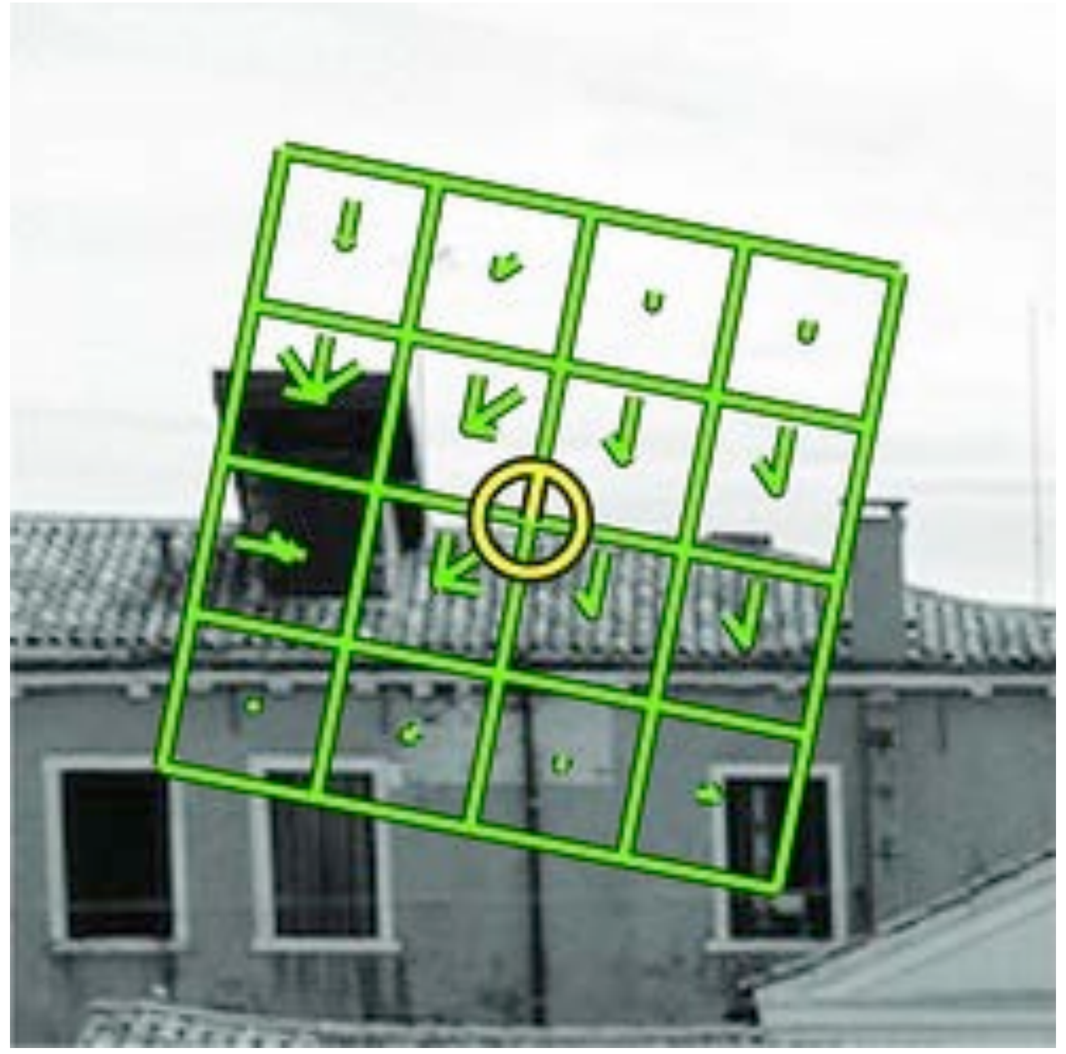


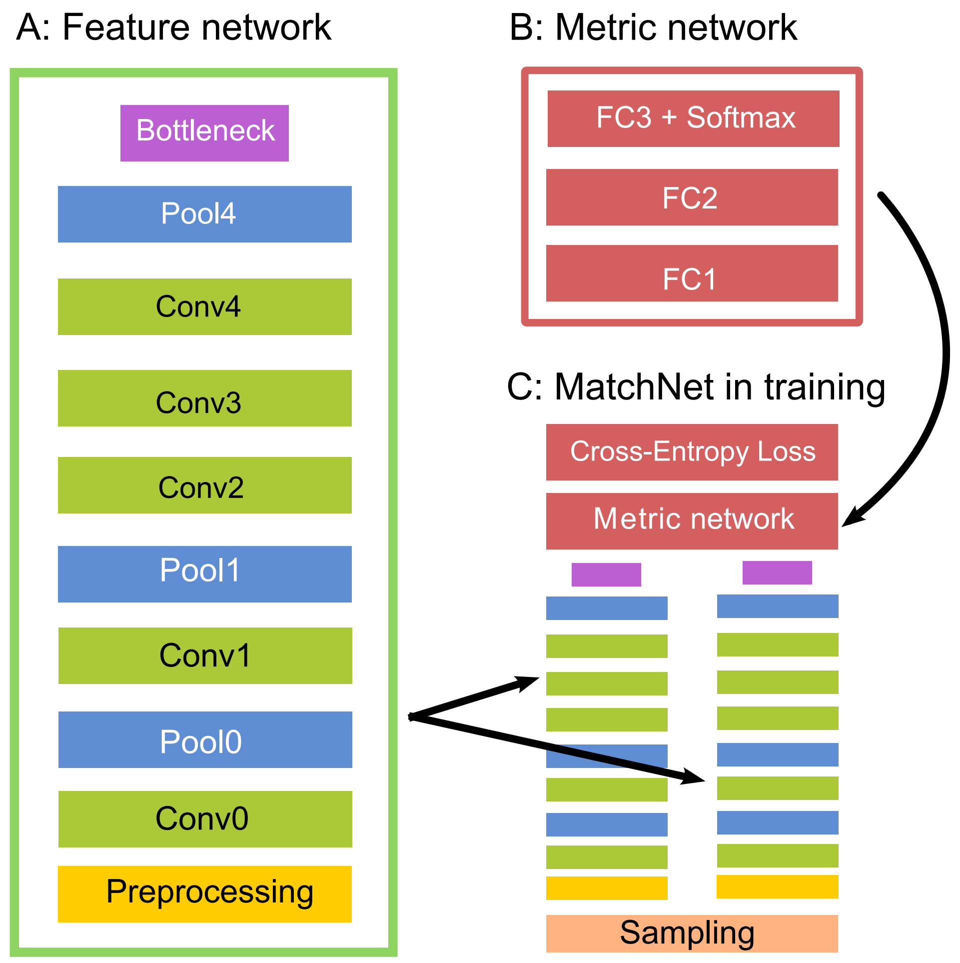
Image Patch



Shape Context



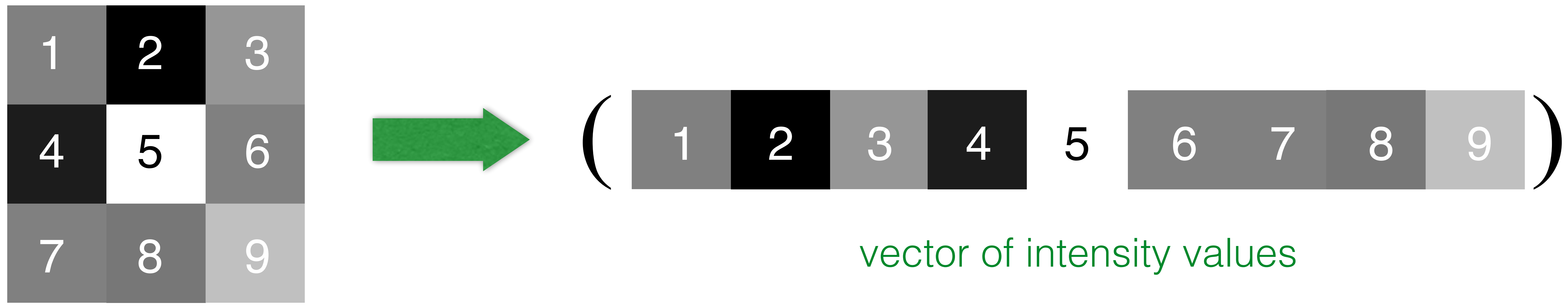
SIFT



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)

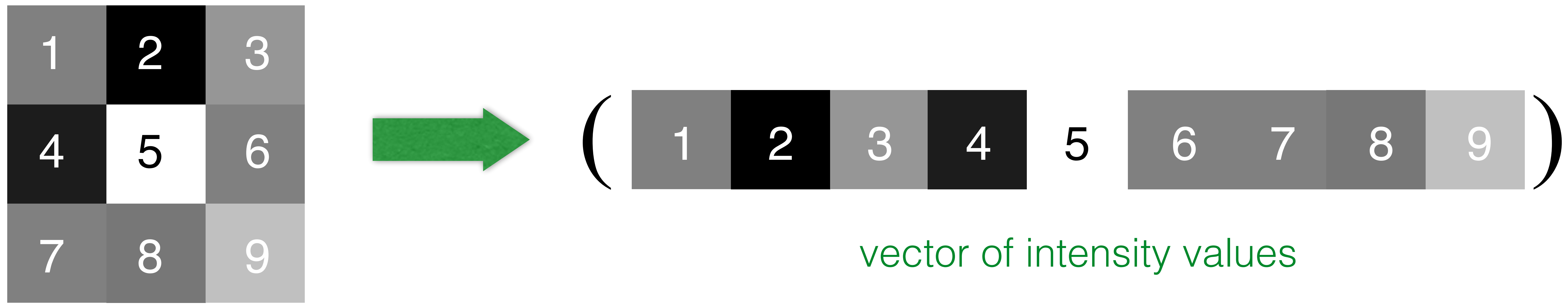
Photometric Transformations

What can we use to deal with this?

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)

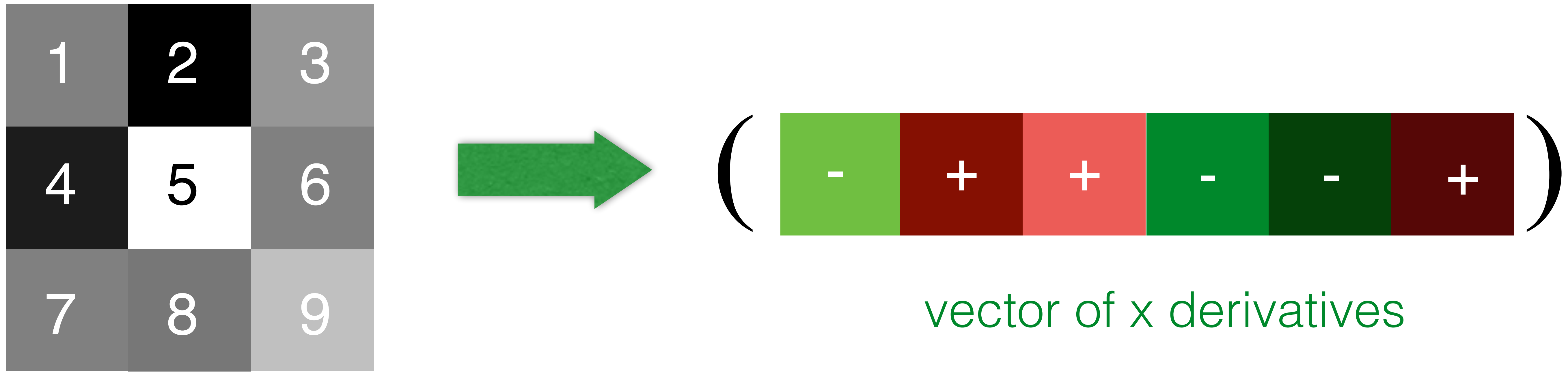
What are the problems?

How can you be less sensitive to absolute intensity values?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

Use pixel differences



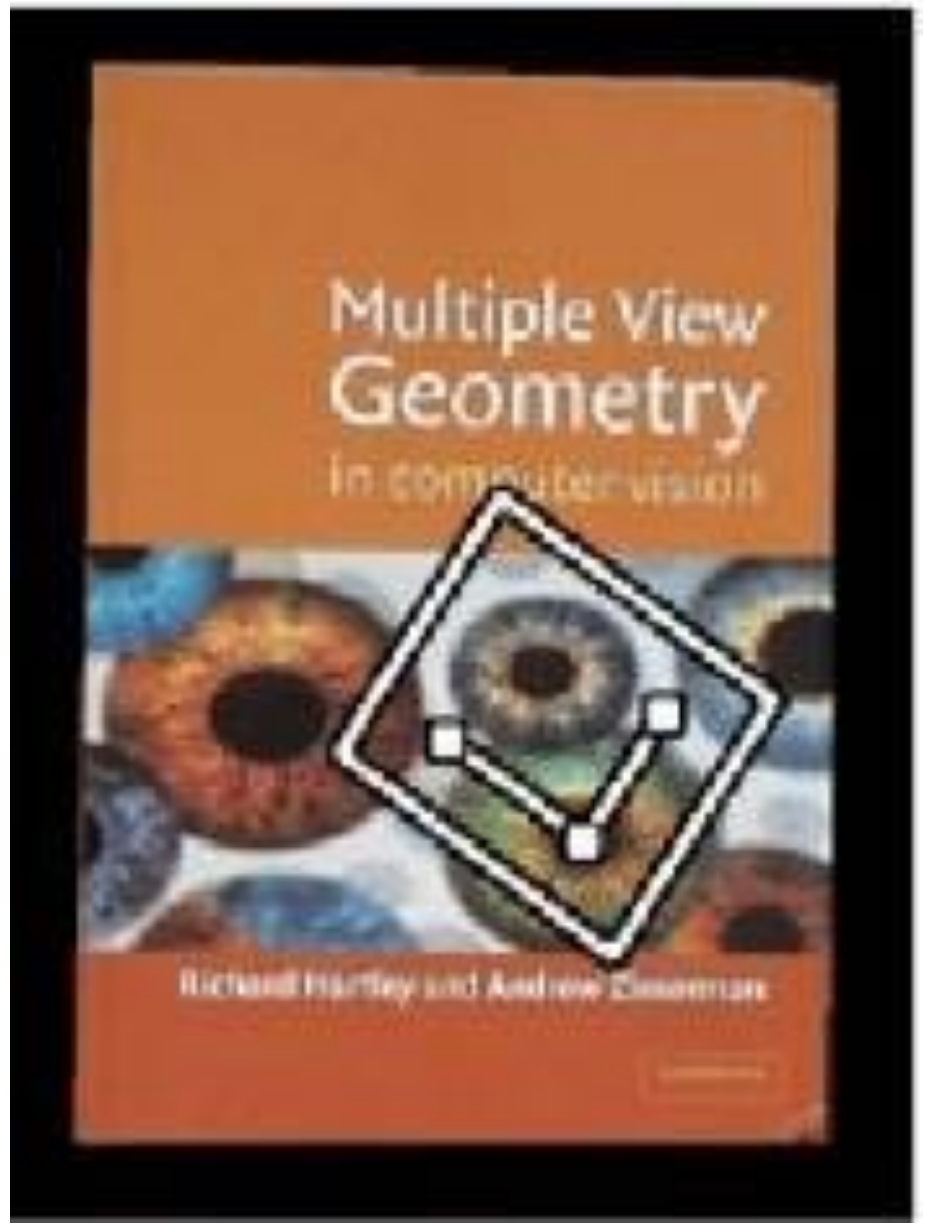
Feature is invariant to absolute intensity values

What are the problems?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Geometric Transformations

How can we deal with this?

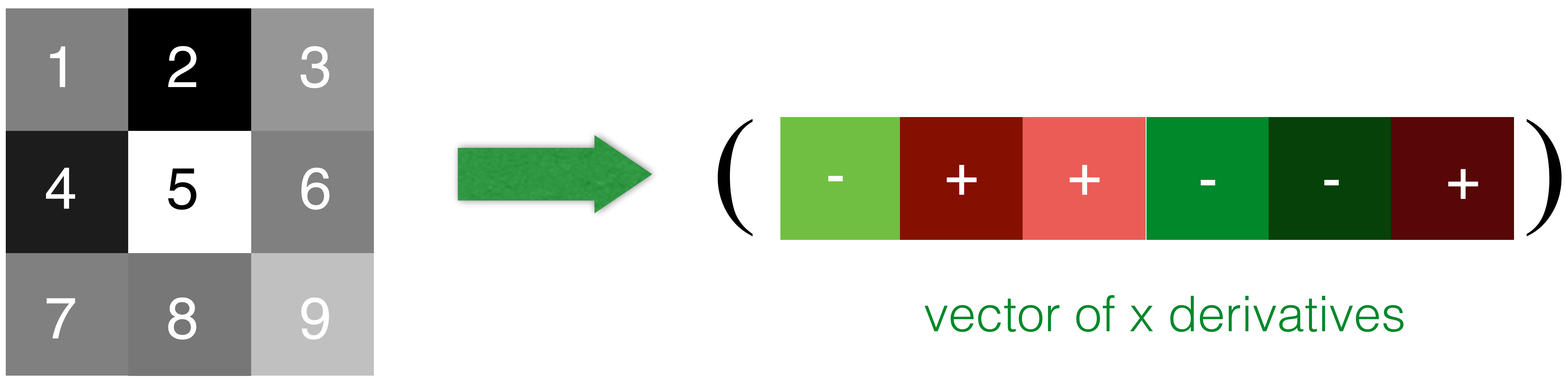


objects will appear at different scales,
translation and rotation

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

Use pixel differences



Feature is invariant to absolute intensity values

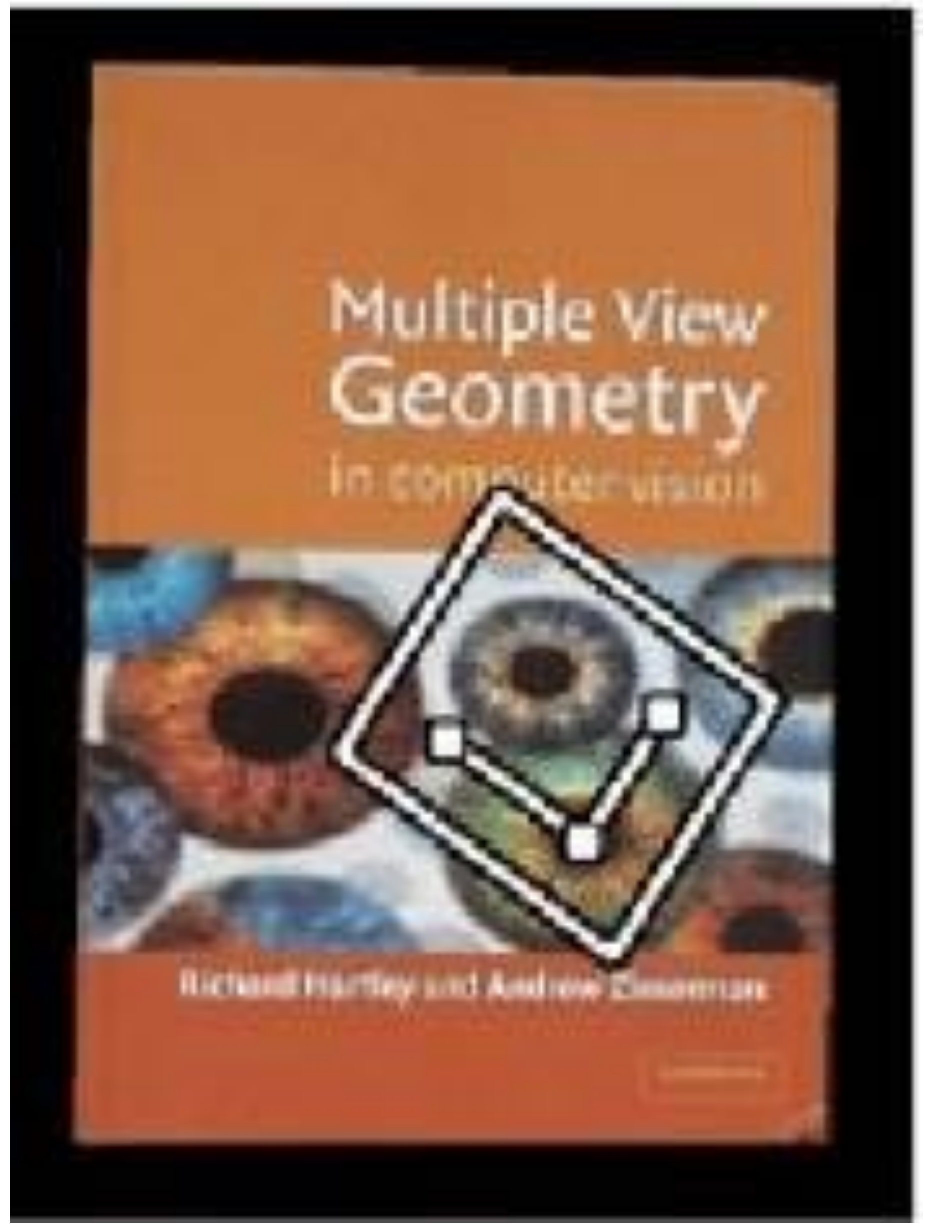
What are the problems?

How can you be less sensitive to deformations?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Geometric Transformations

How can we deal with this?



objects will appear at different scales,
translation and rotation

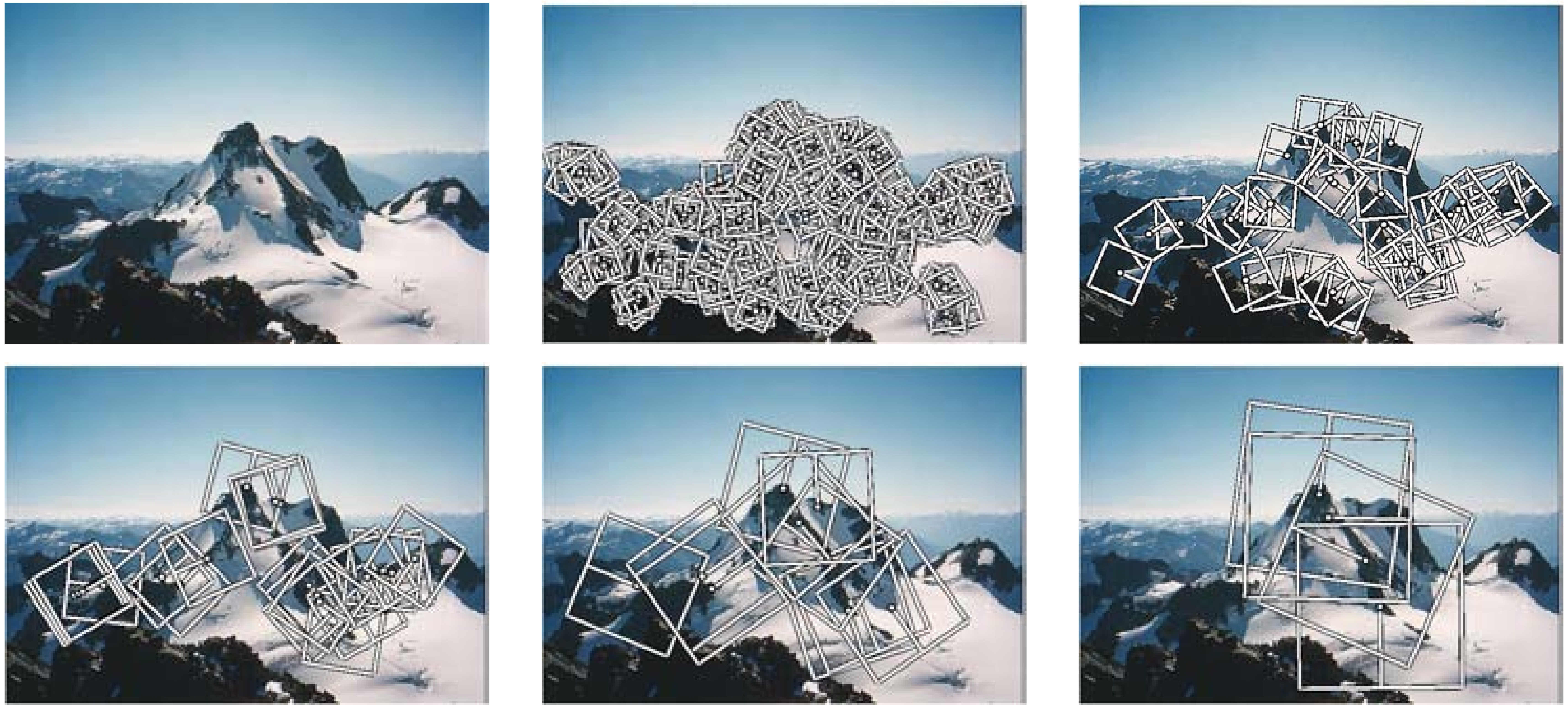
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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**

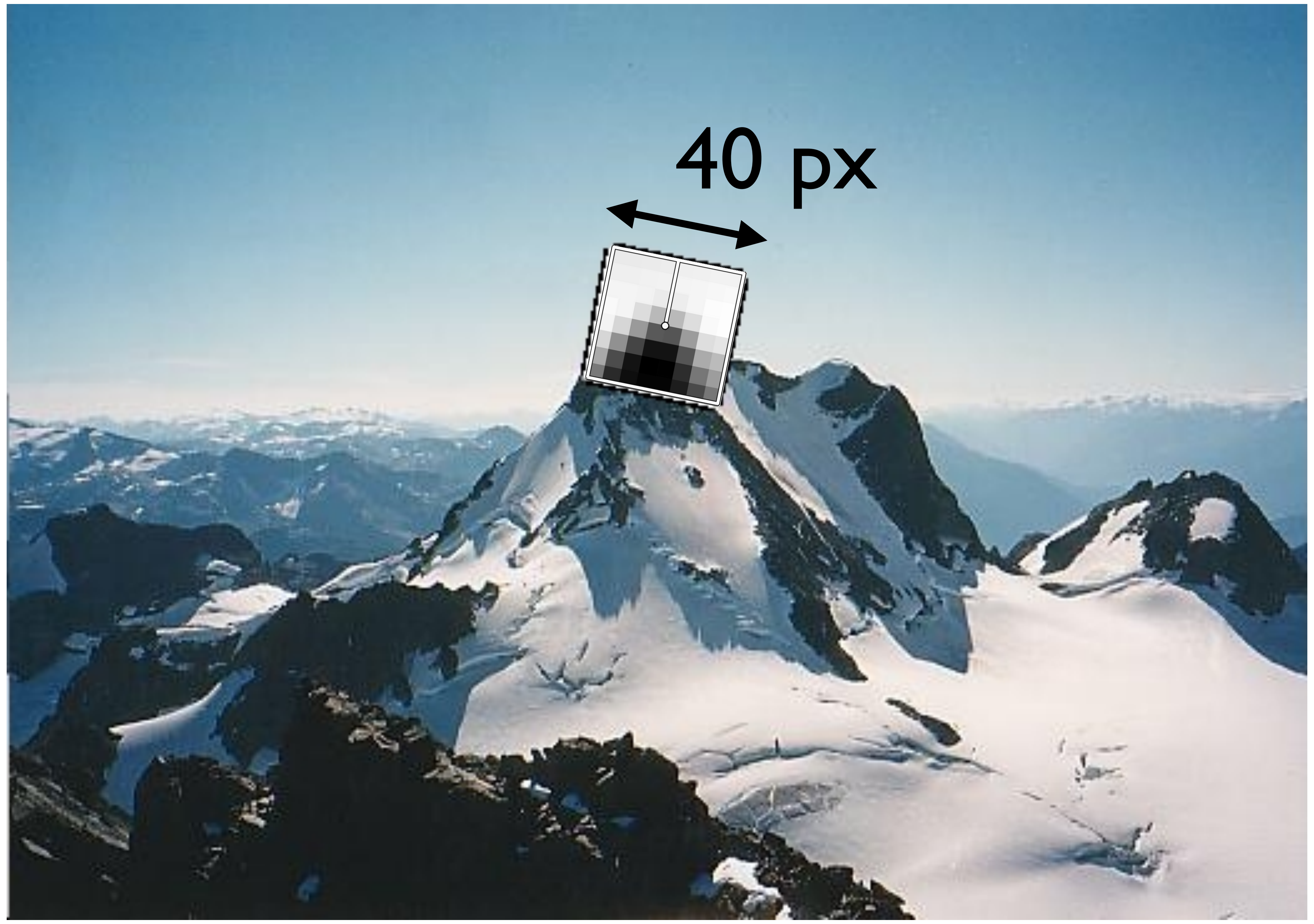
A common approach is to detect a local scale and orientation for each feature point

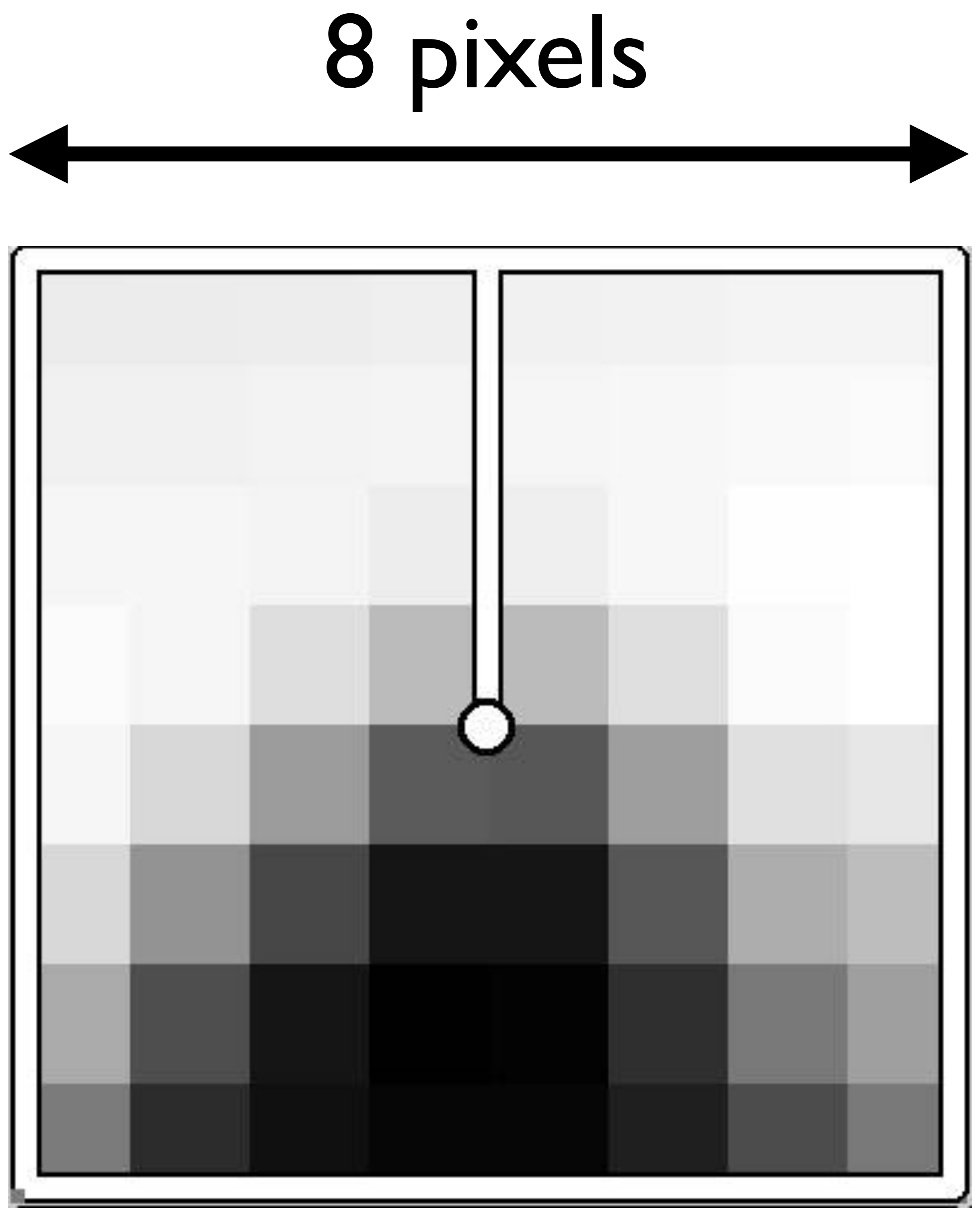


e.g., extract Harris at multiple scales and align to the local gradient

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

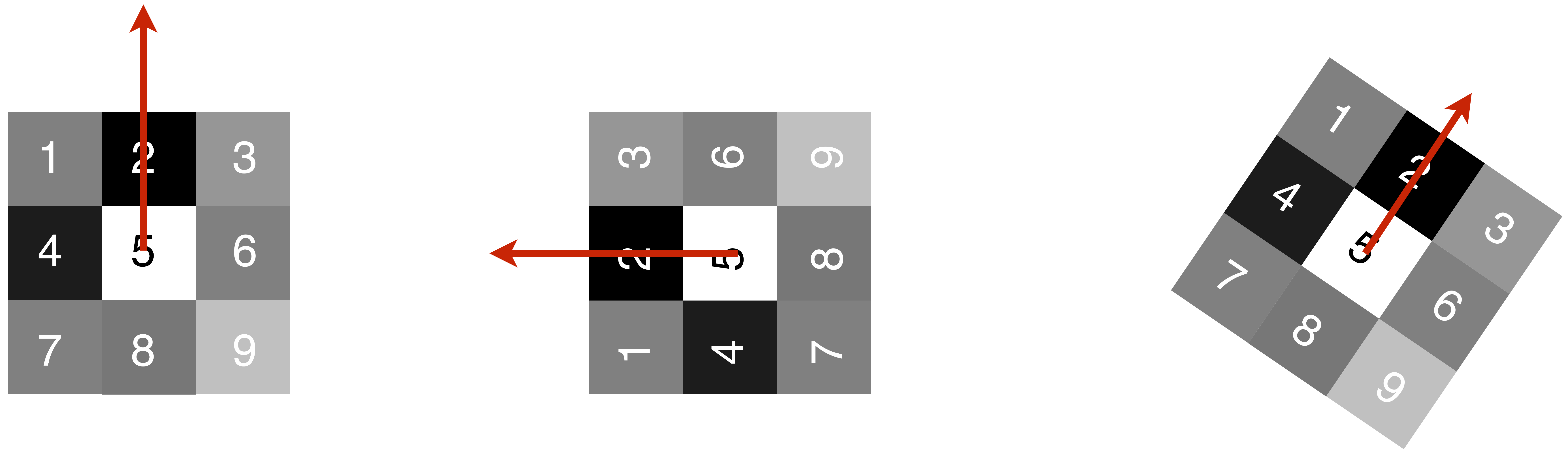
A common approach is to detect a local scale and orientation for each feature point





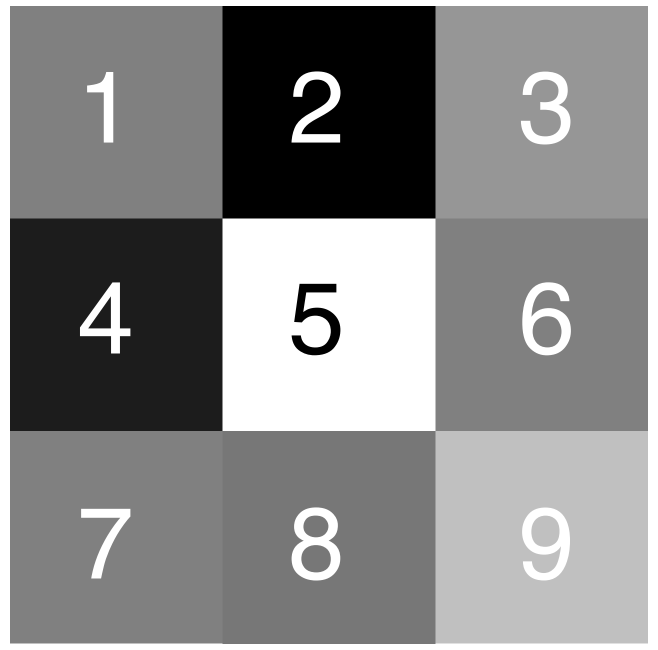
e.g., extract Harris at multiple scales and align to the local gradient

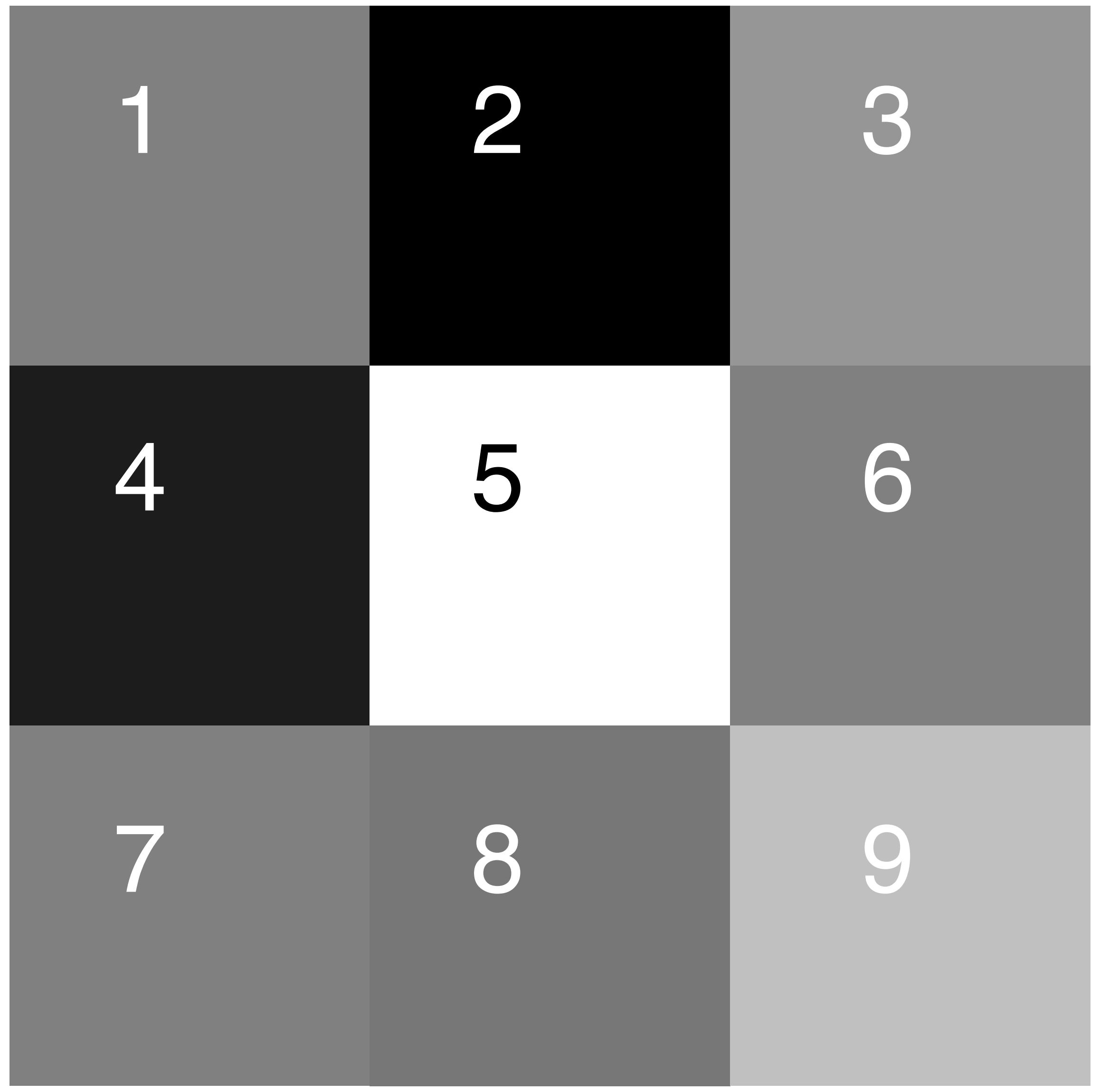
Strategy #1: Compute Features in Local Coordinate Frame

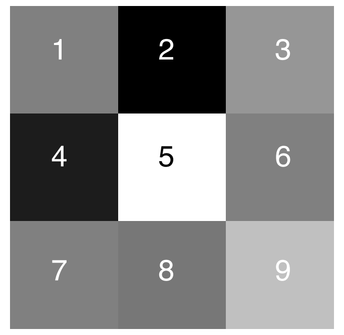


First rotate to canonical frame of reference (e.g., align feature direction with y-axis) and only then compute a feature representation

Strategy #1: Compute Features in Local Coordinate Frame



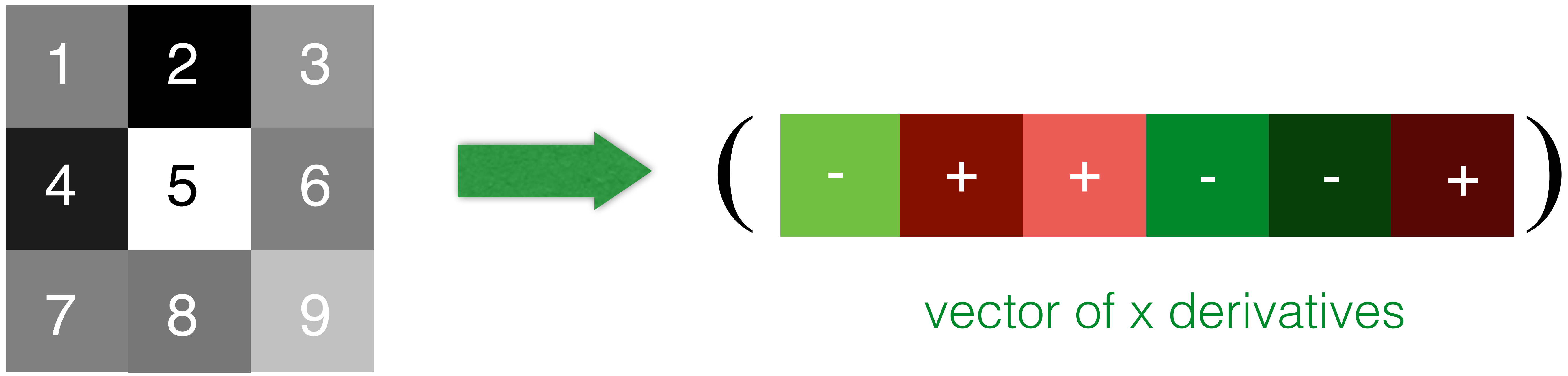




First scale to canonical frame of reference and only then compute a feature representation

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	$\bigtriangledown^2 G$, 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

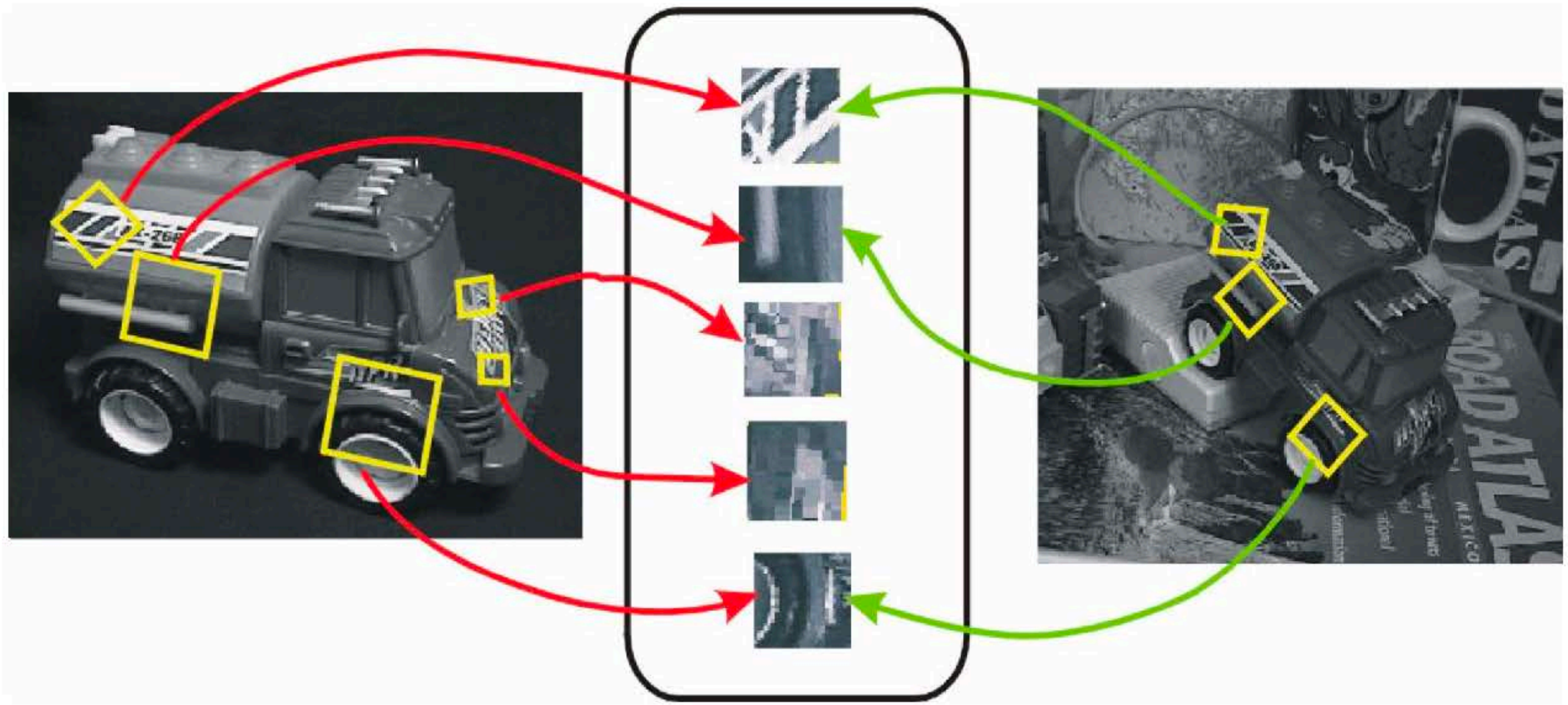
— Location recognition

— Digital camera panoramas

— 3D scene modeling, augmented reality

David Lowe's Invariant Local Features

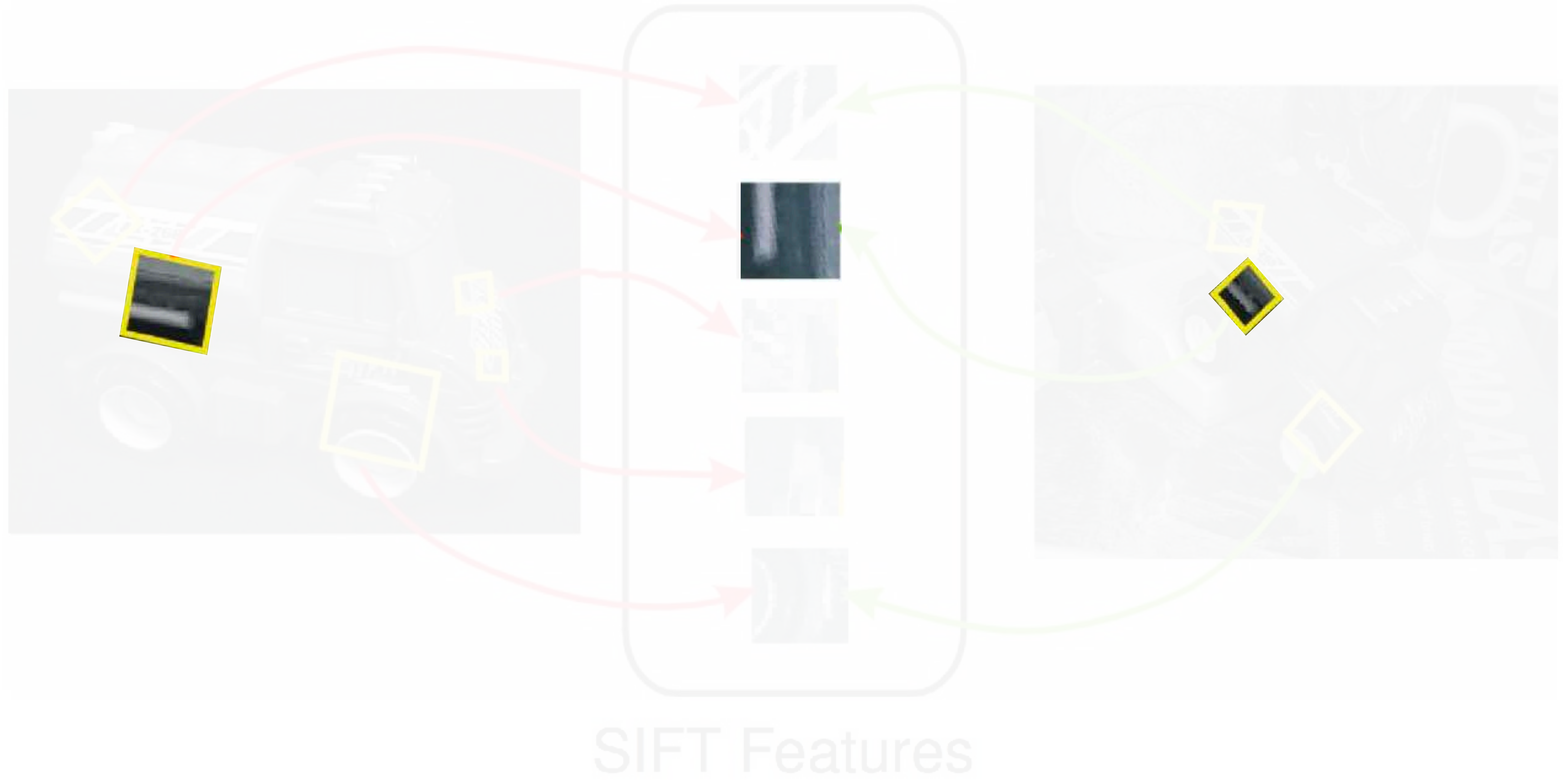
Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



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



Advantages of Invariant Local Features

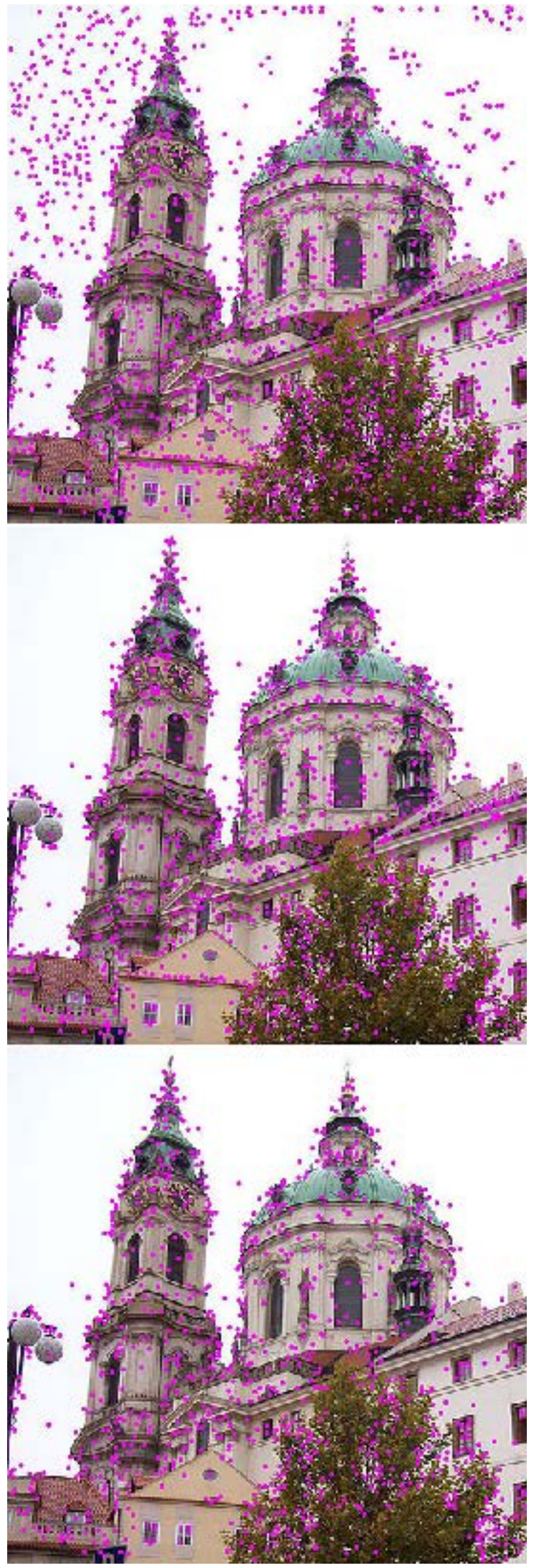
Locality: features are local, so robust to occlusion and clutter (no prior segmentation)

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**)



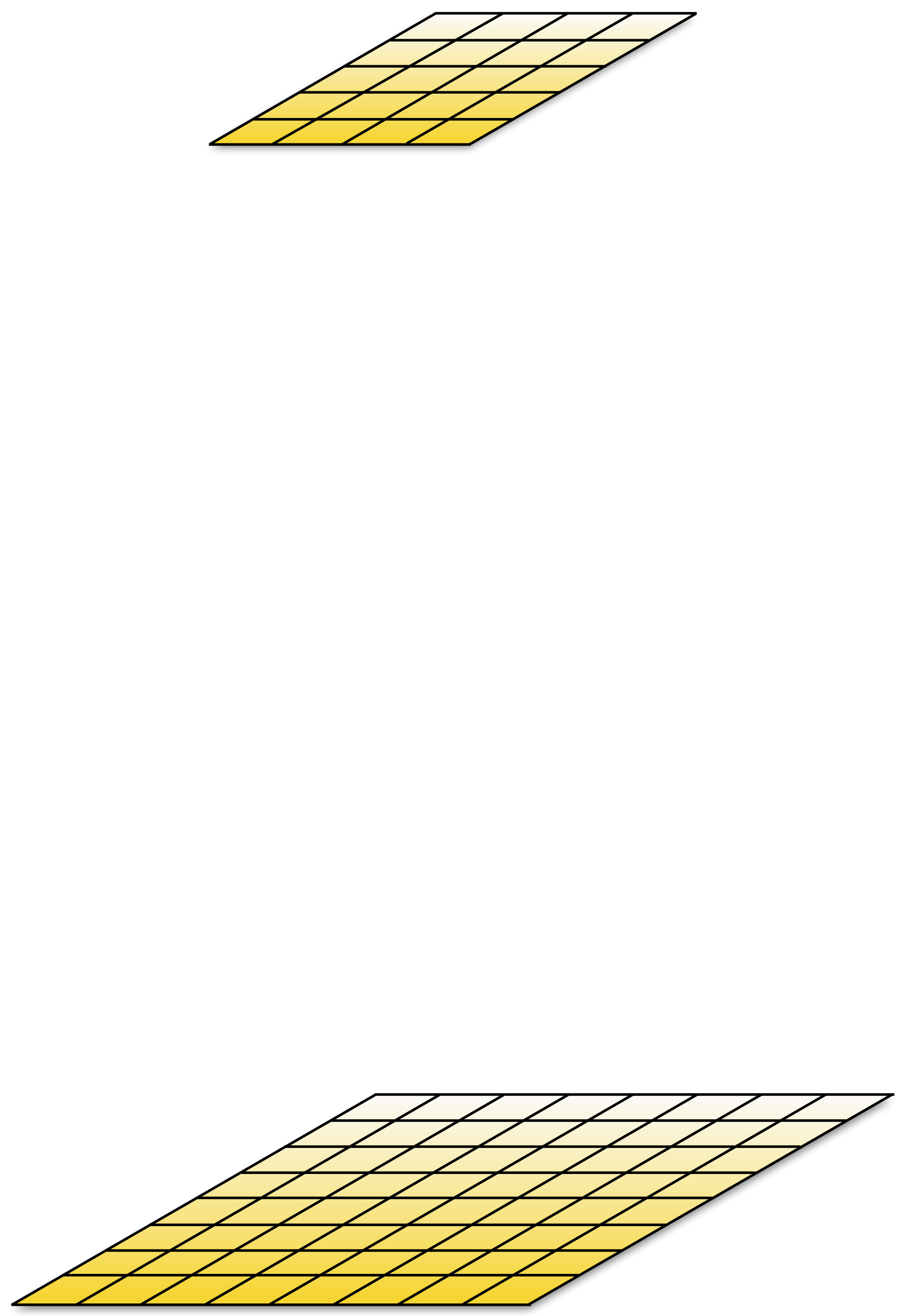
SIFT describes both a **detector** and **descriptor**

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

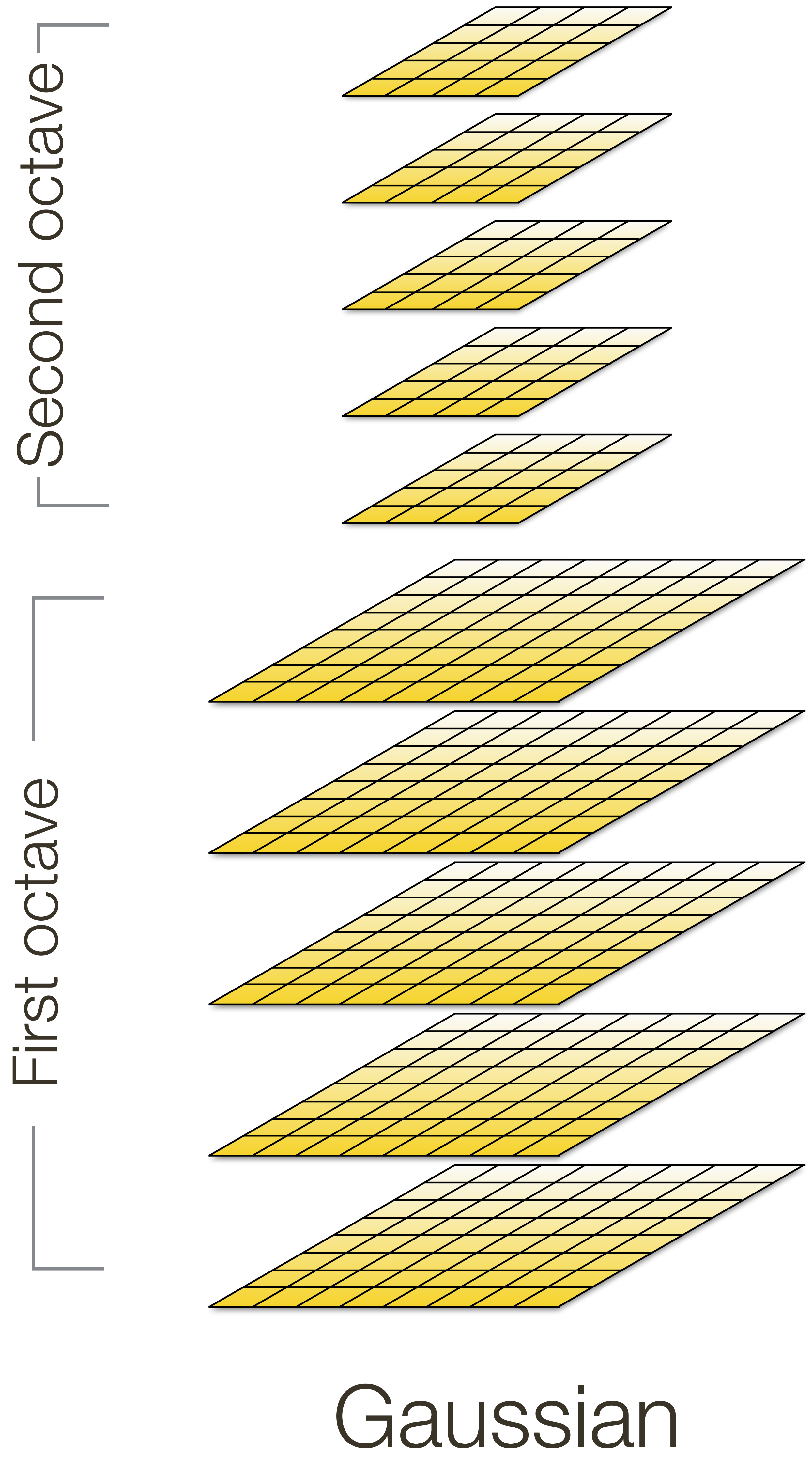
Half the size



Gaussian

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

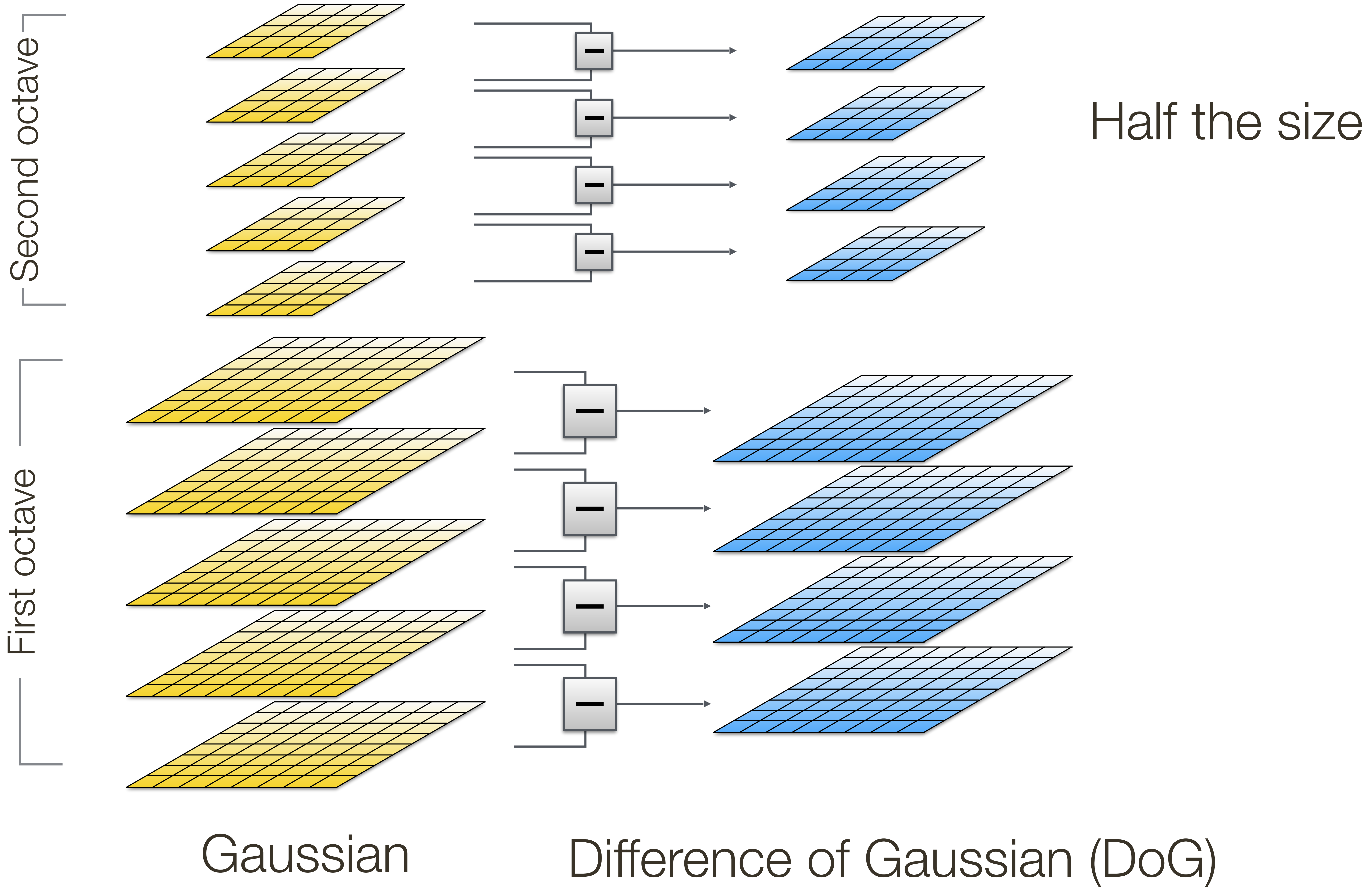
1. Multi-scale Extrema Detection



Half the size

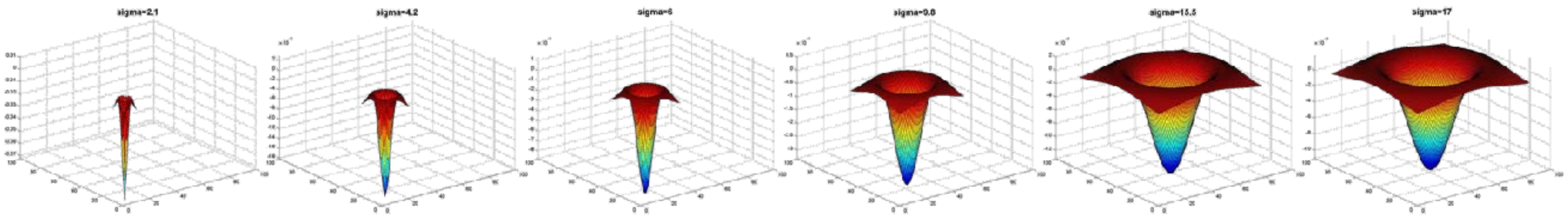
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

1. Multi-scale Extrema Detection



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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



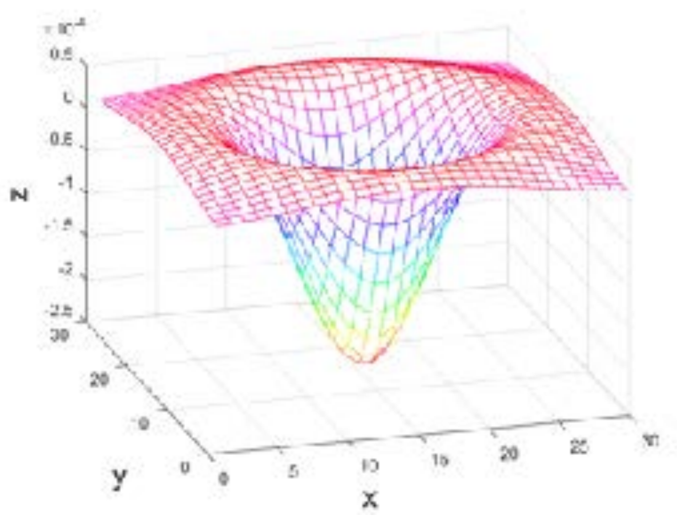
Full size

3/4 size

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

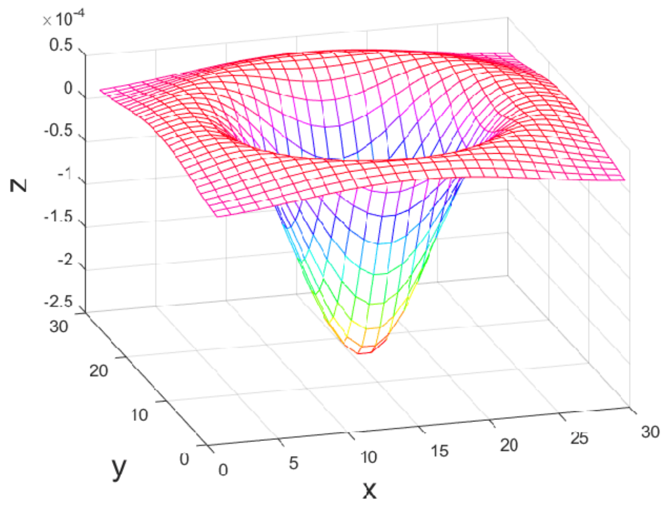
Searching over **Scale**-space

σ

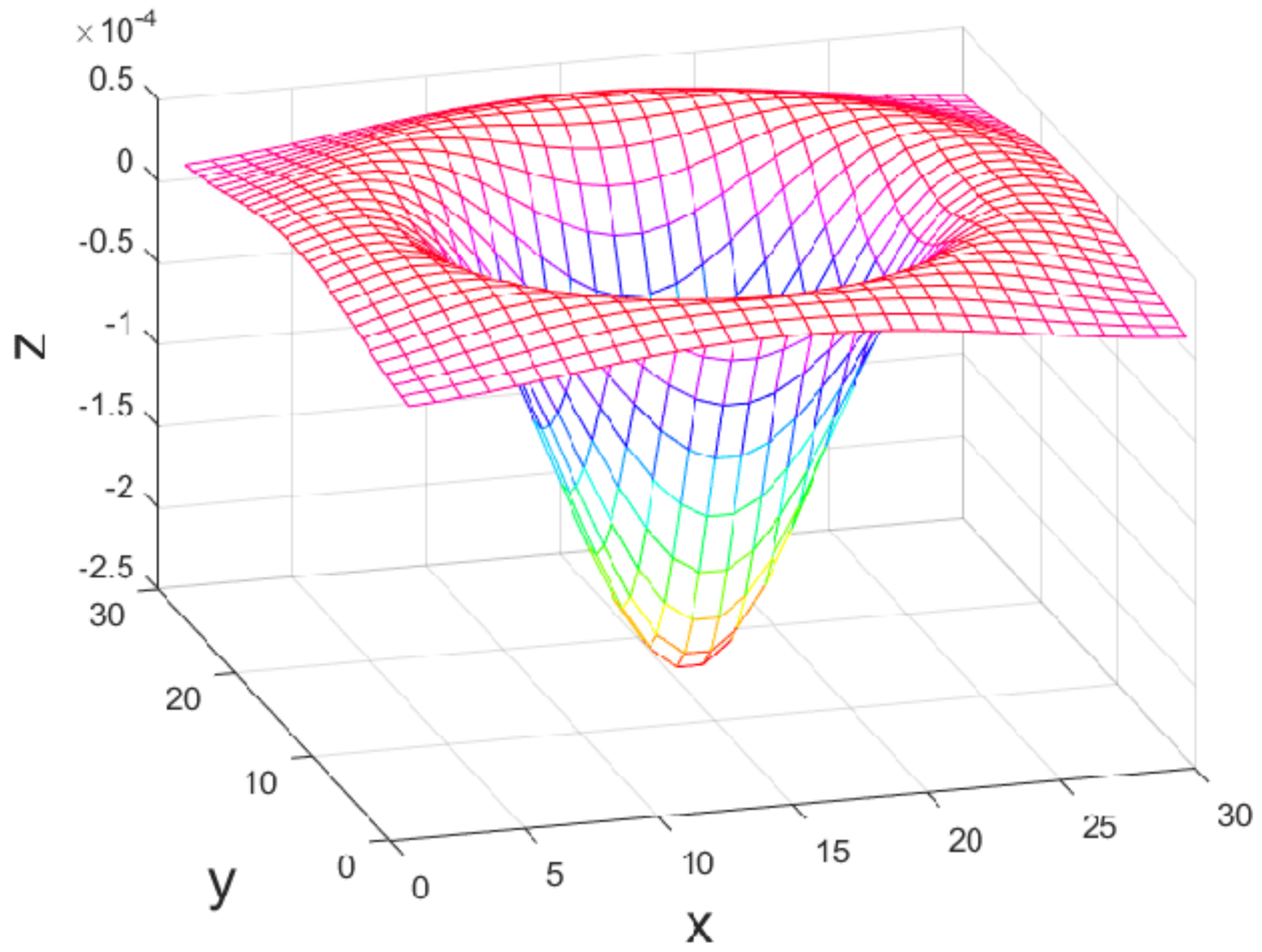


Searching over **Scale**-space

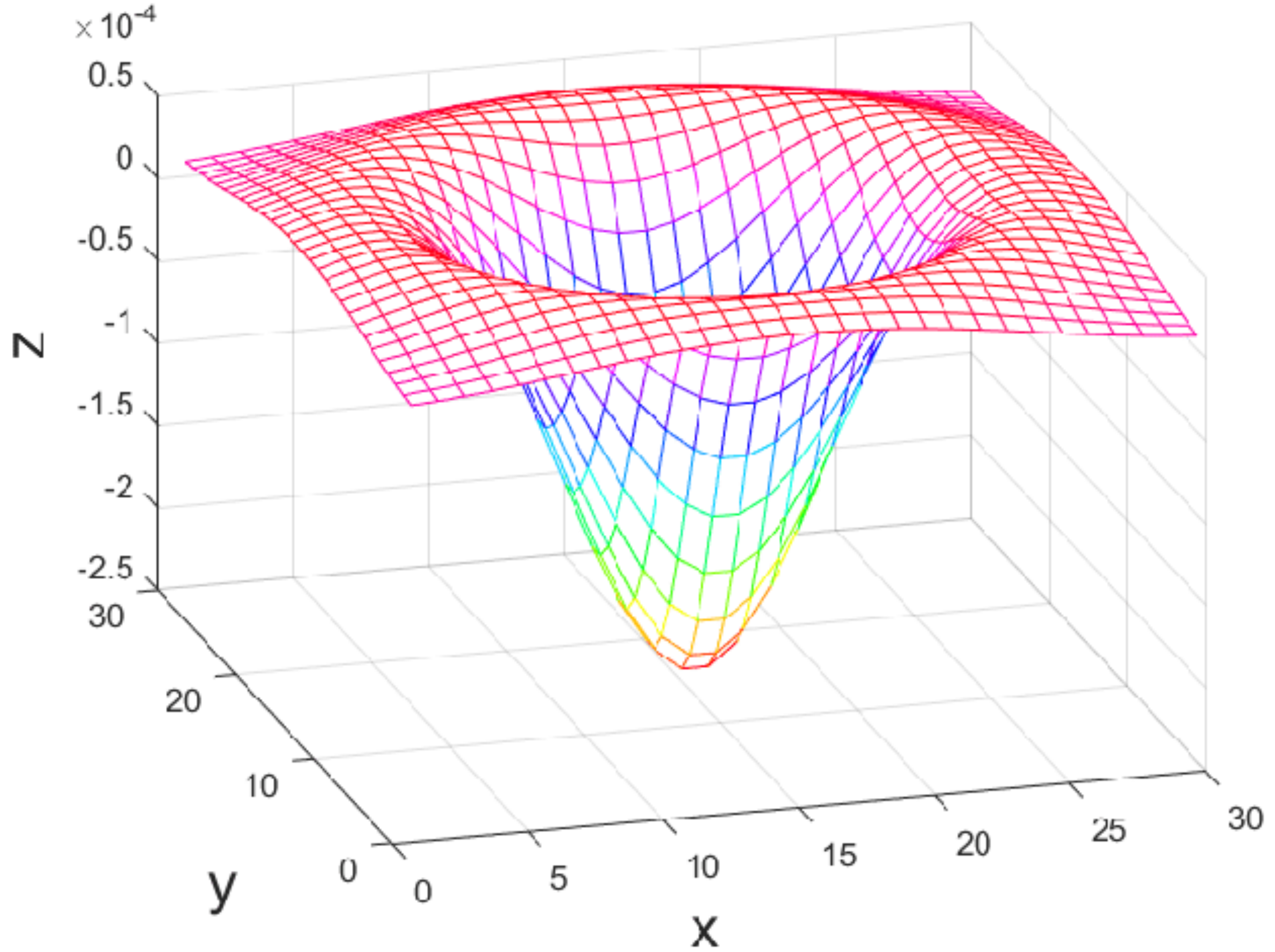
σ

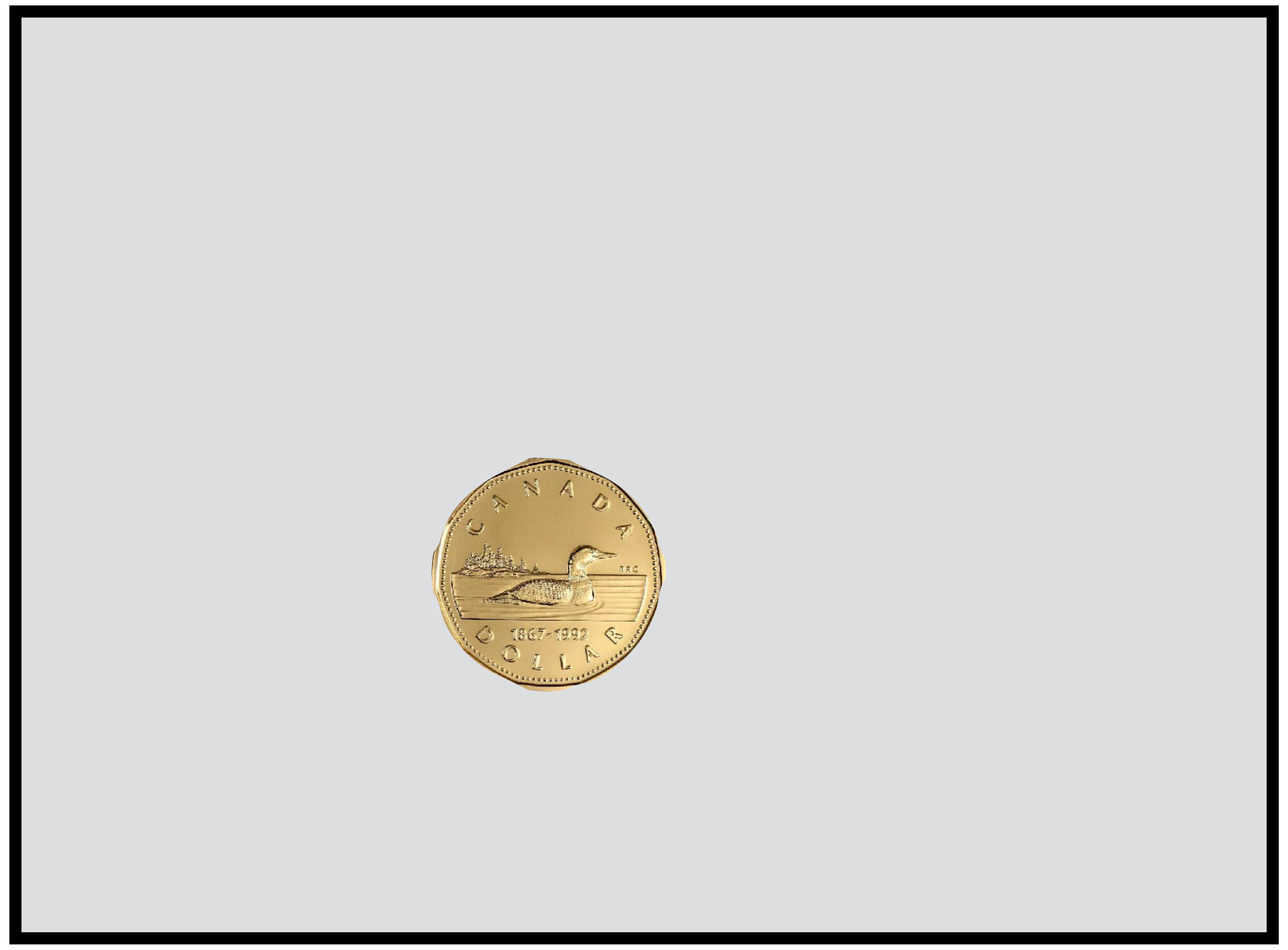


$\sigma' = 2\sigma$



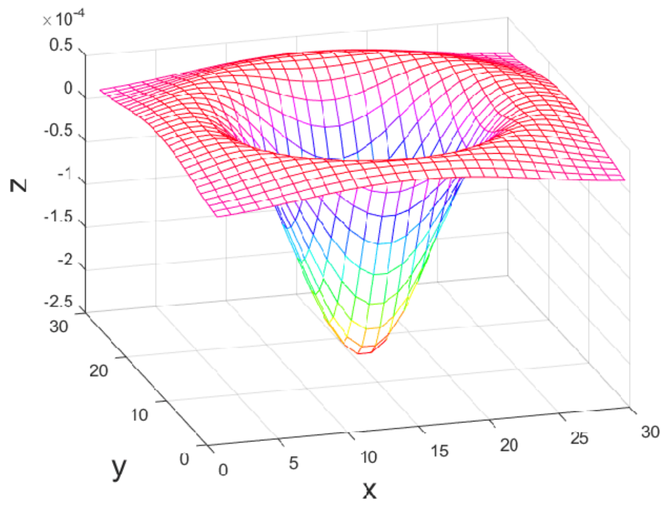
$\sigma' = 3\sigma$



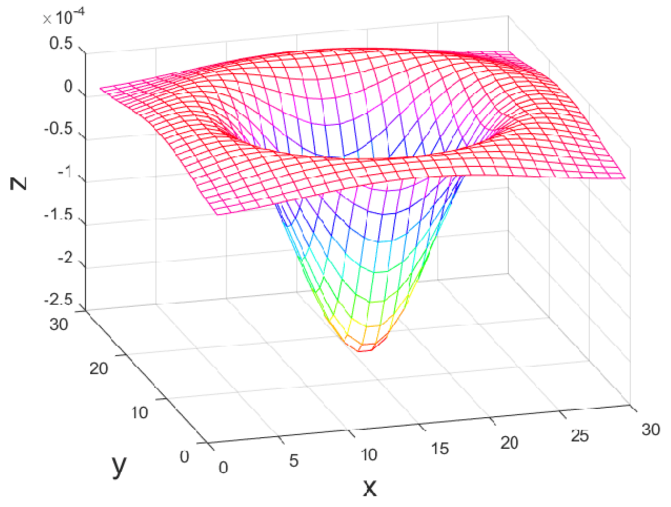


Searching over **Scale**-space

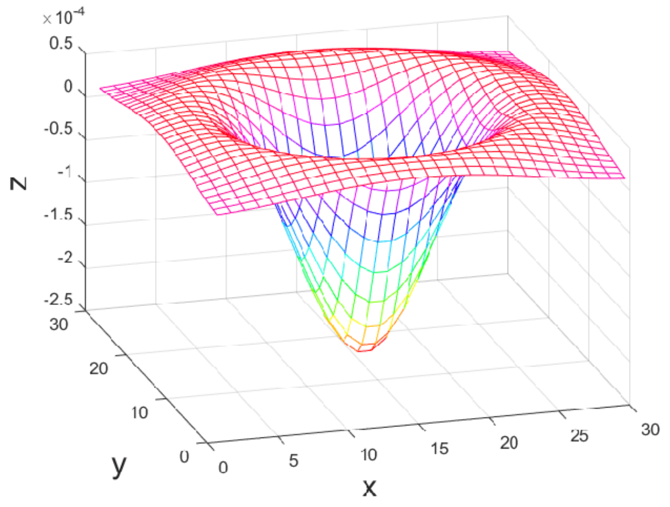
σ

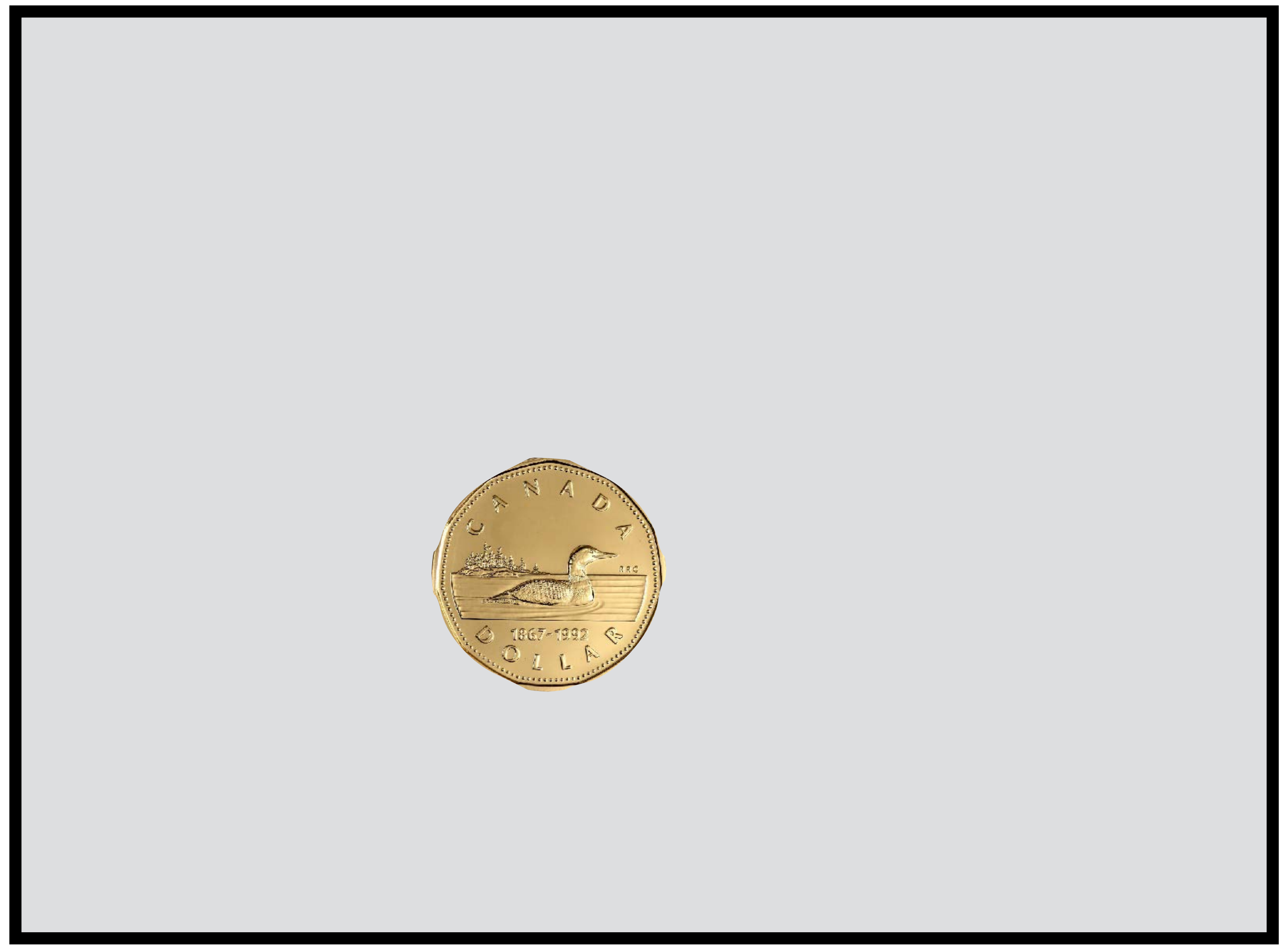


σ



σ

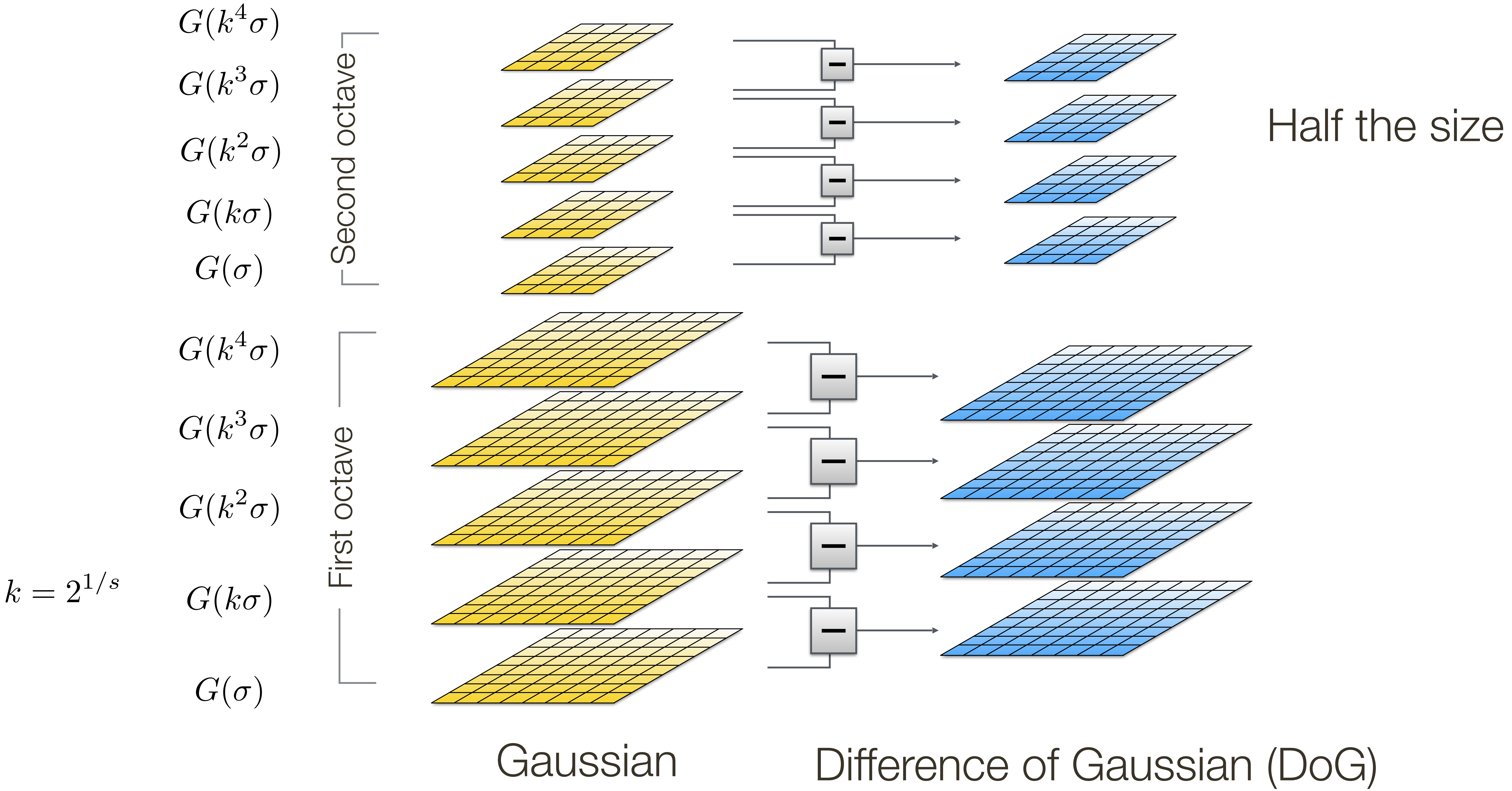




$s = 0.5$

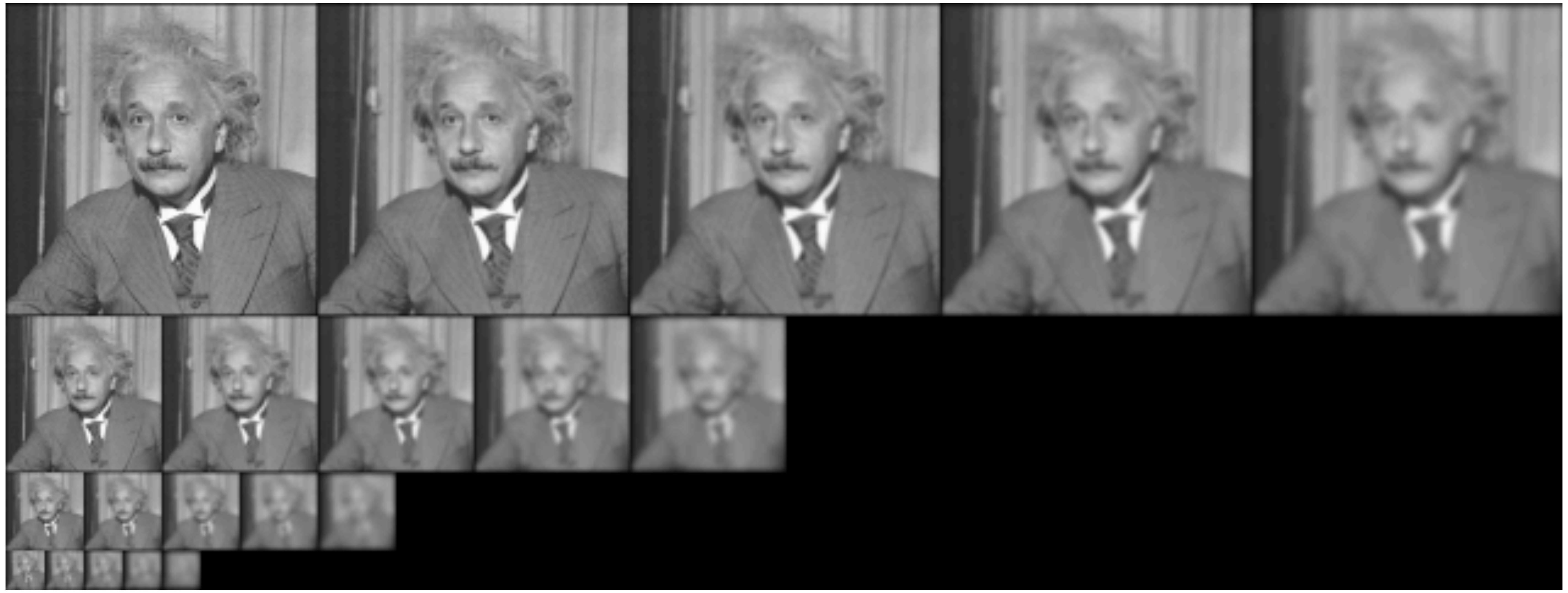
$s = 0.33$

1. Multi-scale Extrema Detection



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

1. Multi-scale Extrema Detection

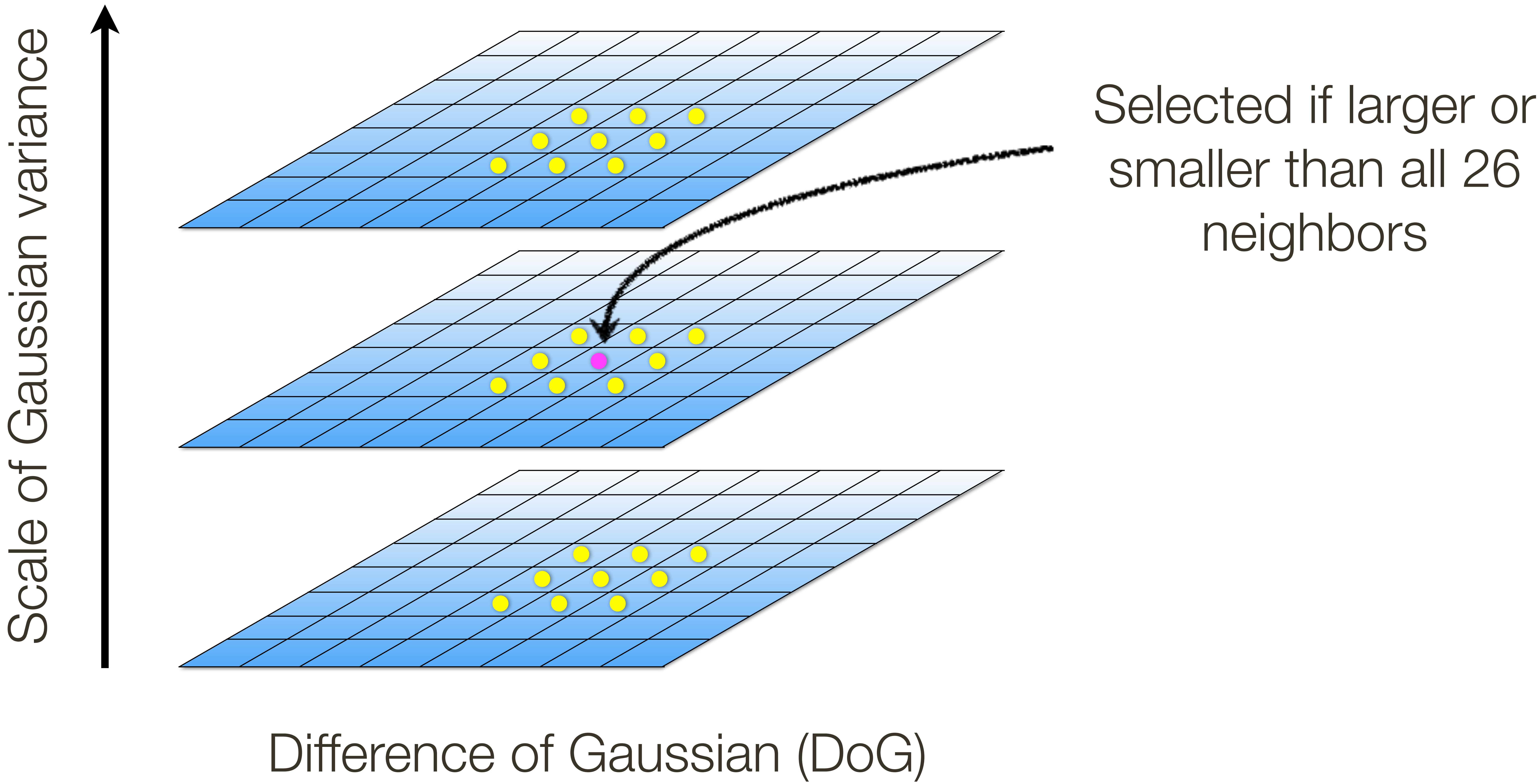

Gaussian


Laplacian

1. Multi-scale Extrema Detection

Detect maxima and minima of Difference of Gaussian in scale space

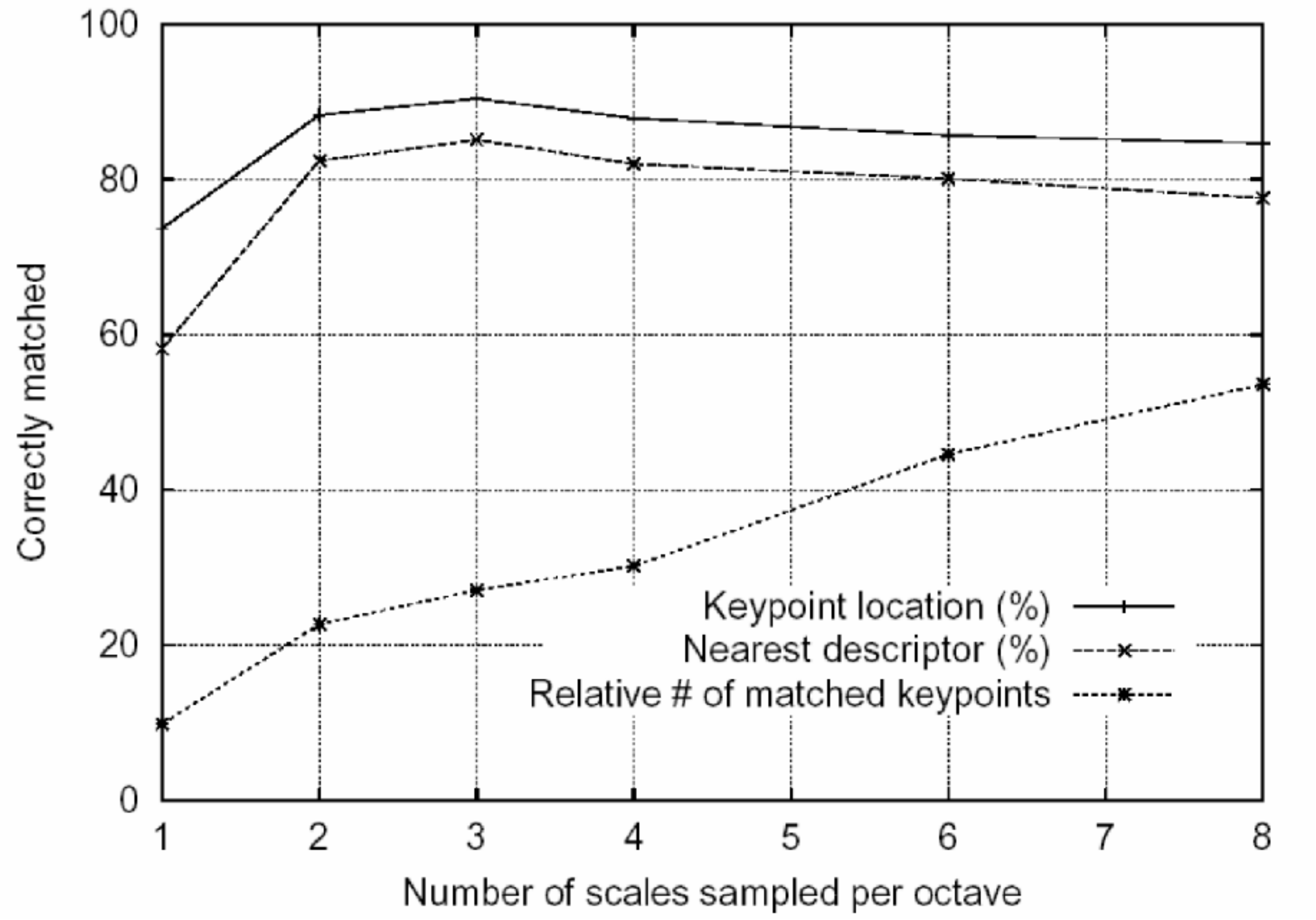
$$\sigma = 2^{1/s}$$



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

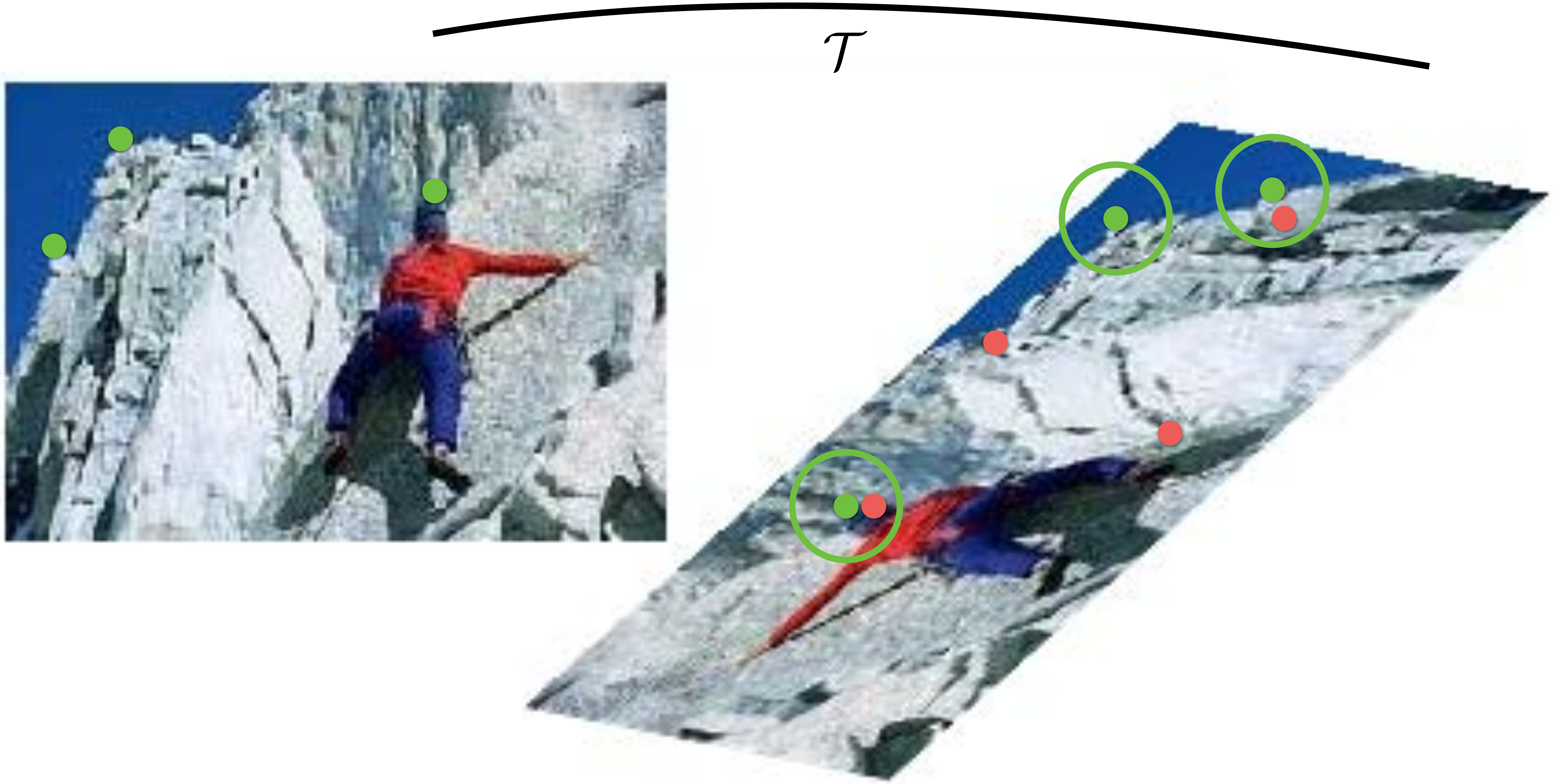
1. Multi-scale Extrema Detection — Sampling Frequency

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



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

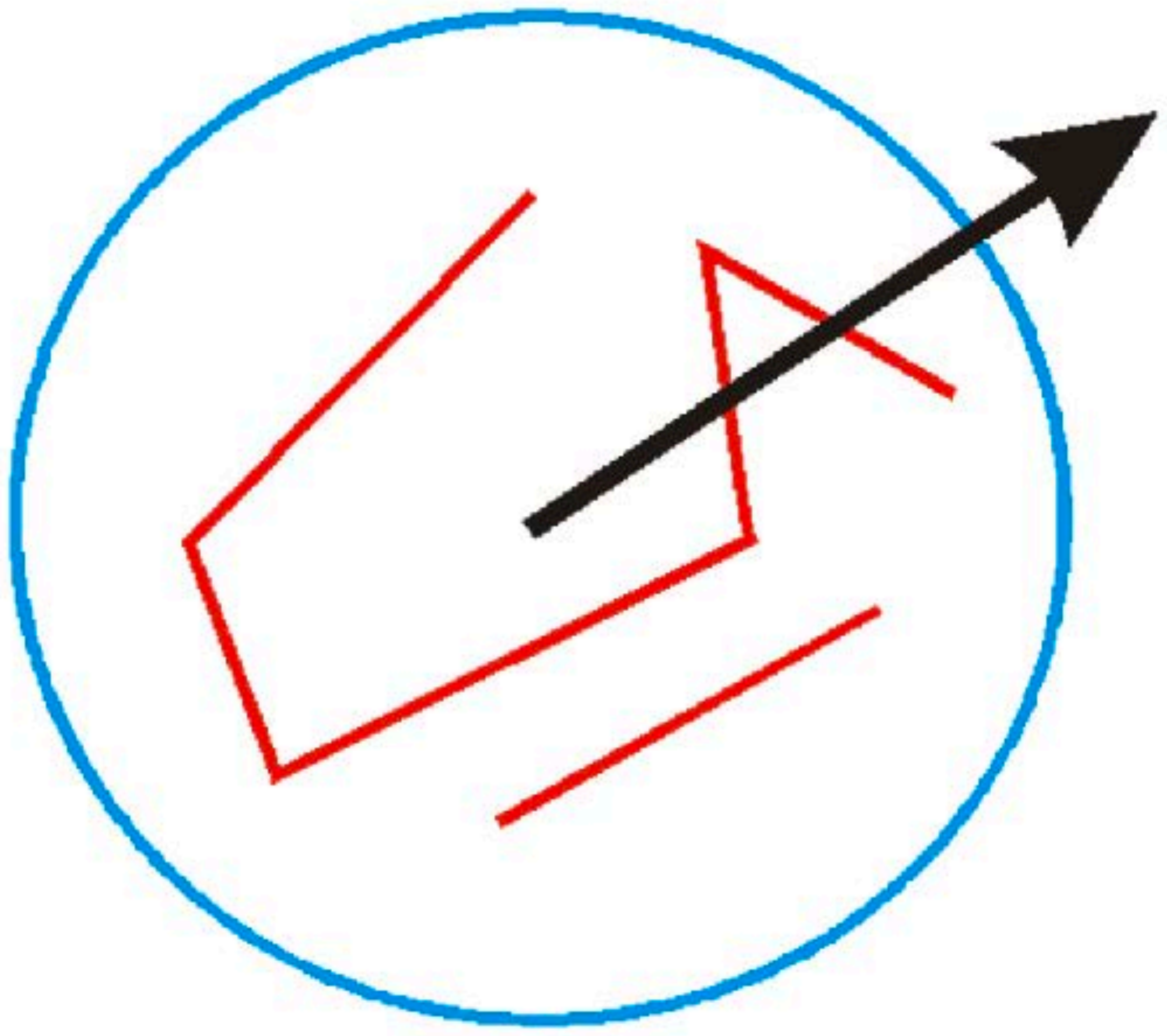
(d) 536 left after testing ratio of principal curvatures

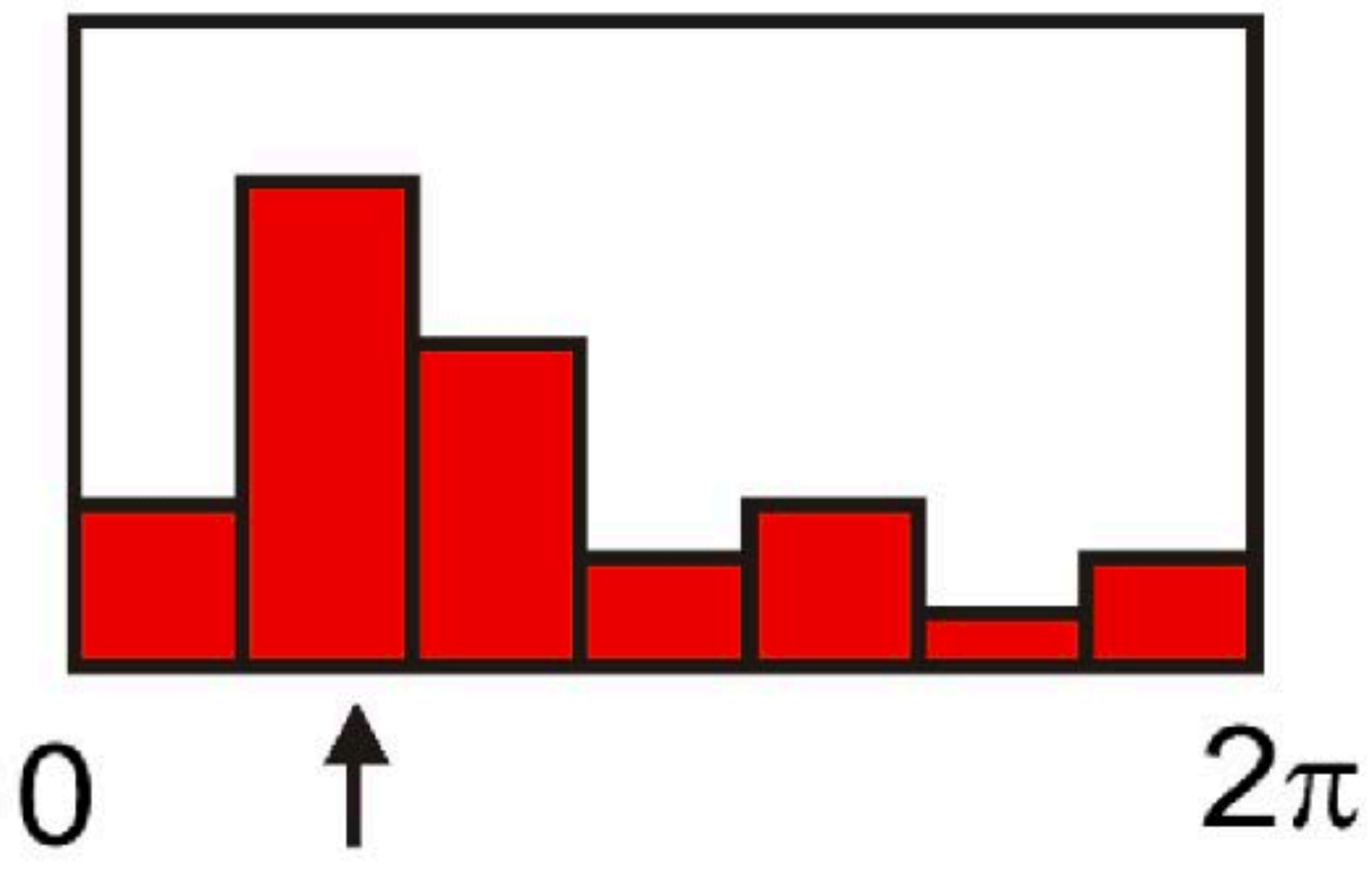
3. Orientation Assignment

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

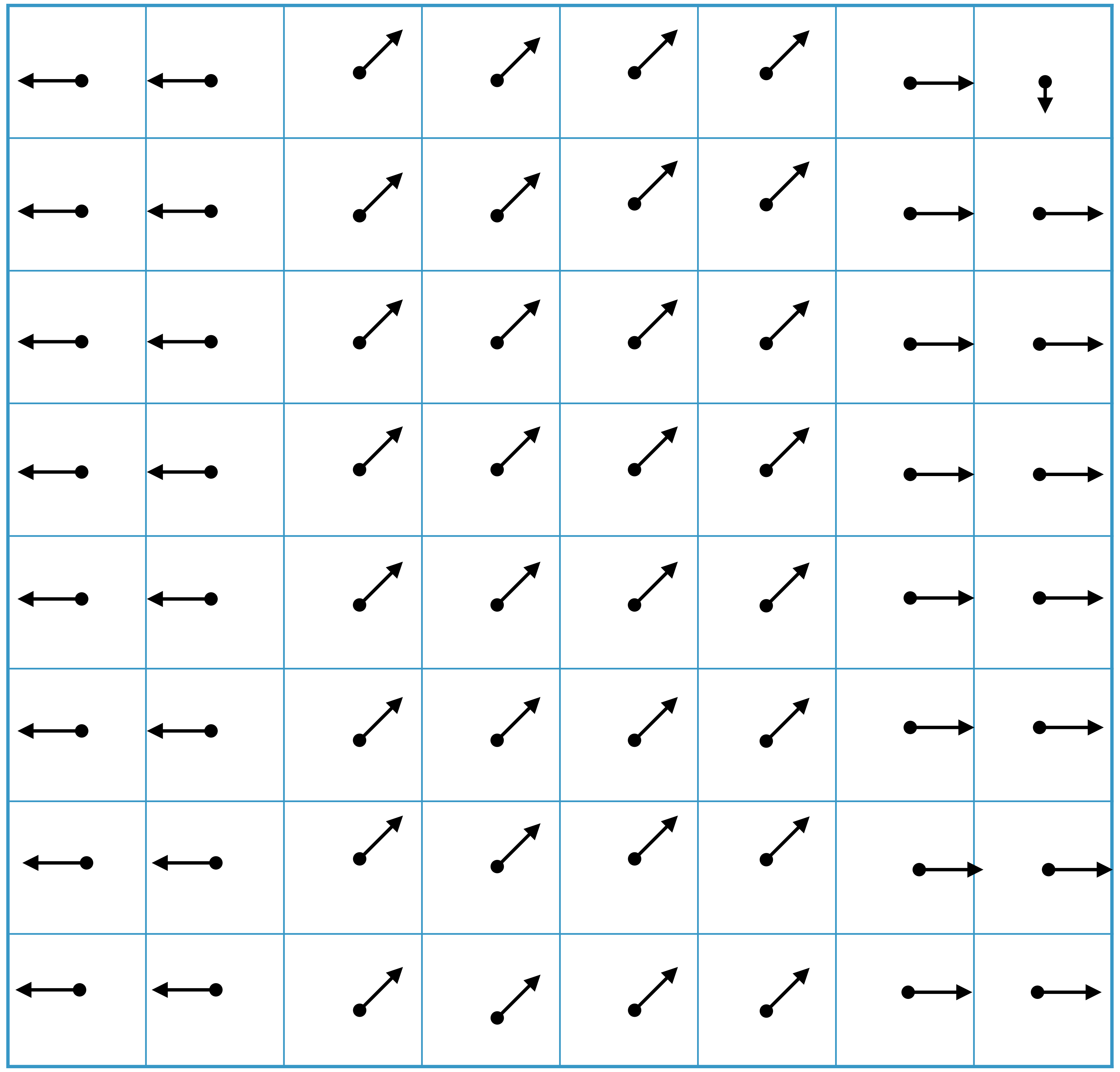
— Assign **canonical orientation** at peak of smoothed histogram

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



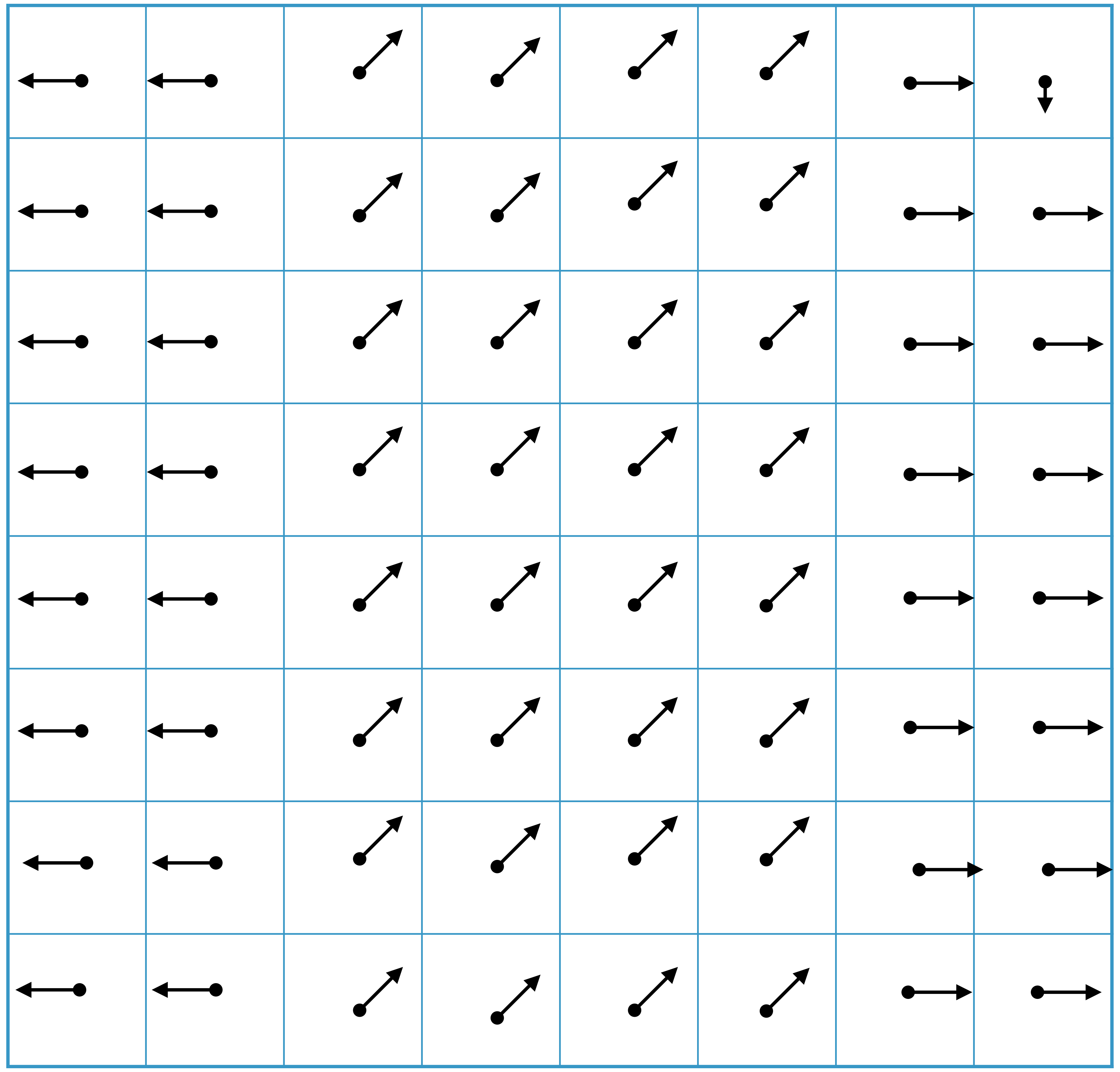


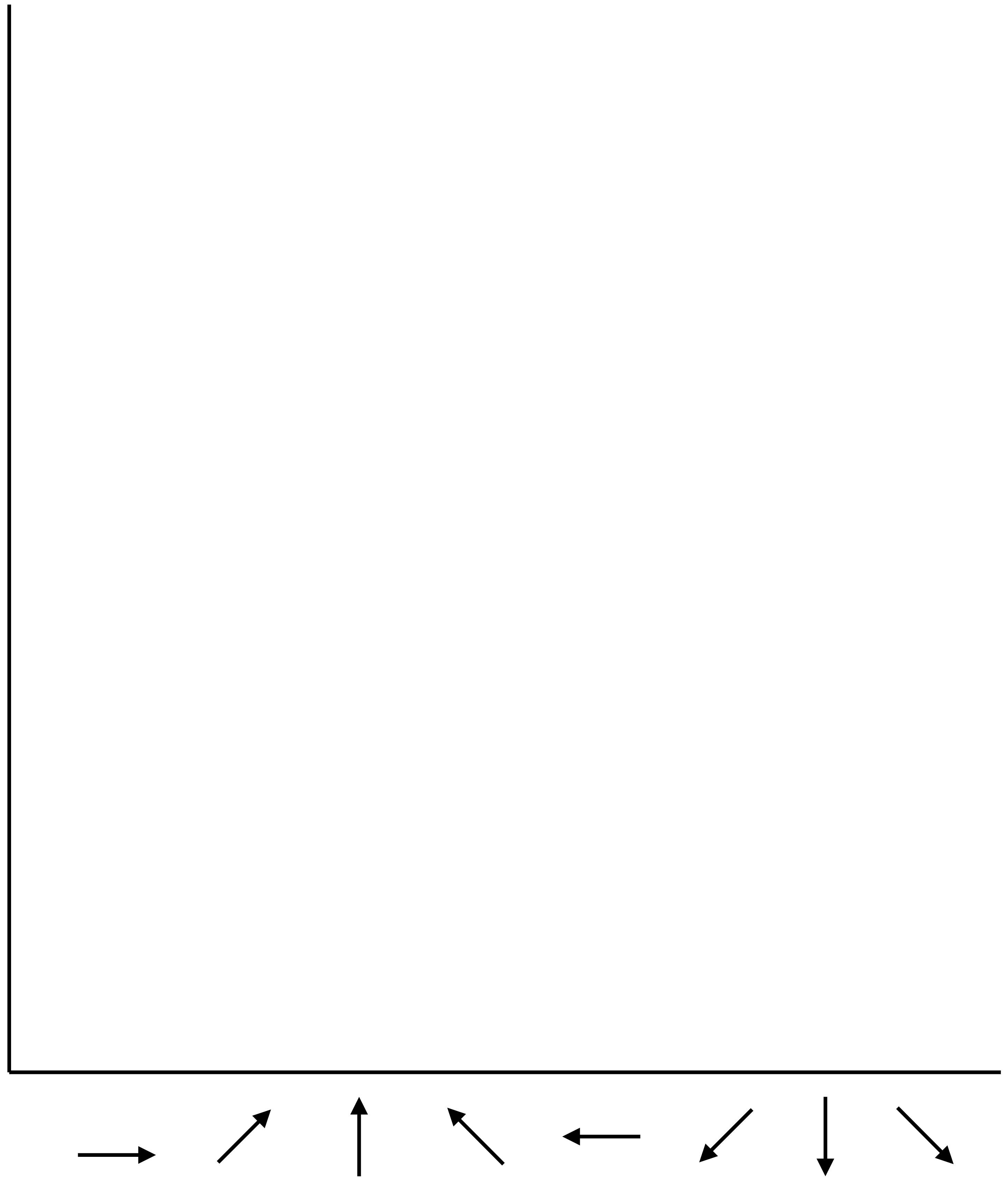
3. Orientation Assignment



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

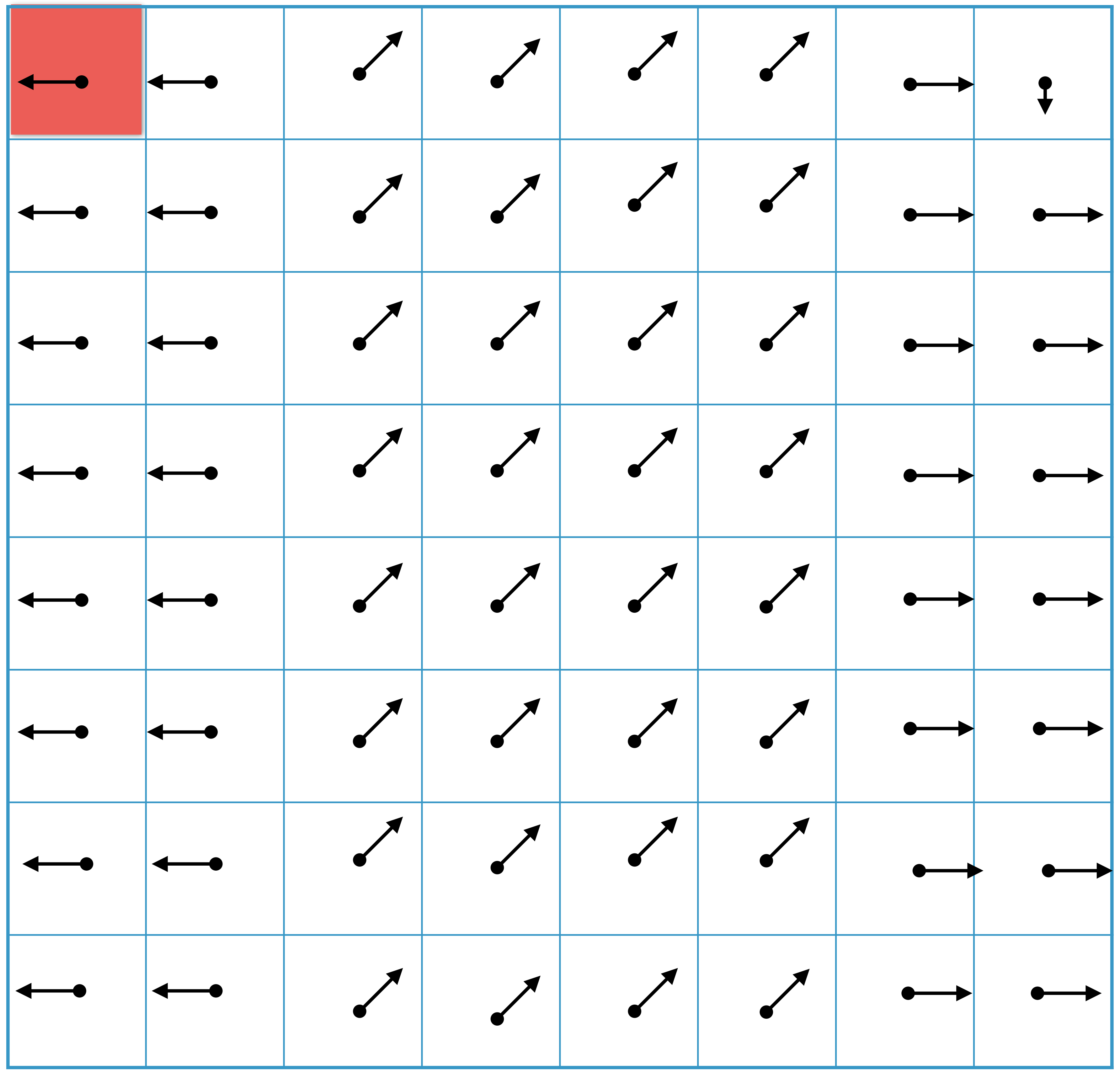
3. Orientation Assignment



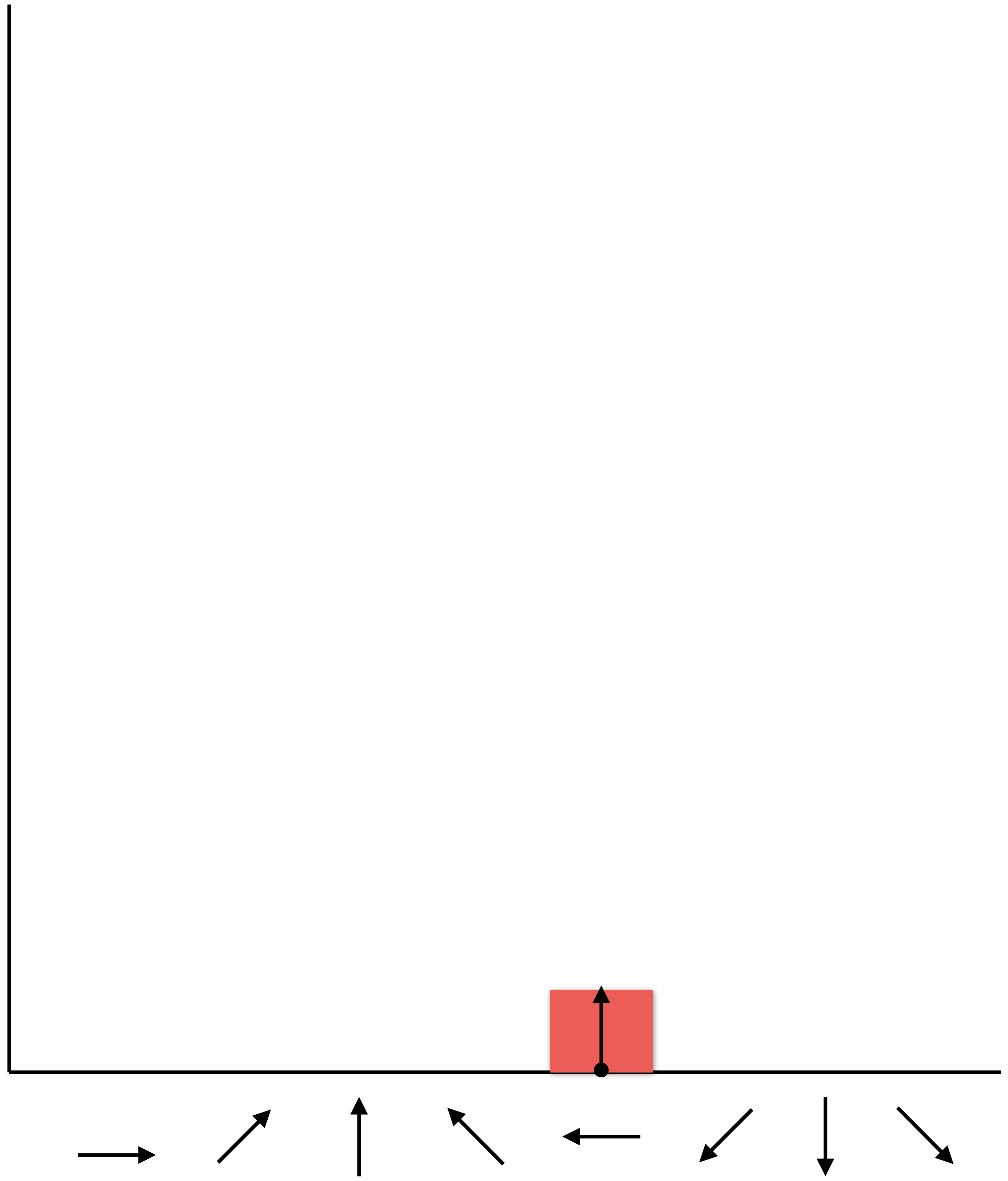


Arrows illustrate **gradient orientation** (direction)
and **gradient magnitude** (arrow length)

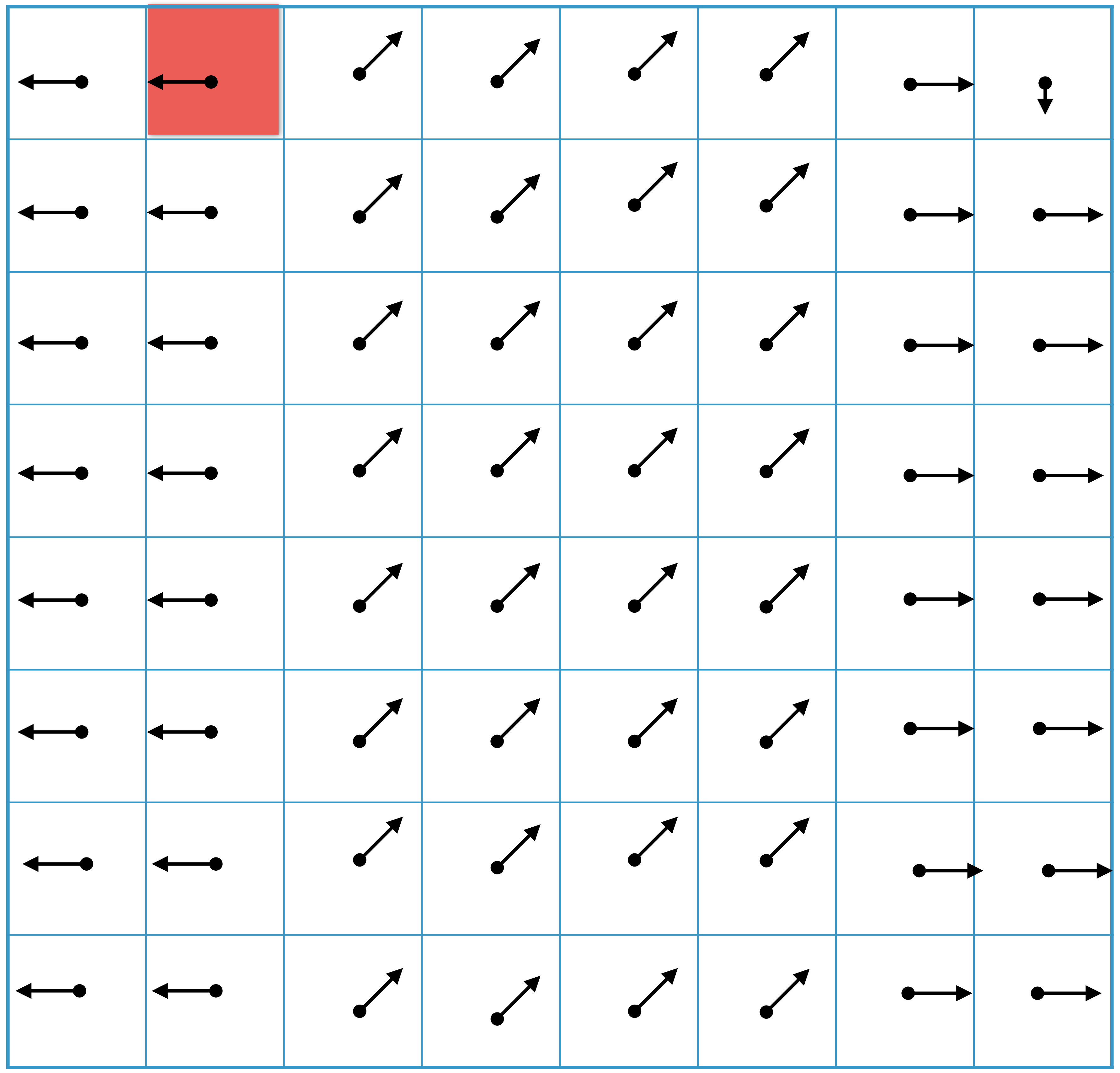
3. Orientation Assignment



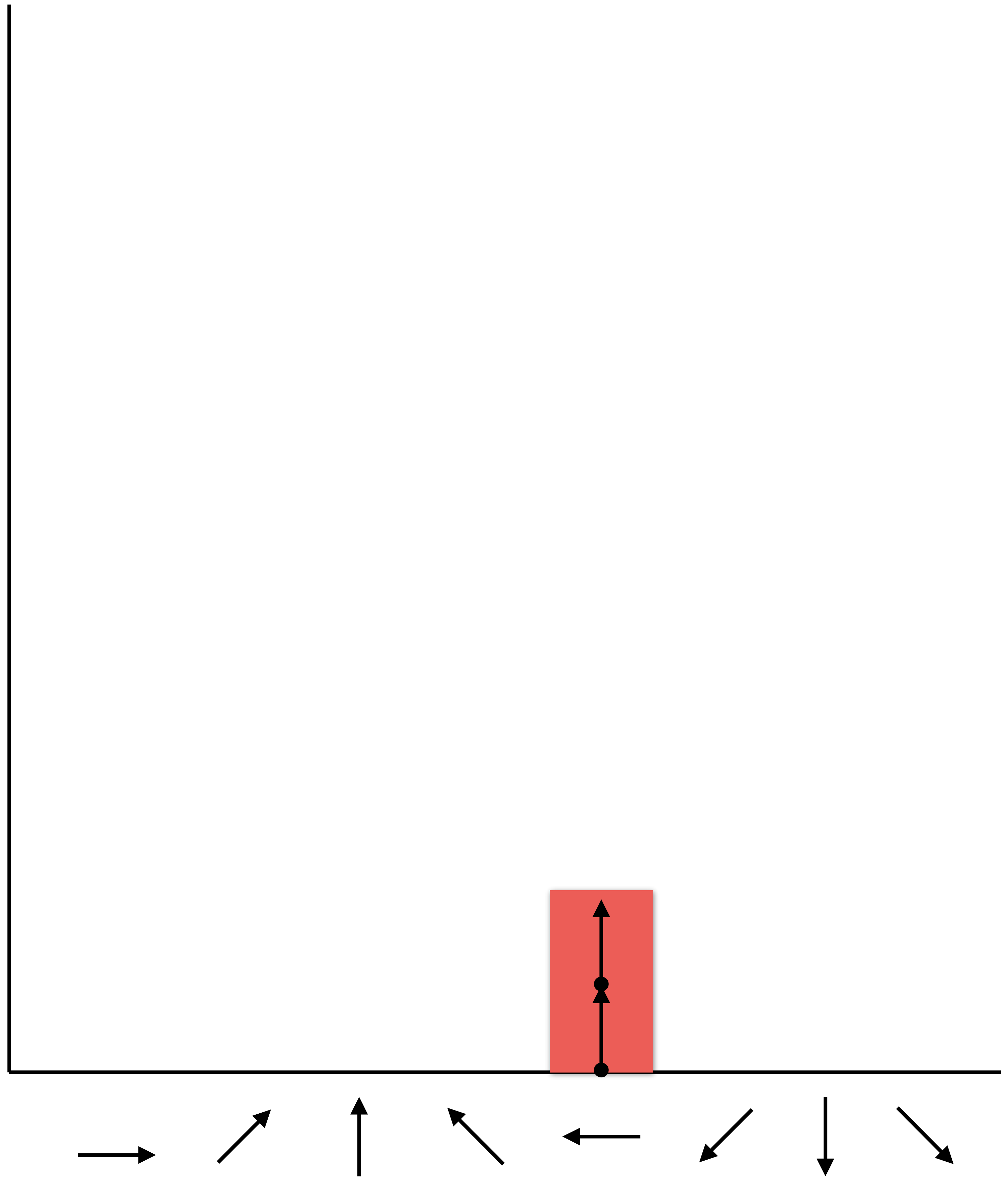
Arrows illustrate **gradient orientation** (direction) and **gradient magnitude** (arrow length)



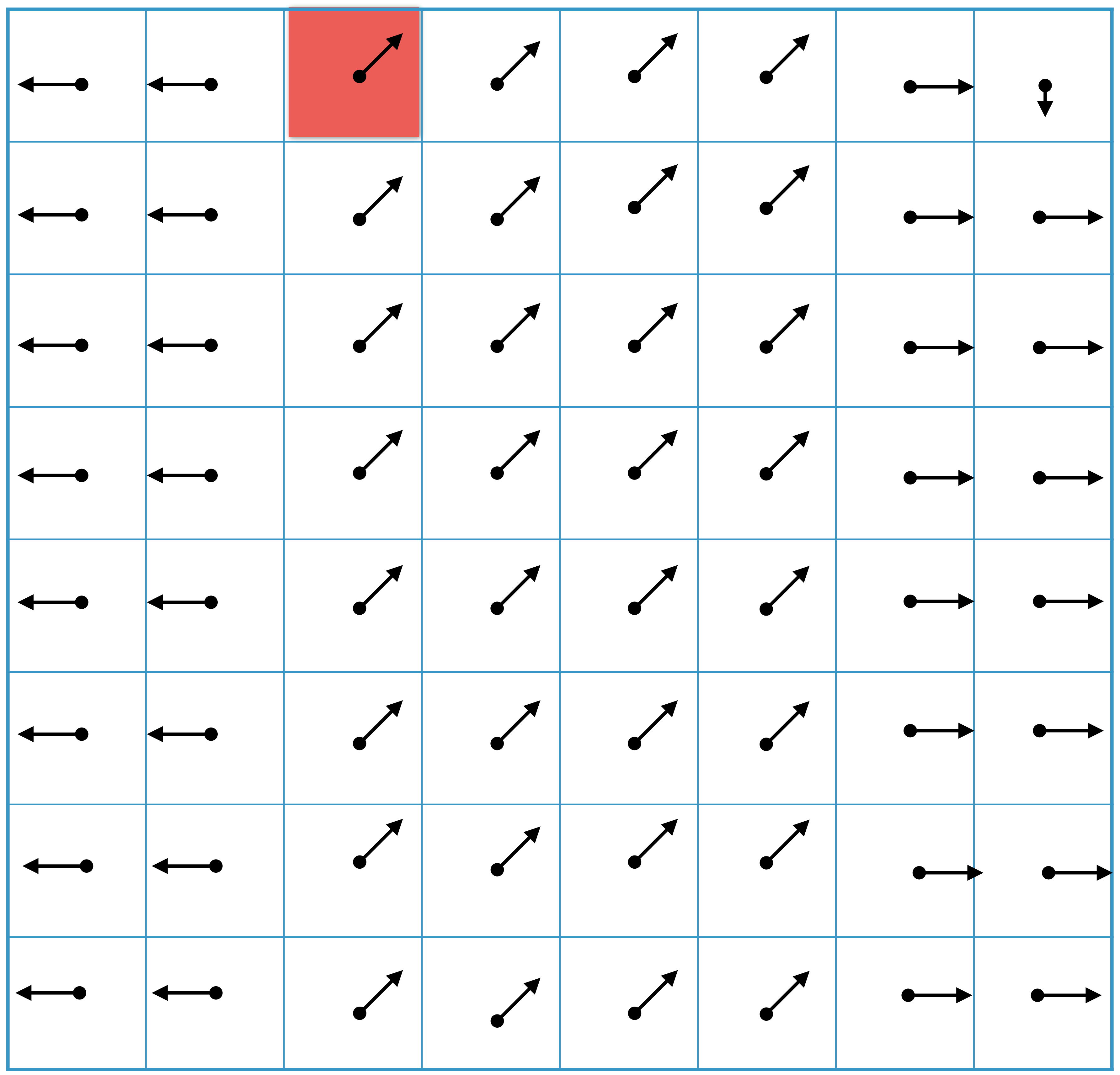
3. Orientation Assignment

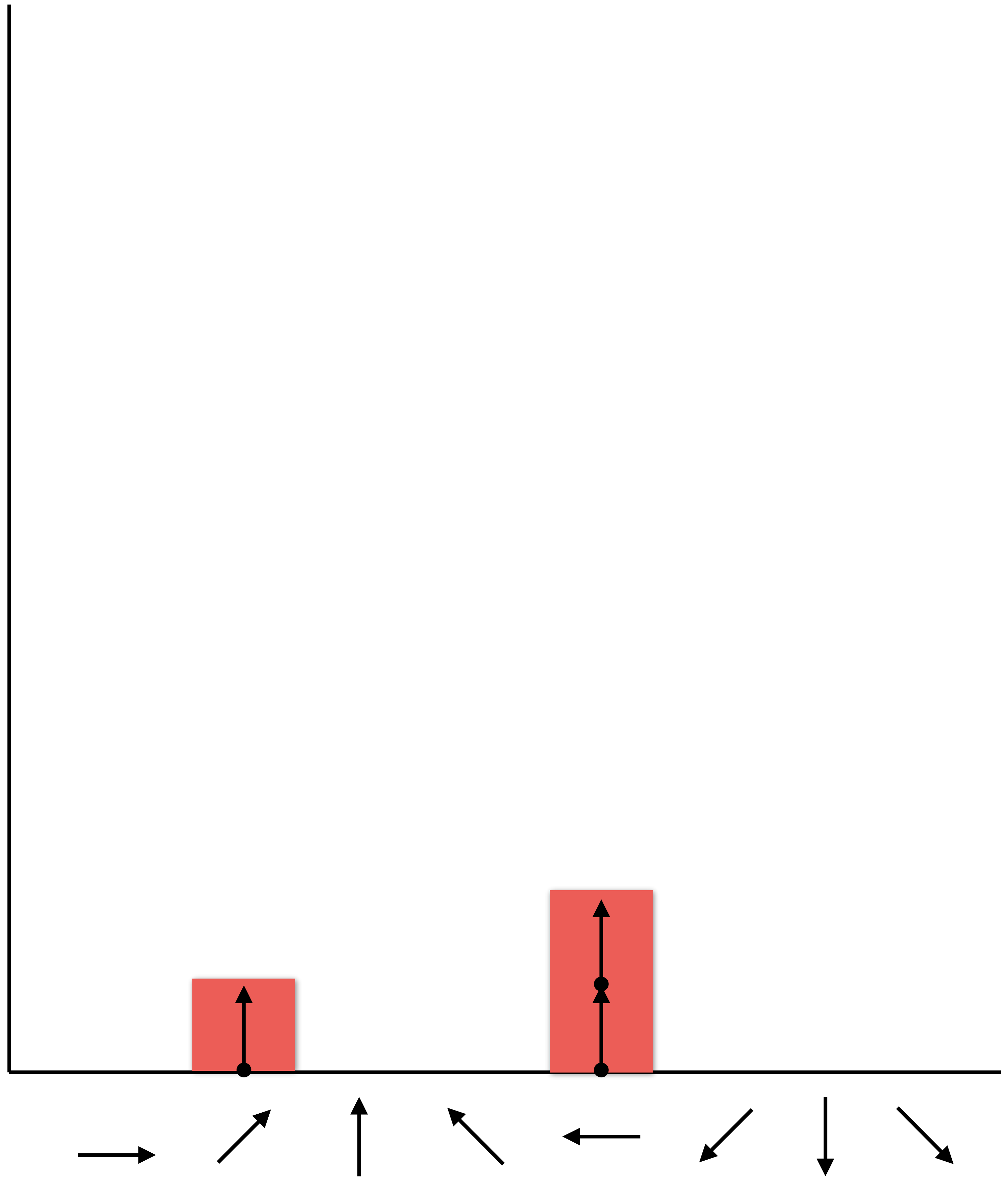


Arrows illustrate **gradient orientation** (direction) and **gradient magnitude** (arrow length)



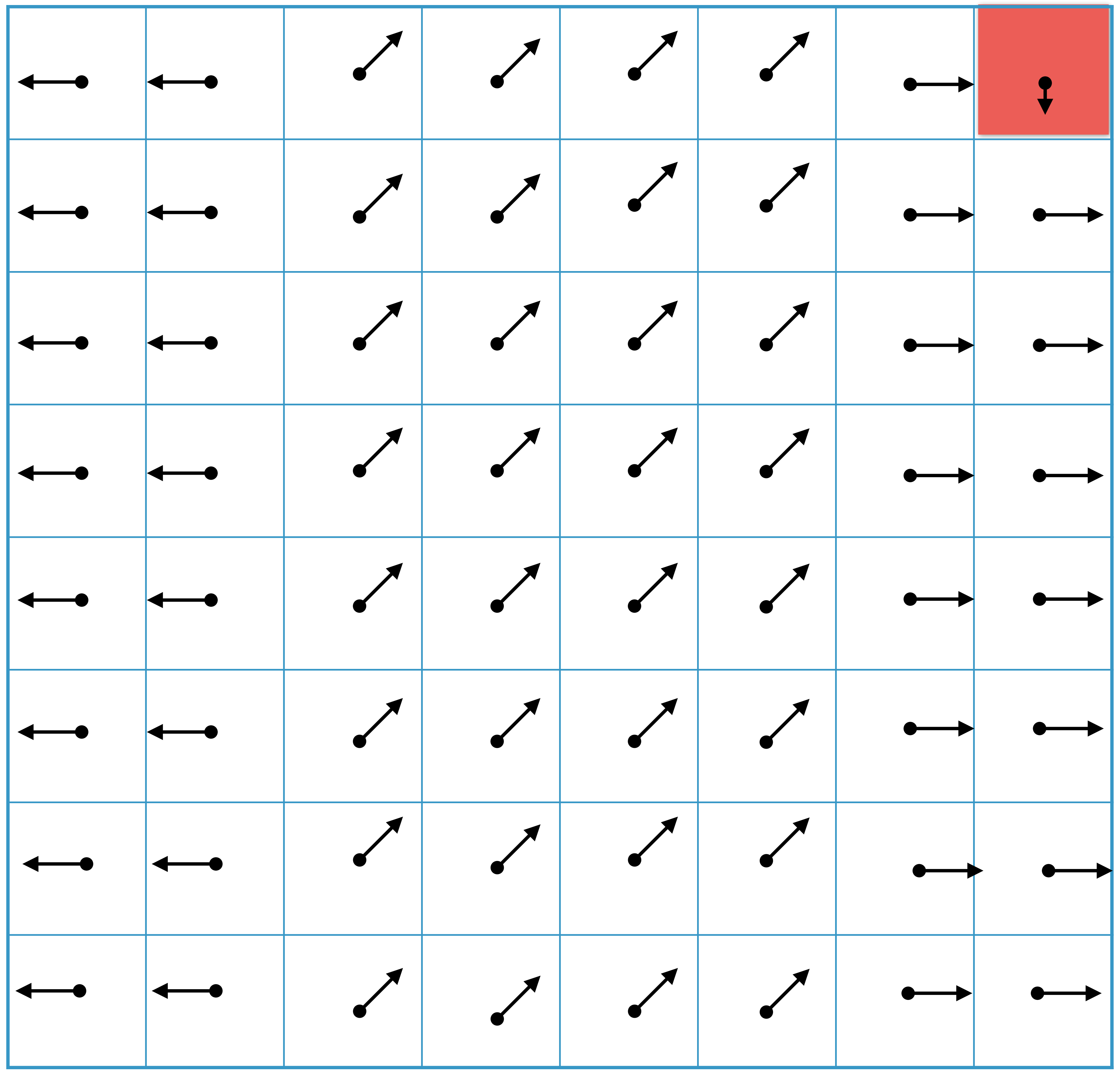
3. Orientation Assignment

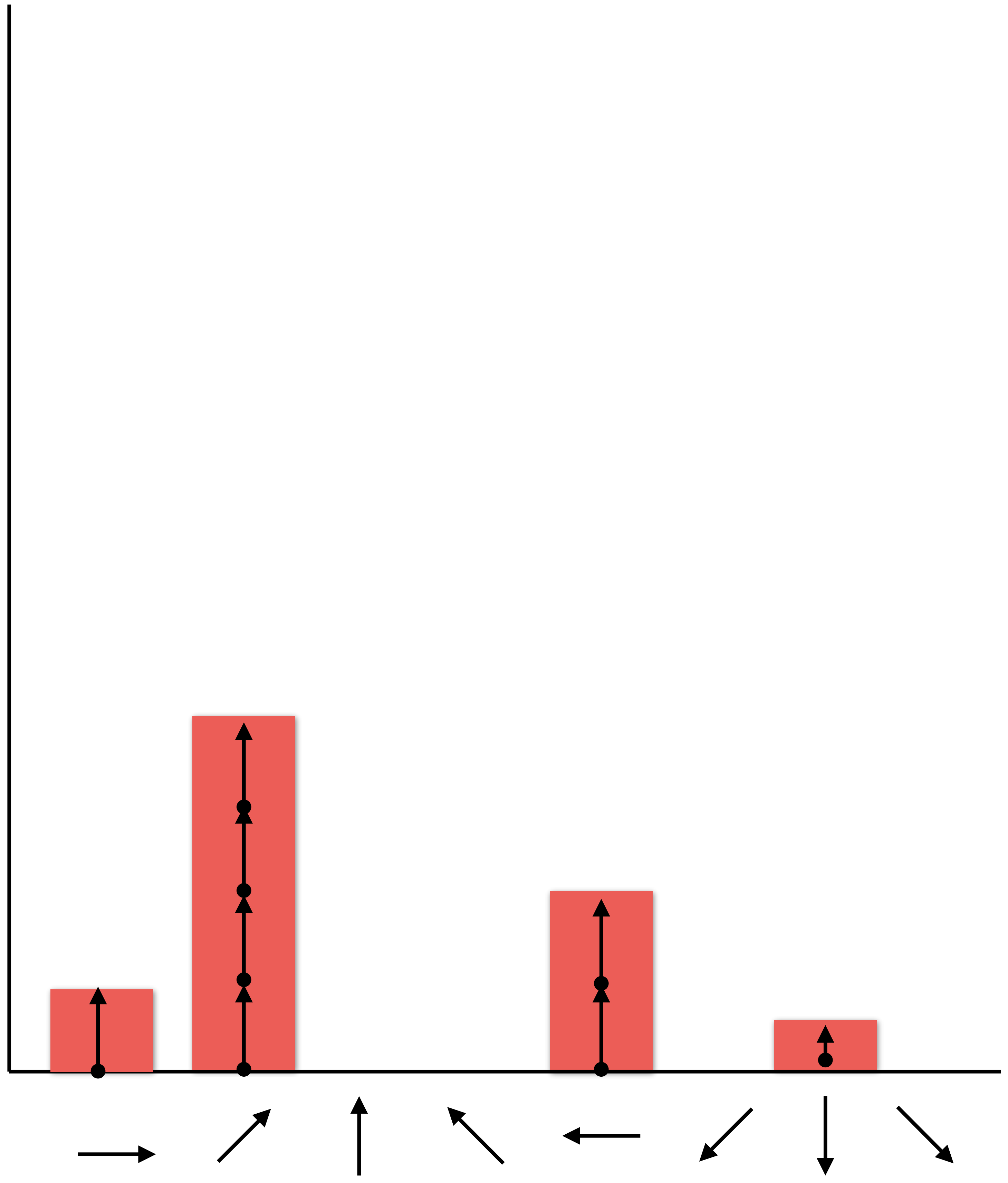




Arrows illustrate **gradient orientation** (direction) and **gradient magnitude** (arrow length)

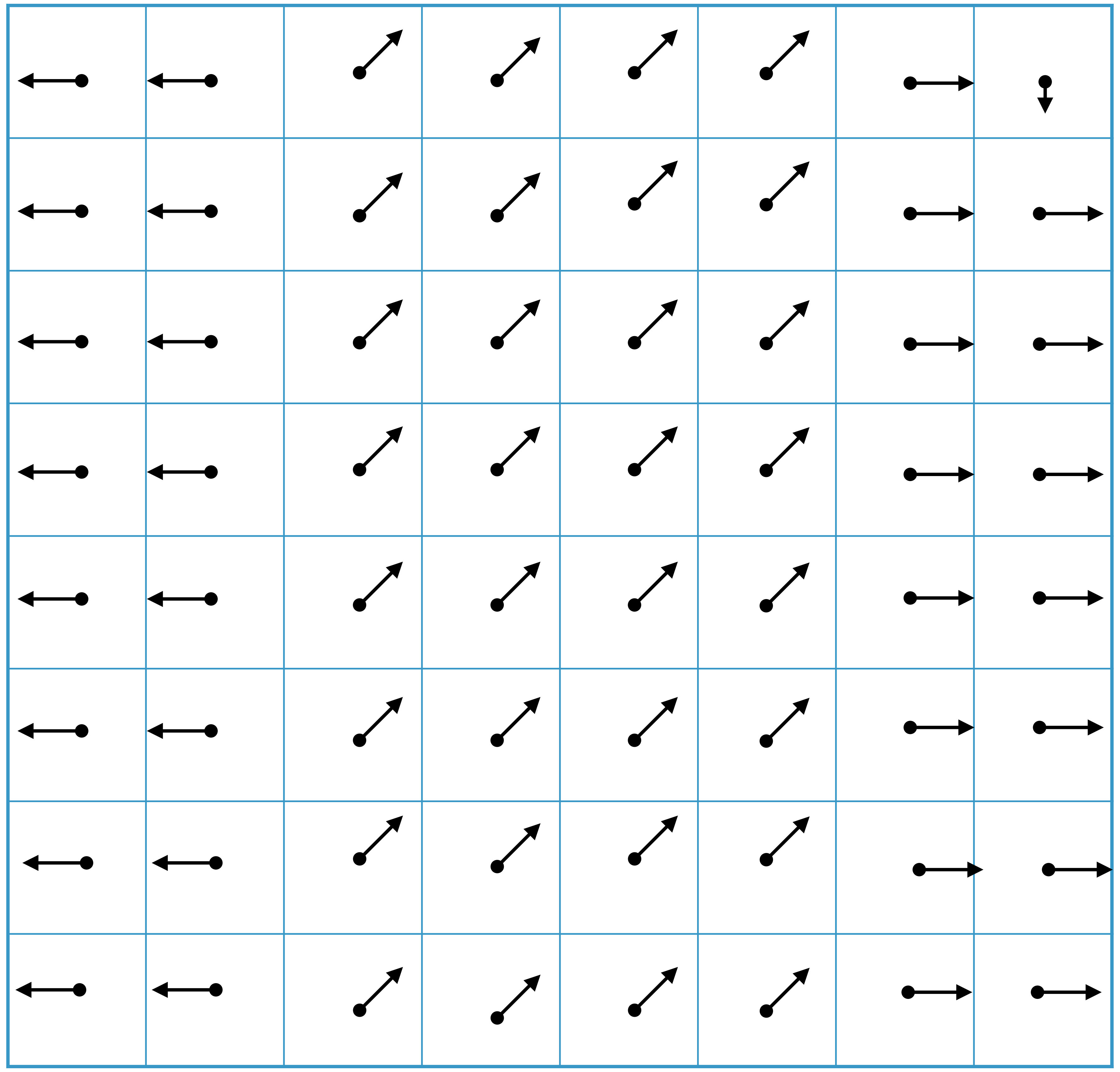
3. Orientation Assignment



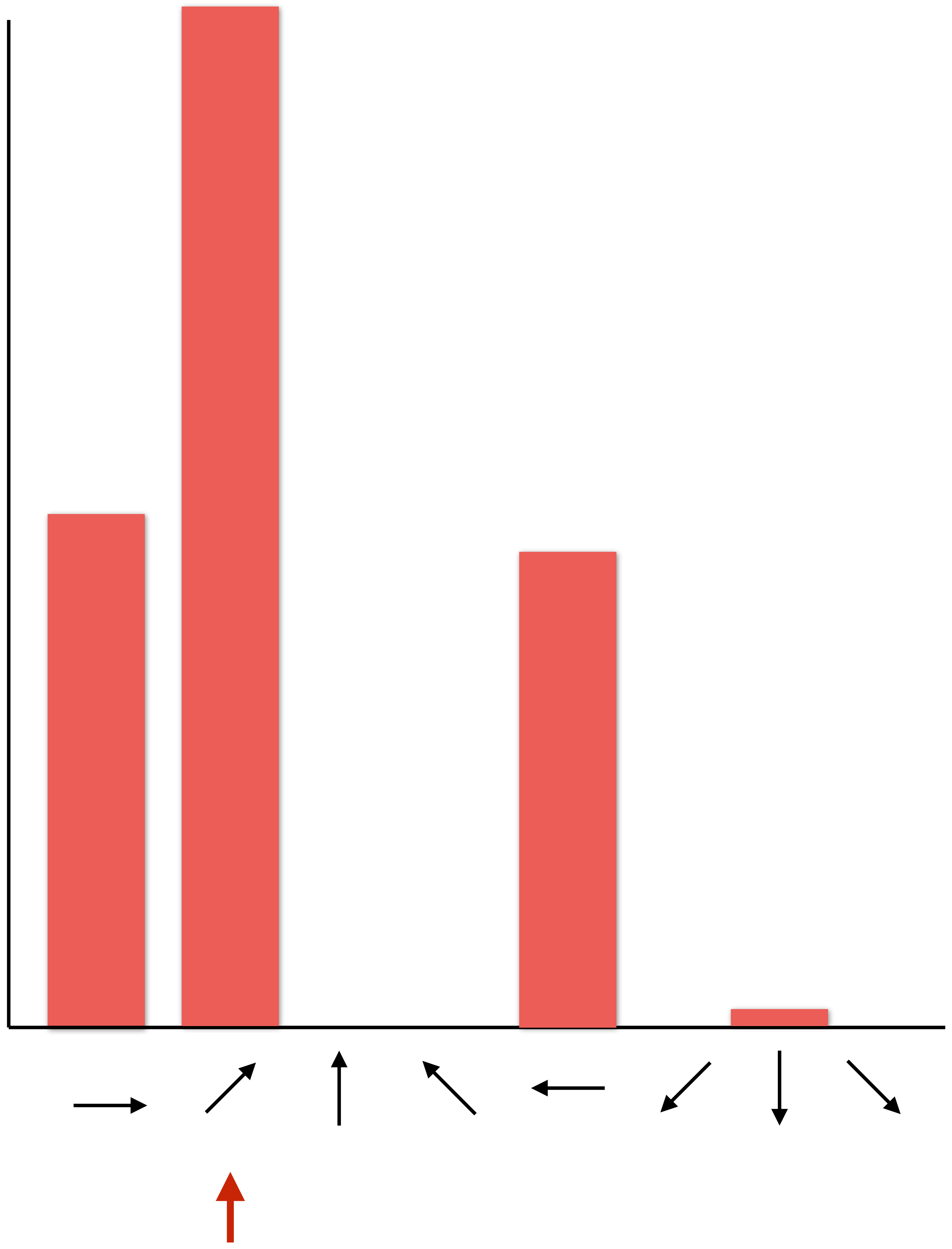


Arrows illustrate **gradient orientation** (direction) and **gradient magnitude** (arrow length)

3. Orientation Assignment

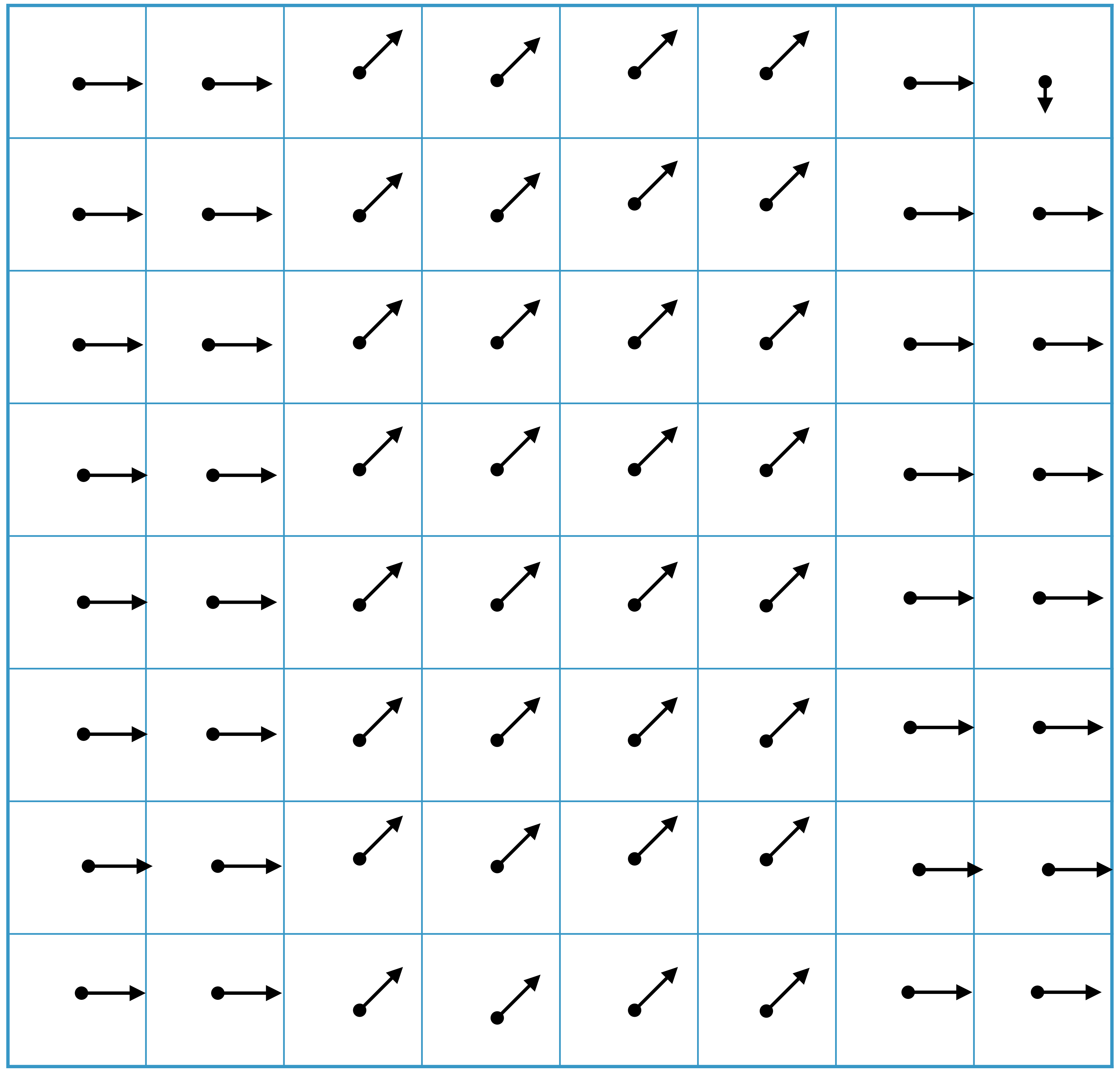


Arrows illustrate **gradient orientation** (direction) and **gradient magnitude** (arrow length)

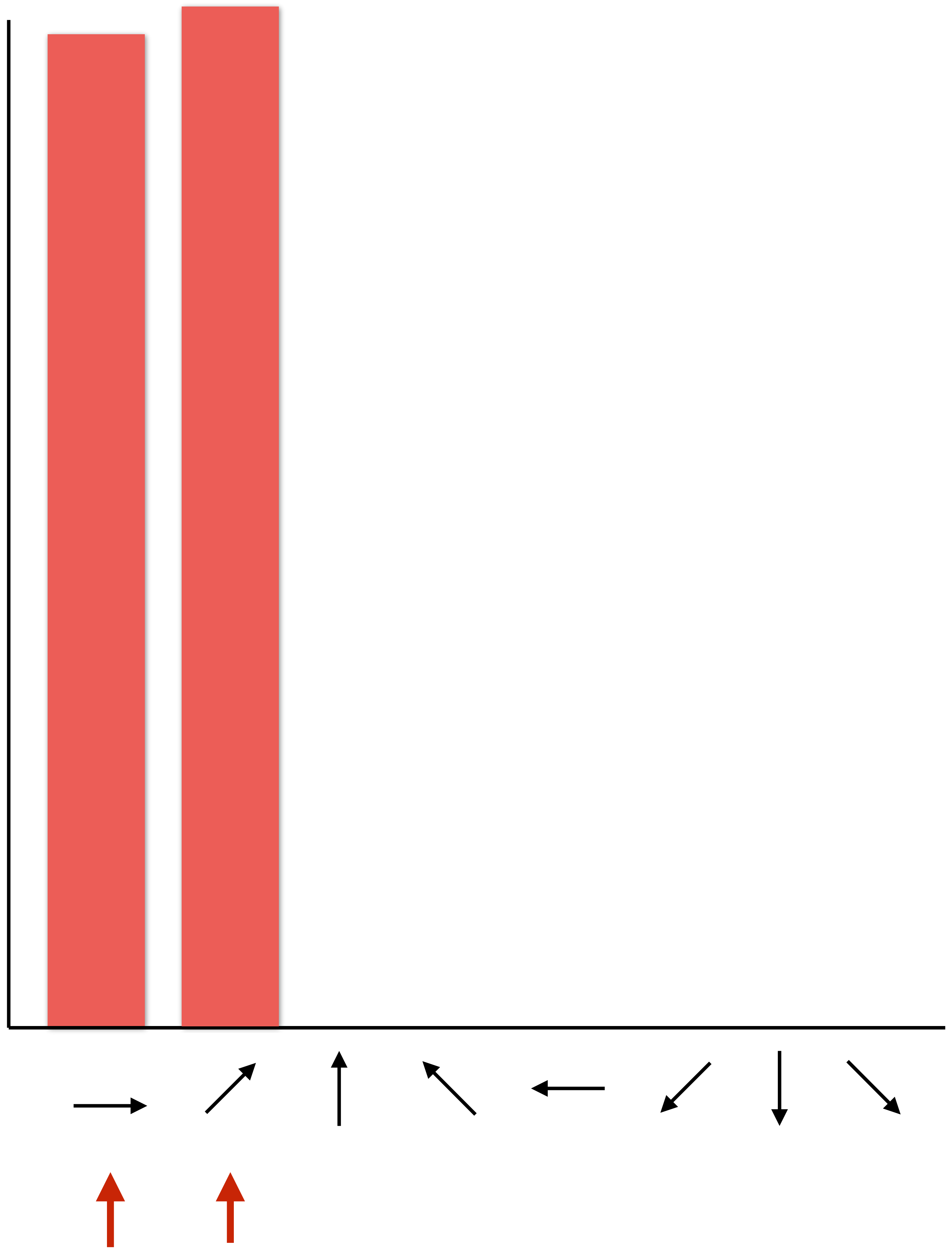


Assigned Orientation

3. Orientation Assignment



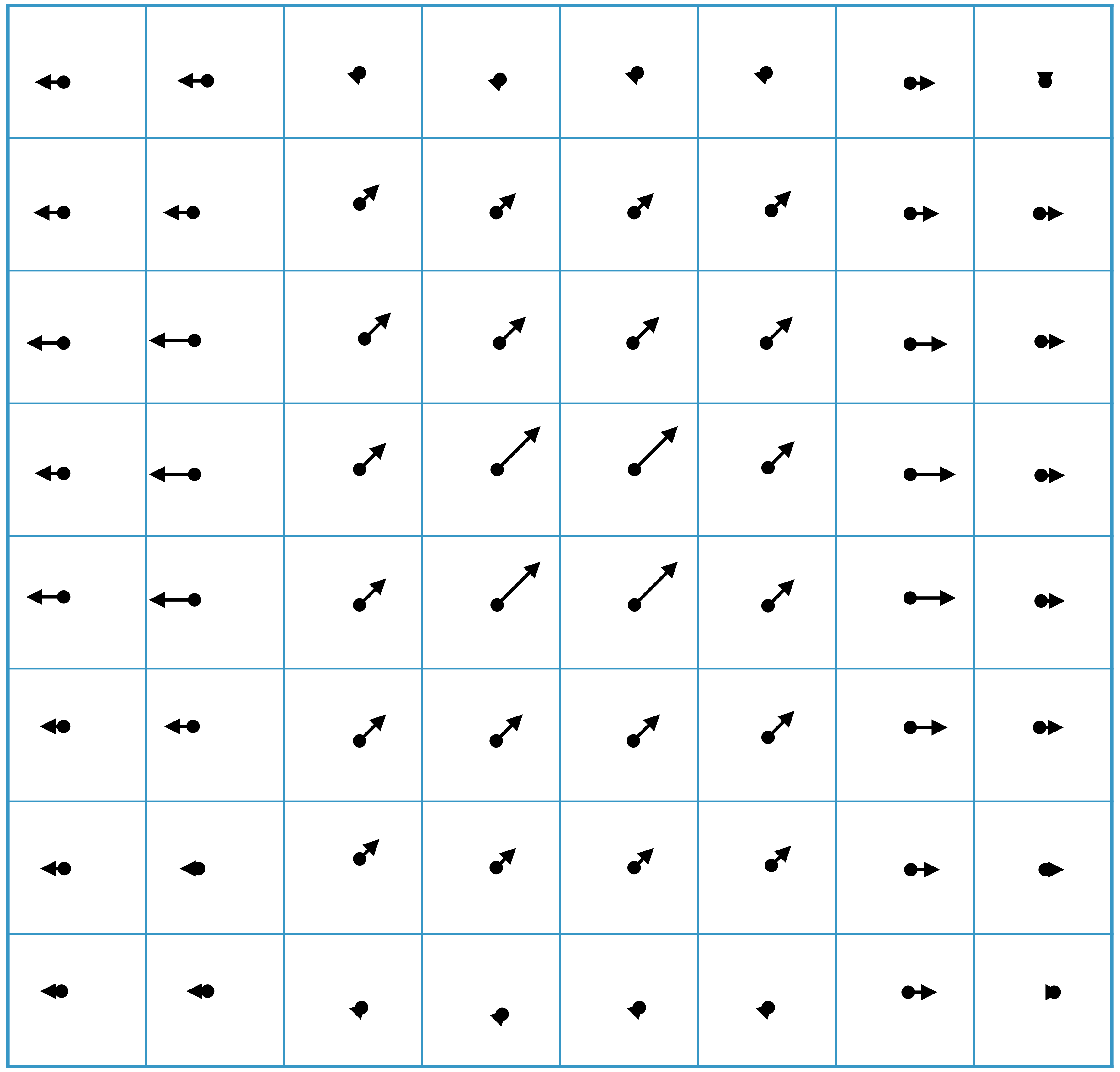
Arrows illustrate **gradient orientation** (direction) and **gradient magnitude** (arrow length)



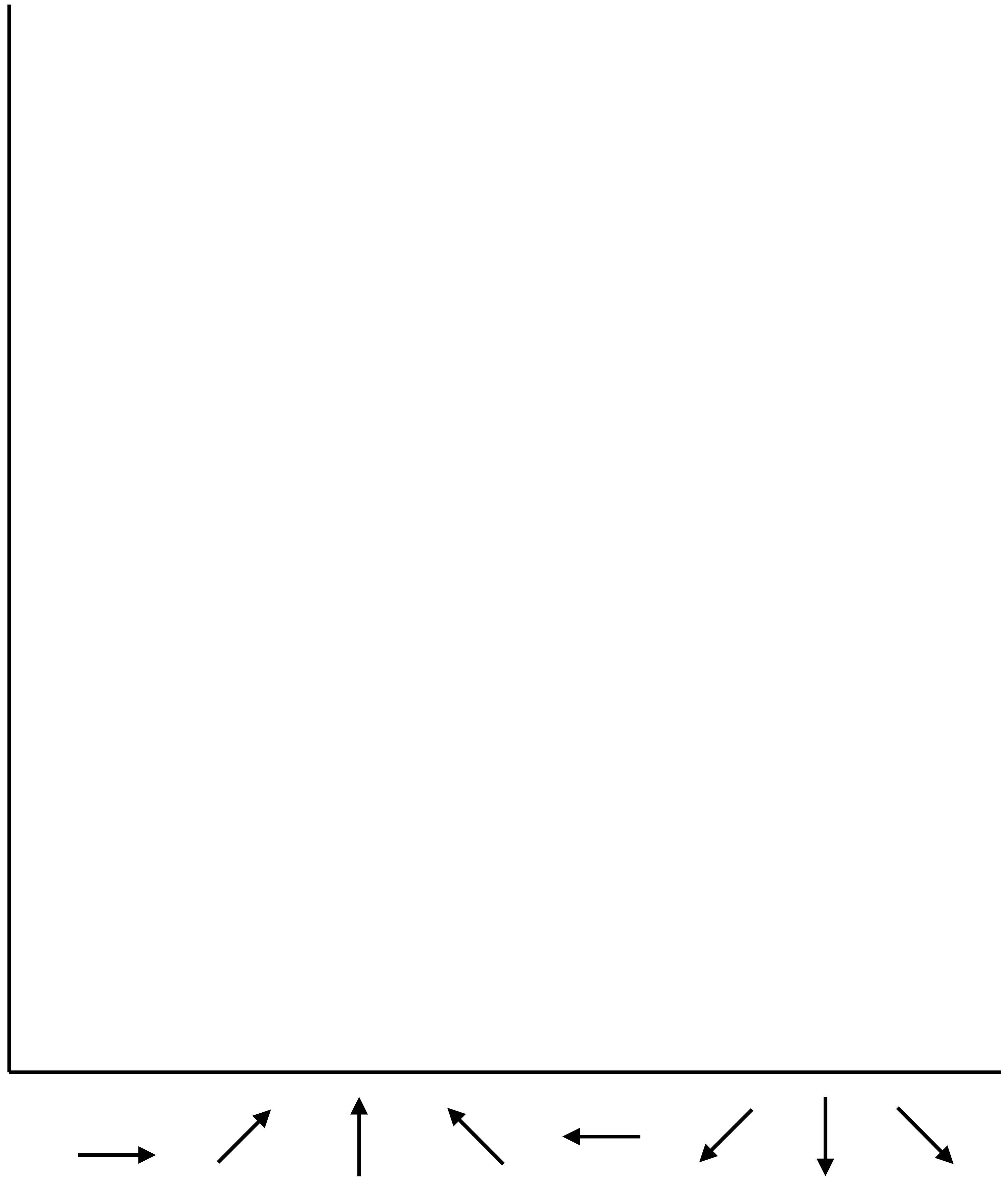
Assigned Orientation

3. Orientation Assignment

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



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



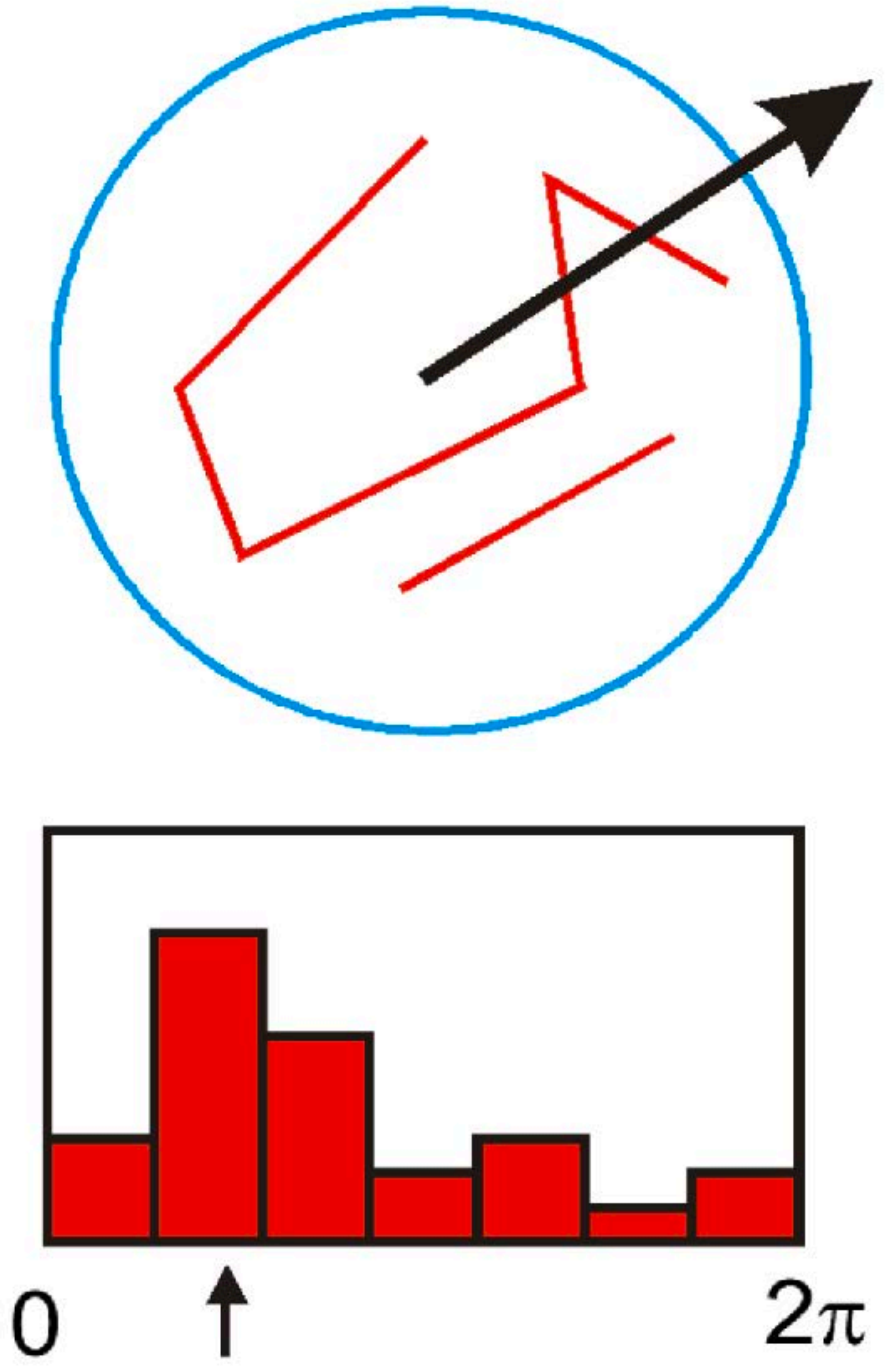
3. Orientation Assignment

— **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:

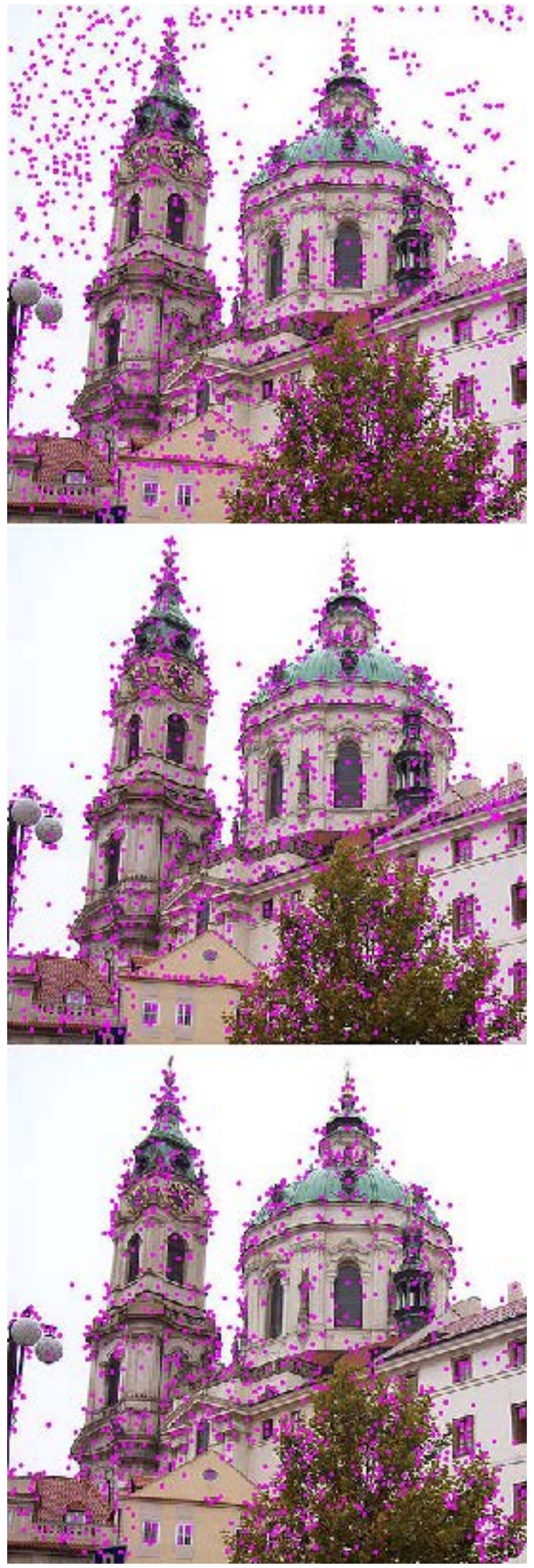
(a) 233×189 image

(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

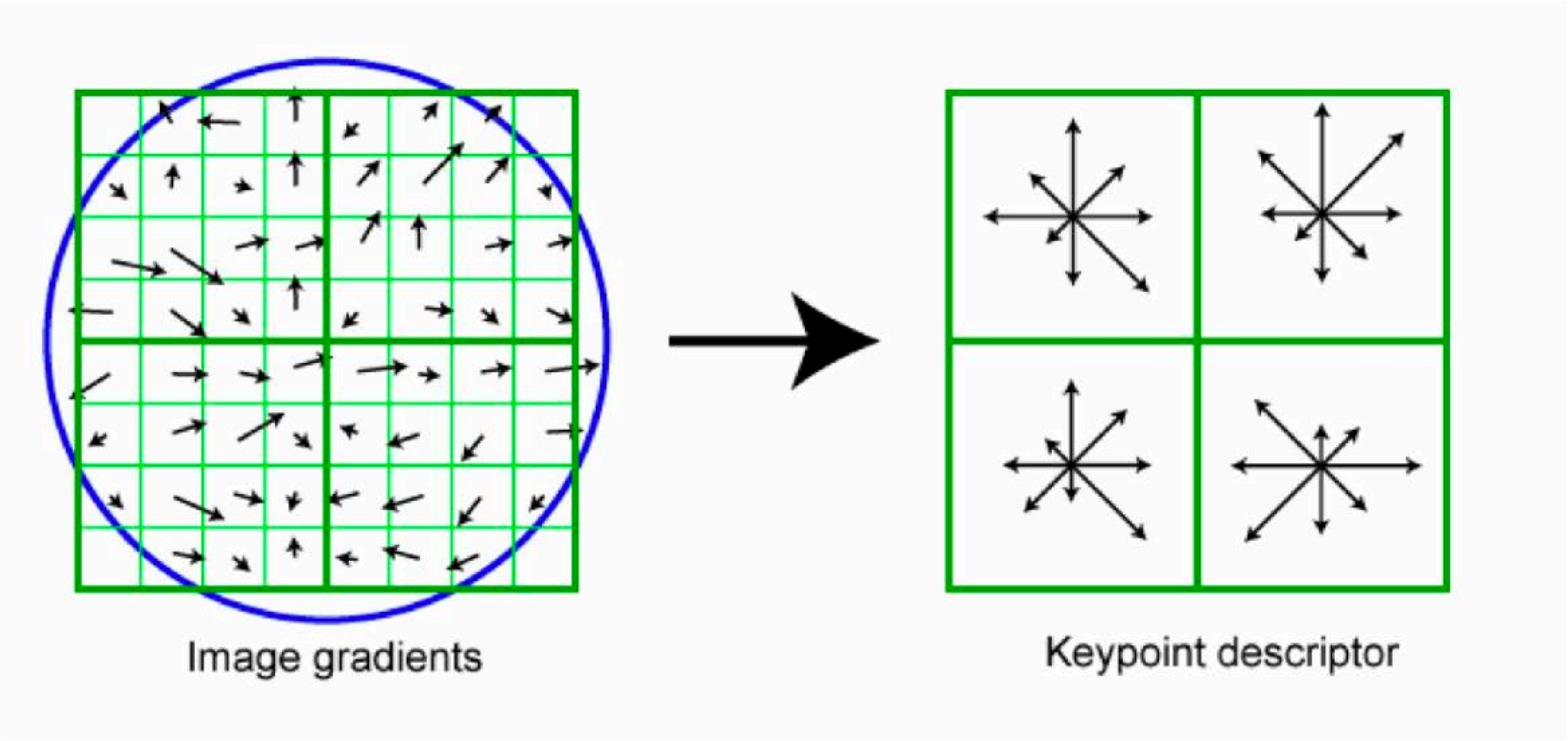
— Keypoint detection is not the same as keypoint description, e.g. some applications skip keypoint detection and extract SIFT descriptors on a regularly spaced grid

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

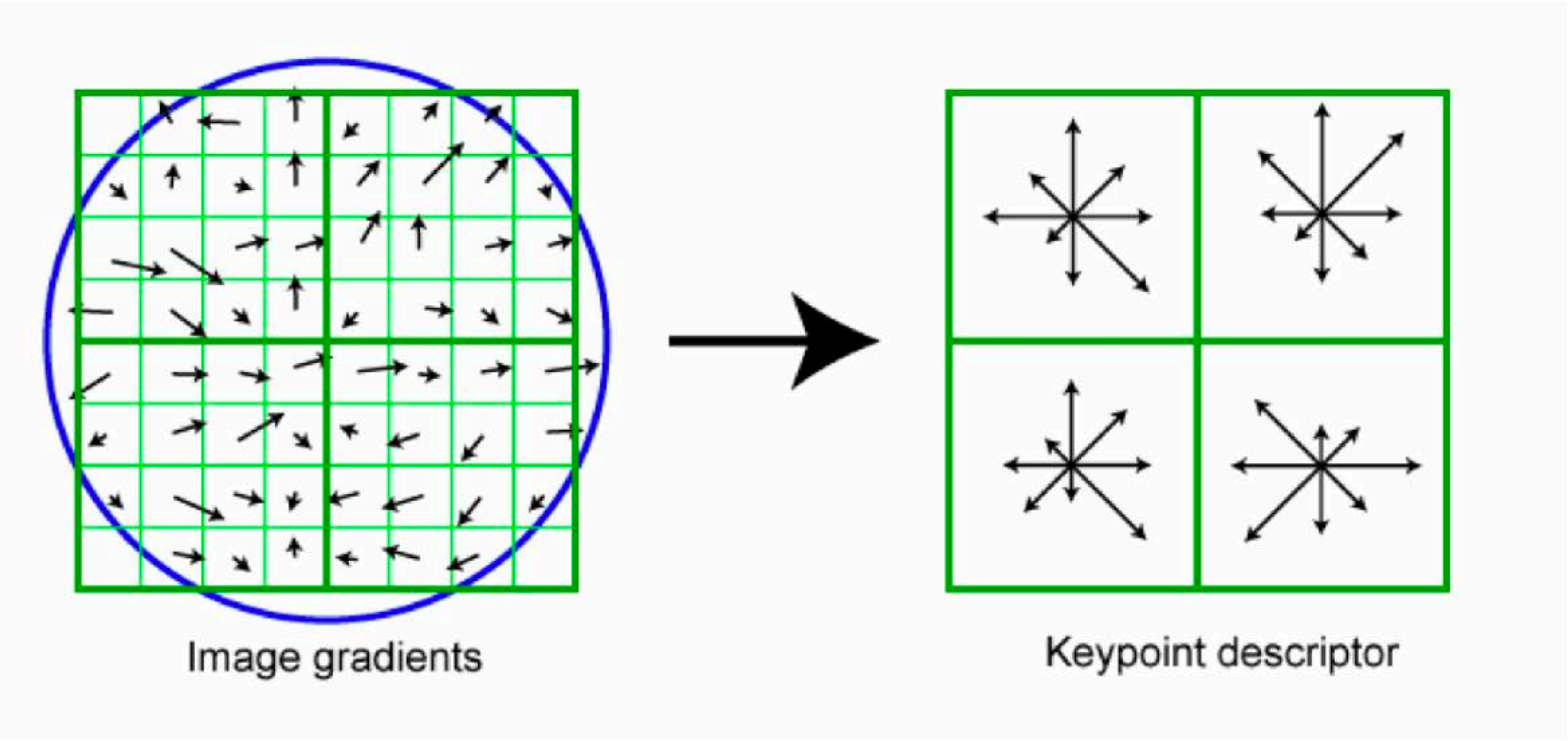
— 8 orientations × 4 × 4 histogram array



4. SIFT Descriptor

How many dimensions are there in a SIFT descriptor?

(**Note**: 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

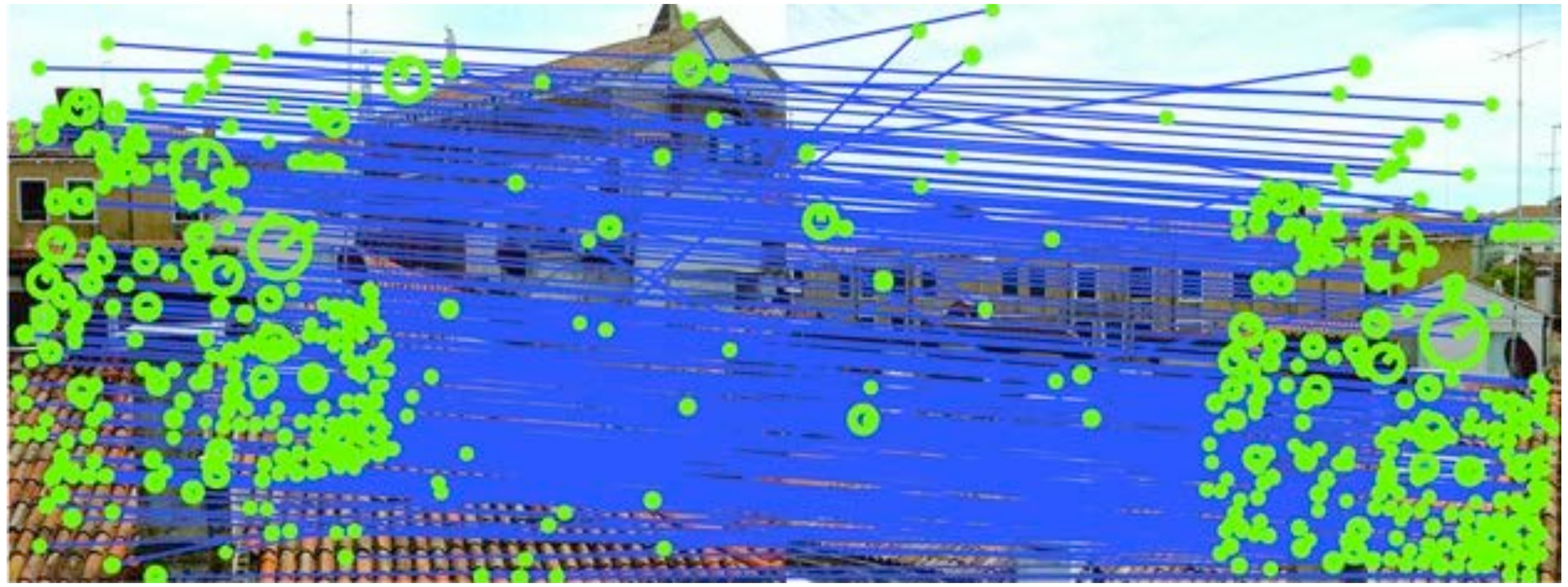
Extract features from the image …

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:

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

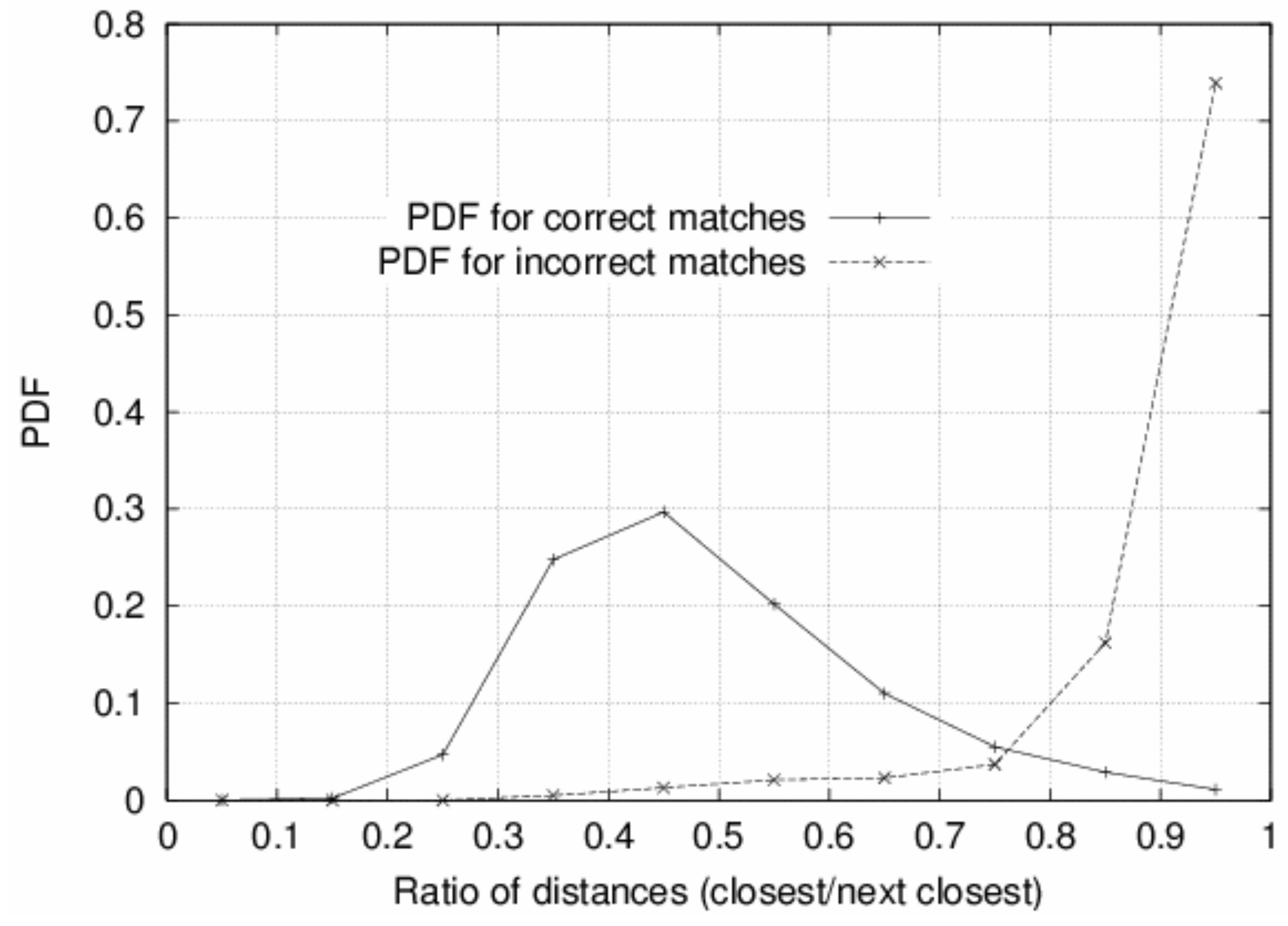
— This is expensive (linear time), but good approximation algorithms exist

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

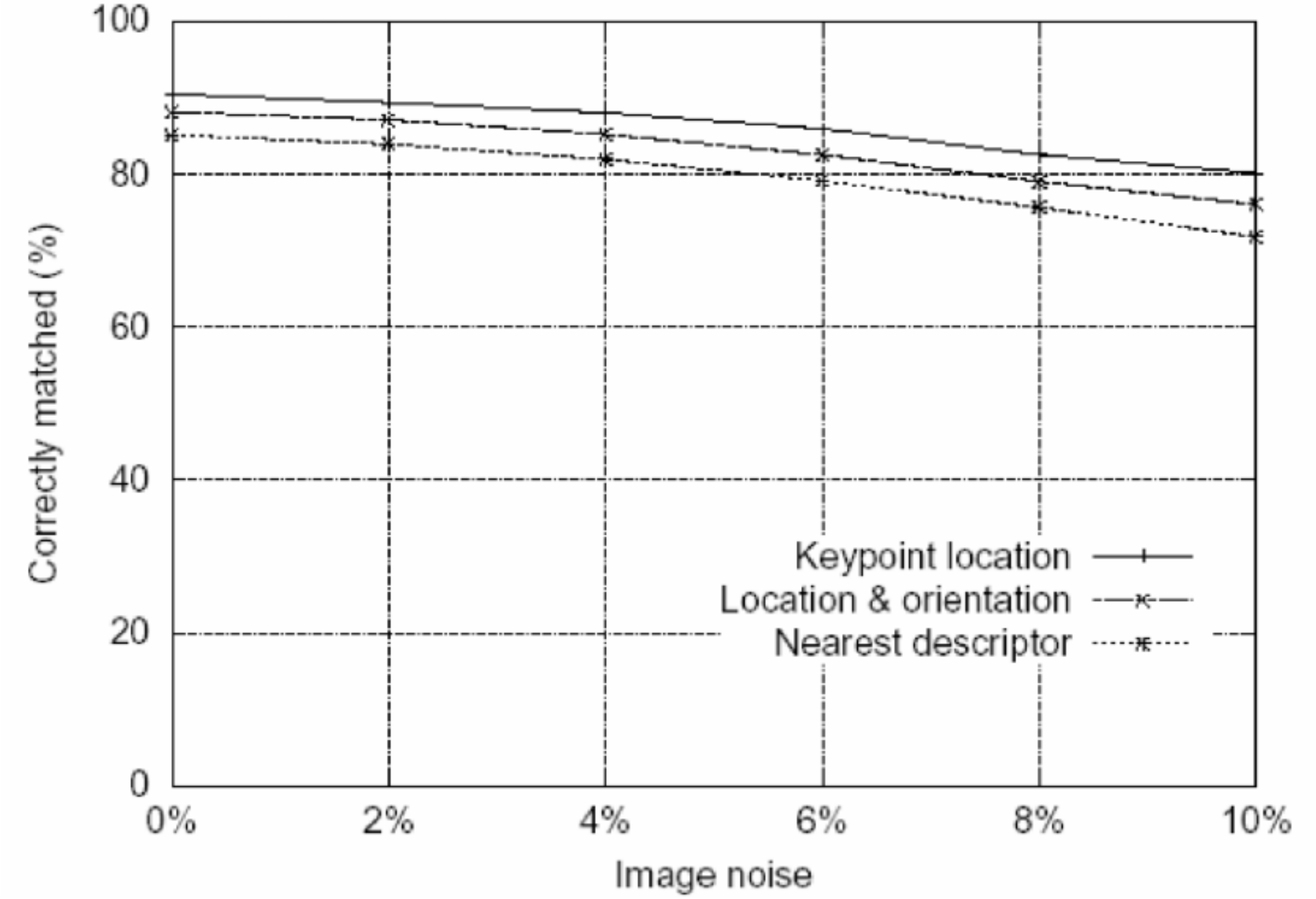


Any other ways to filter out matches?

Feature Stability to **Noise**

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

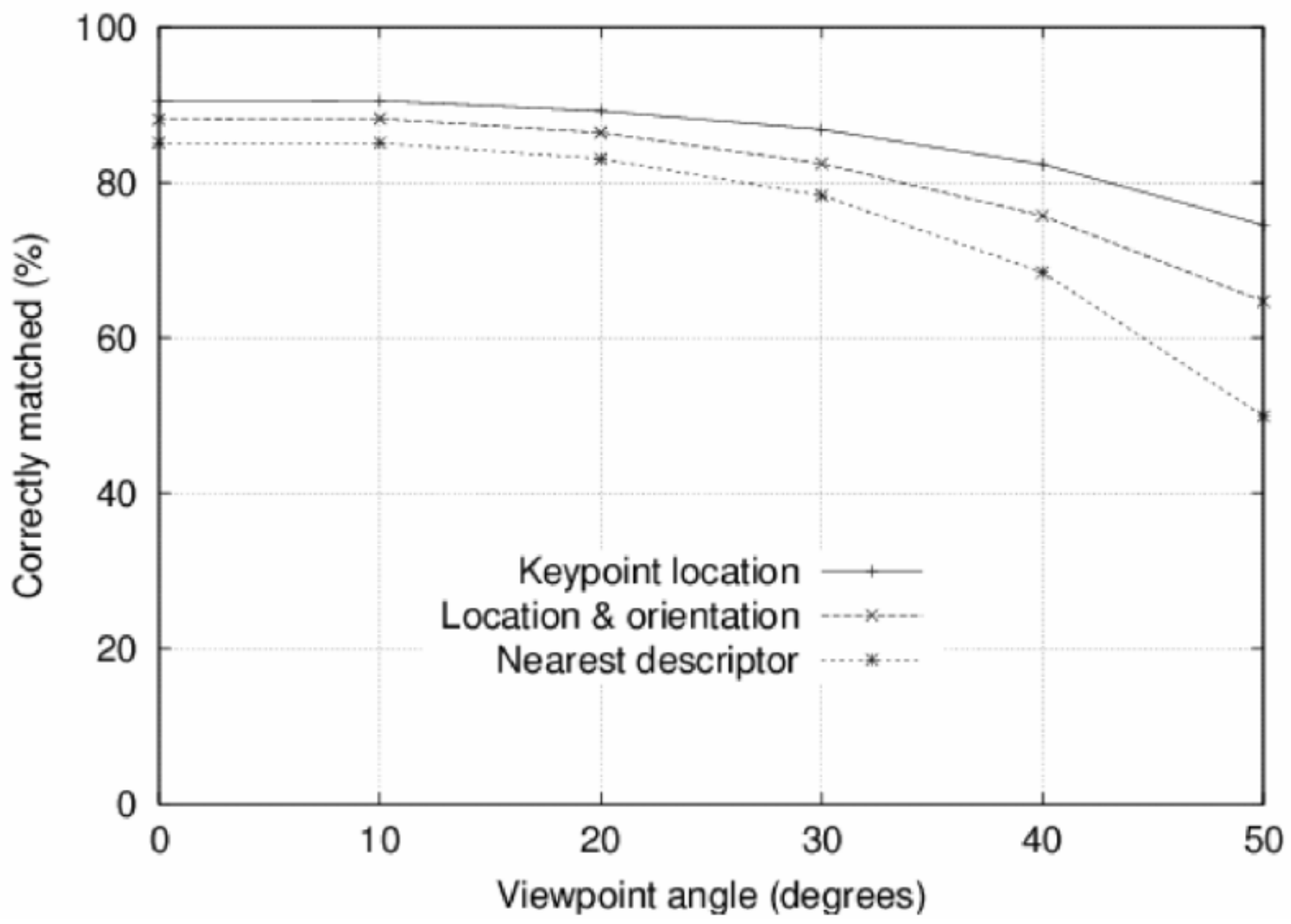
Find nearest neighbour in database of 30,000 features



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)

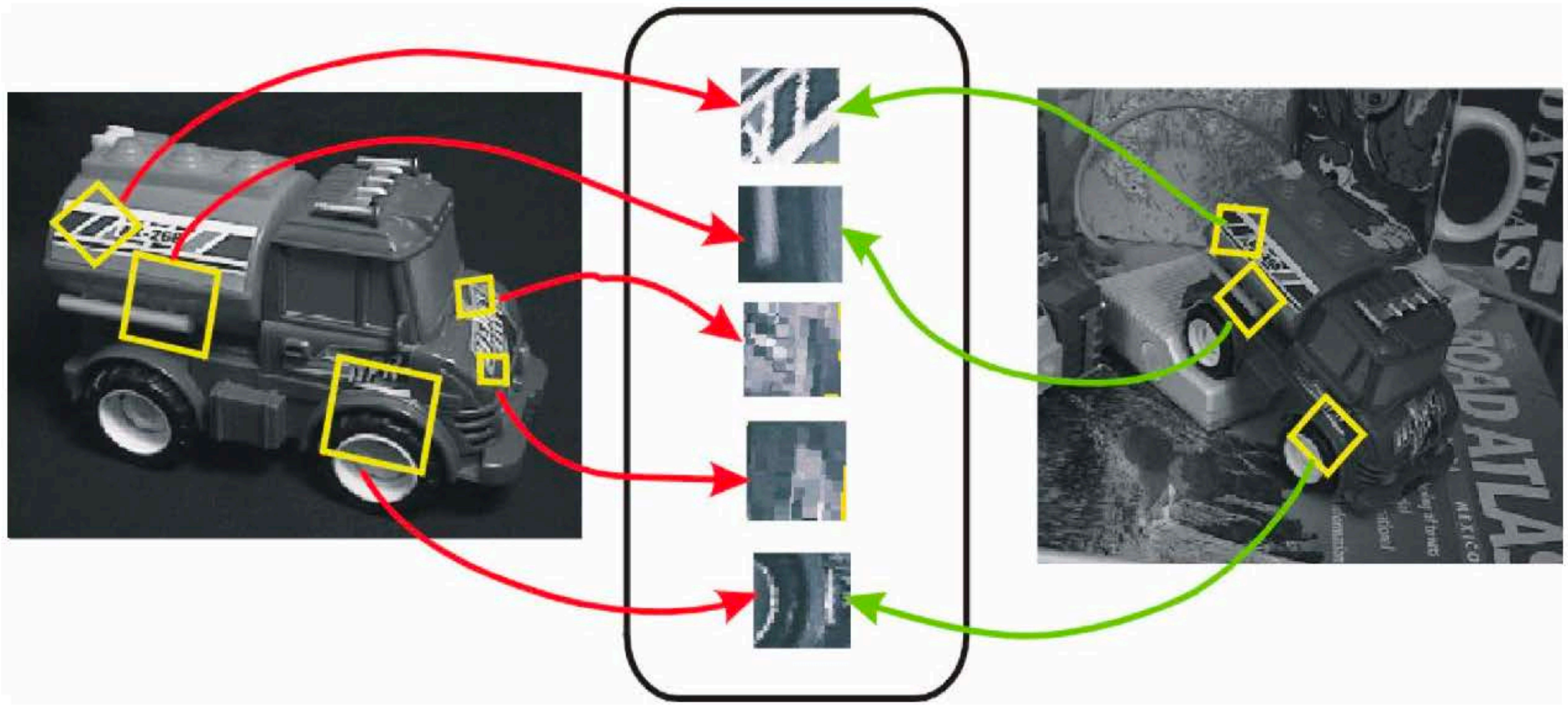
STOPPED HERE FOR 2025 WT1 OFFERING

Review: Learning **Goals**

1. The design philosophy behind SIFT

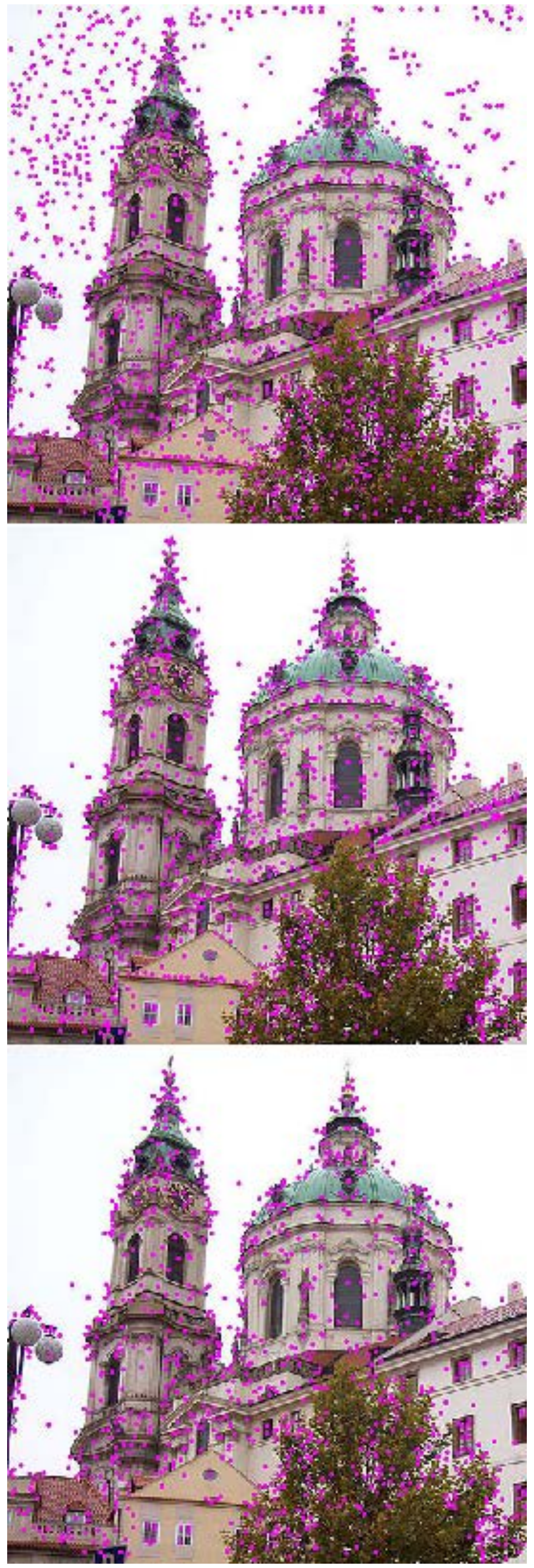
David Lowe's Invariant Local Features

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



SIFT Features

Scale Invariant Feature Transform (**SIFT**)

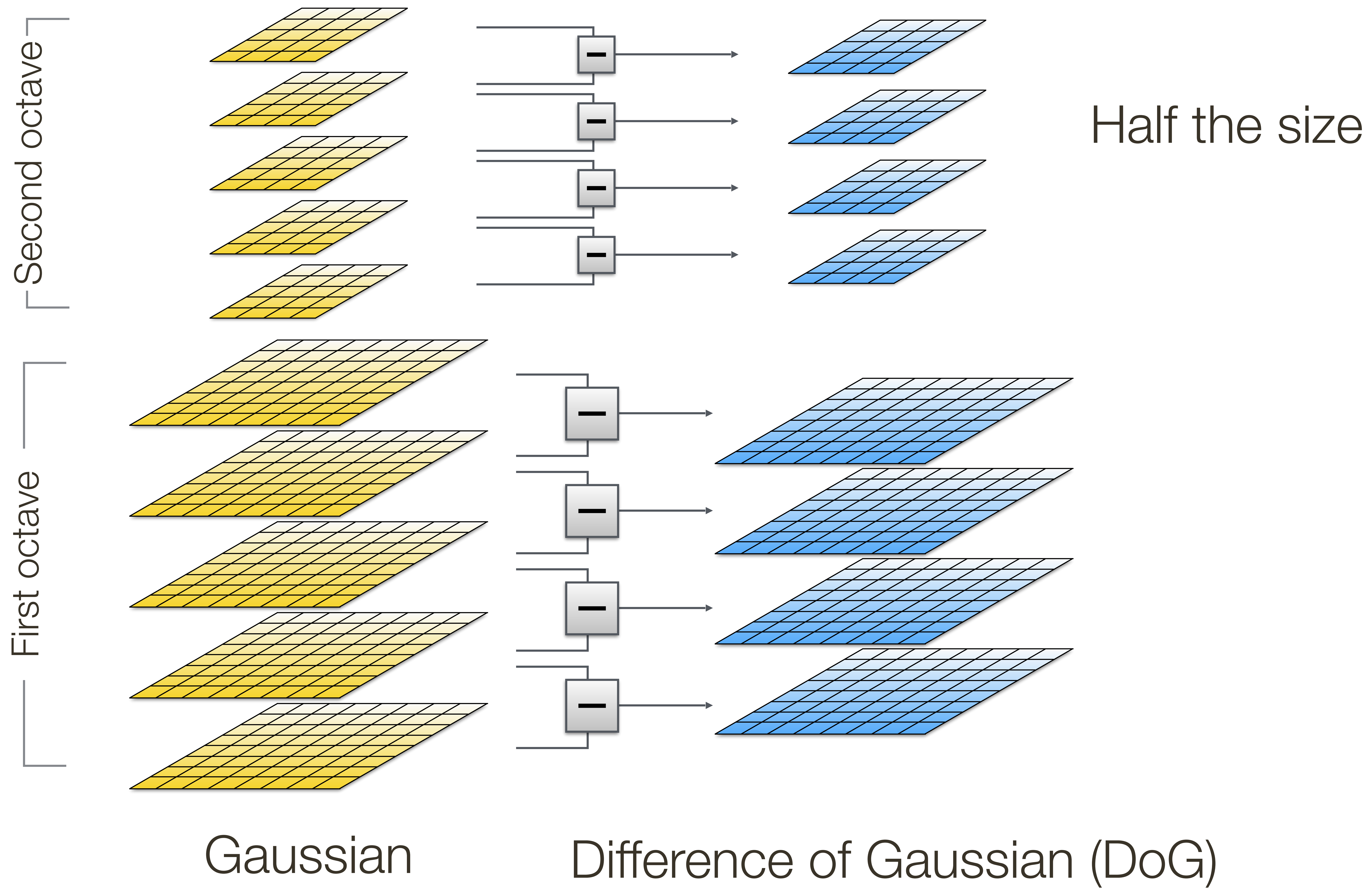


SIFT describes both a **detector** and **descriptor**

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

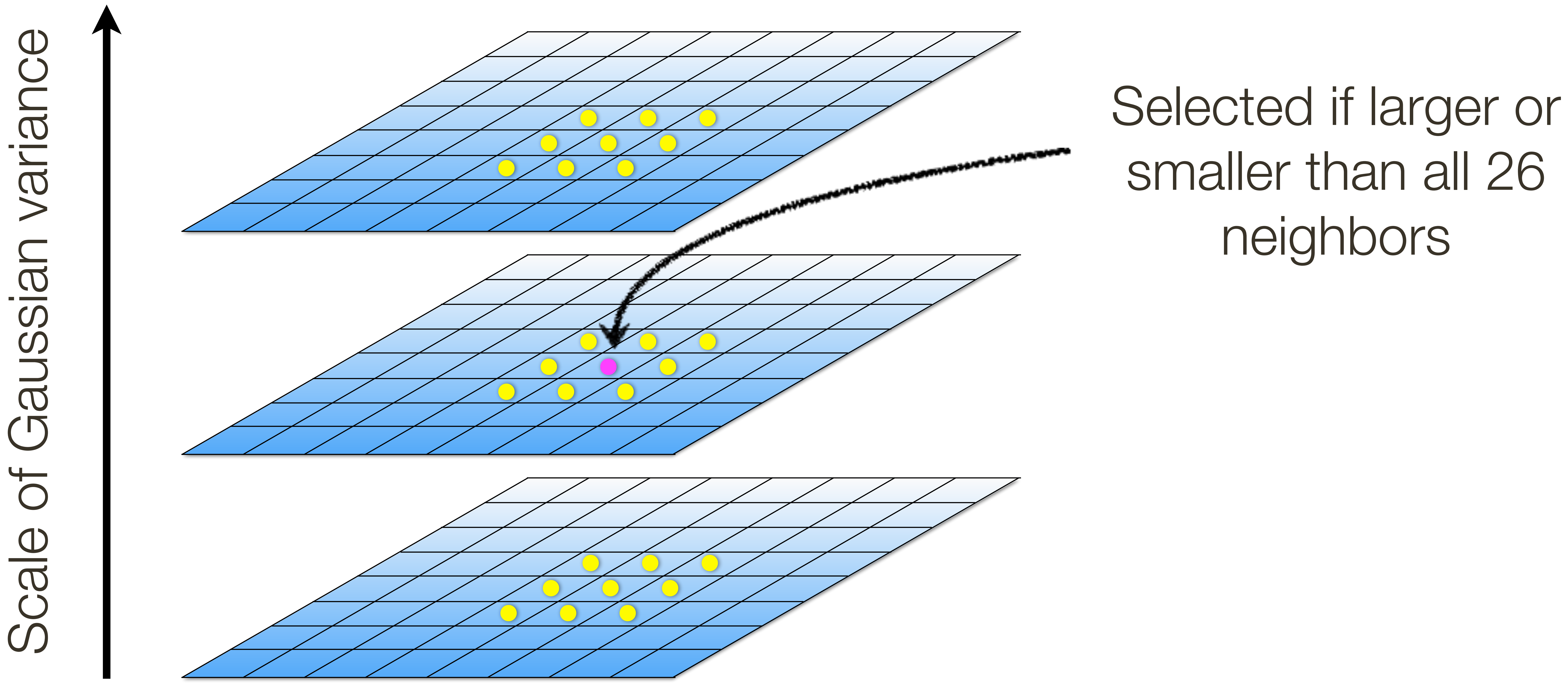


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

1. Multi-scale Extrema Detection

Detect maxima and minima of Difference of Gaussian in scale space

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

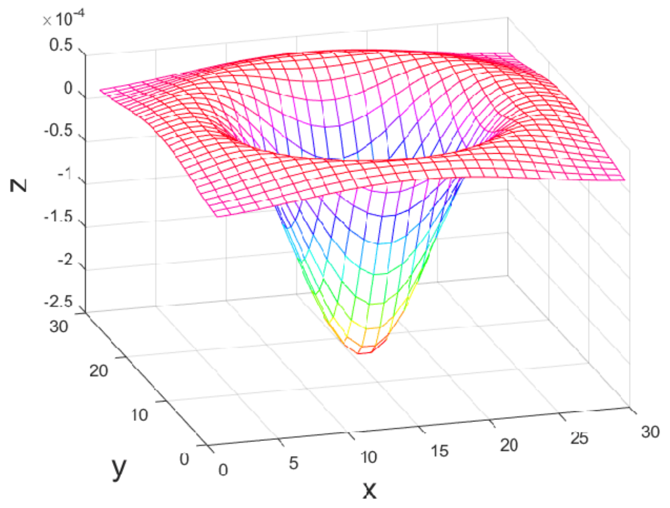


Difference of Gaussian (DoG)

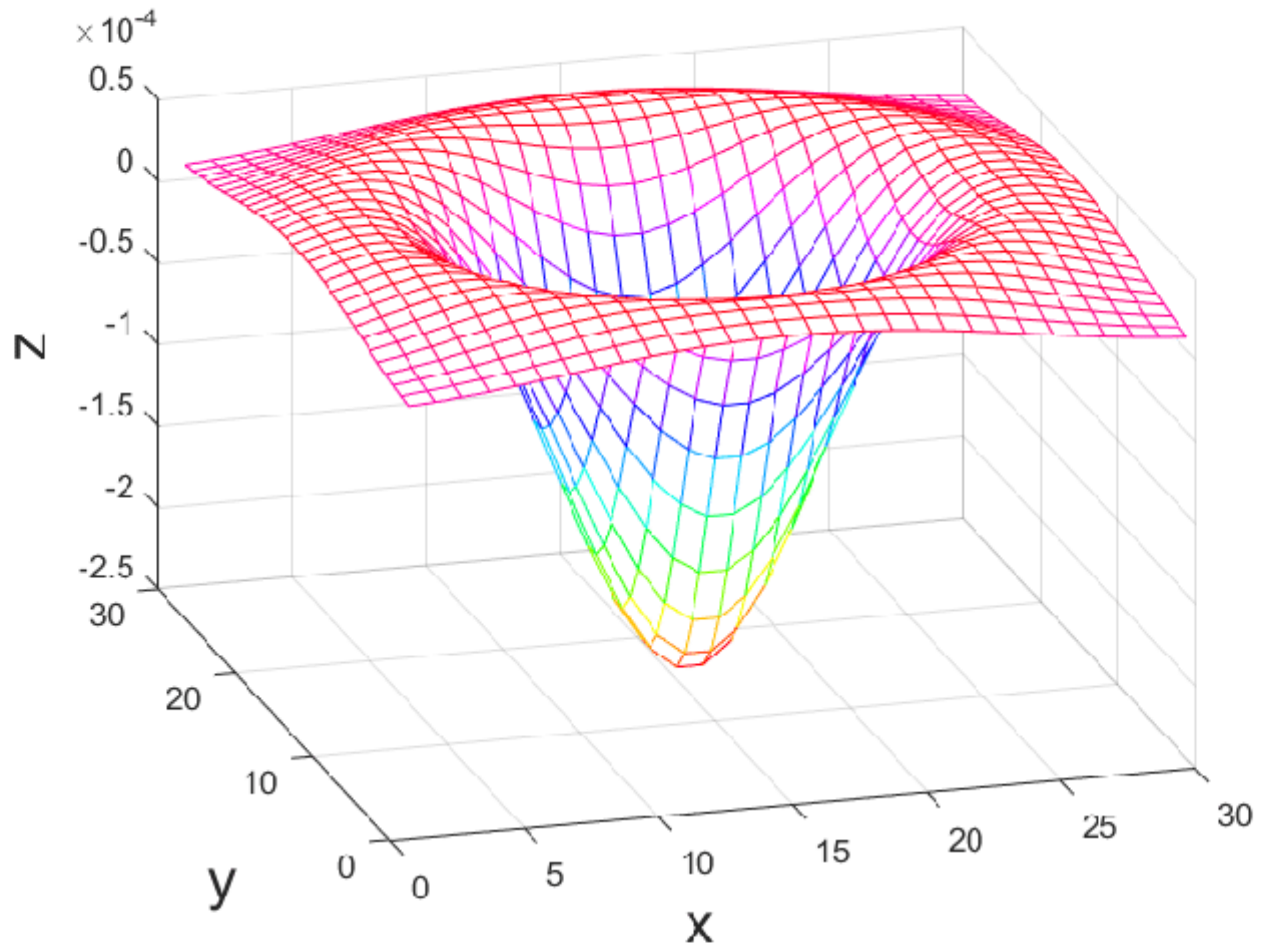
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Searching over **Scale**-space

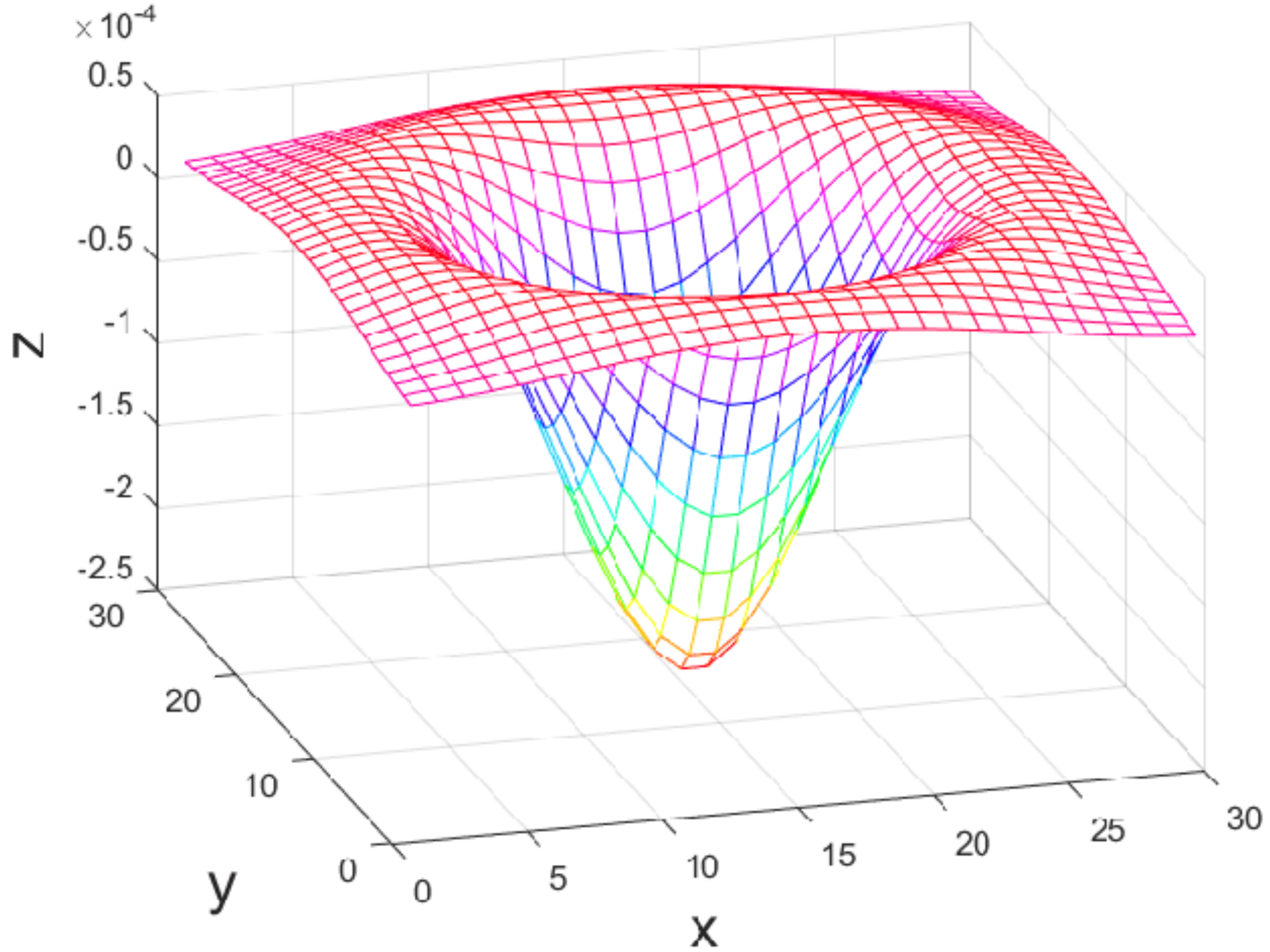
σ

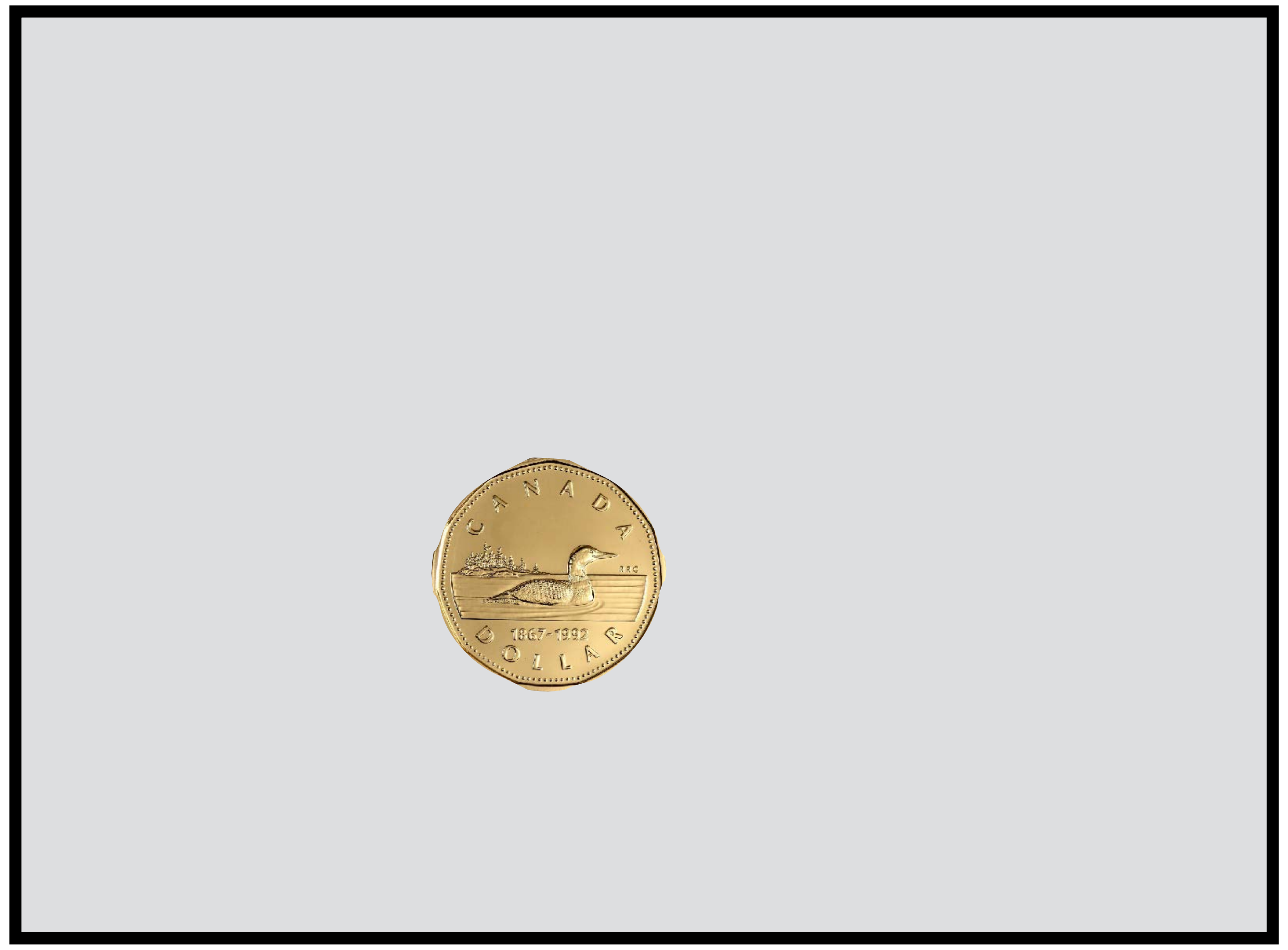


$\sigma' = 2\sigma$



$\sigma' = 3\sigma$





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

Example:

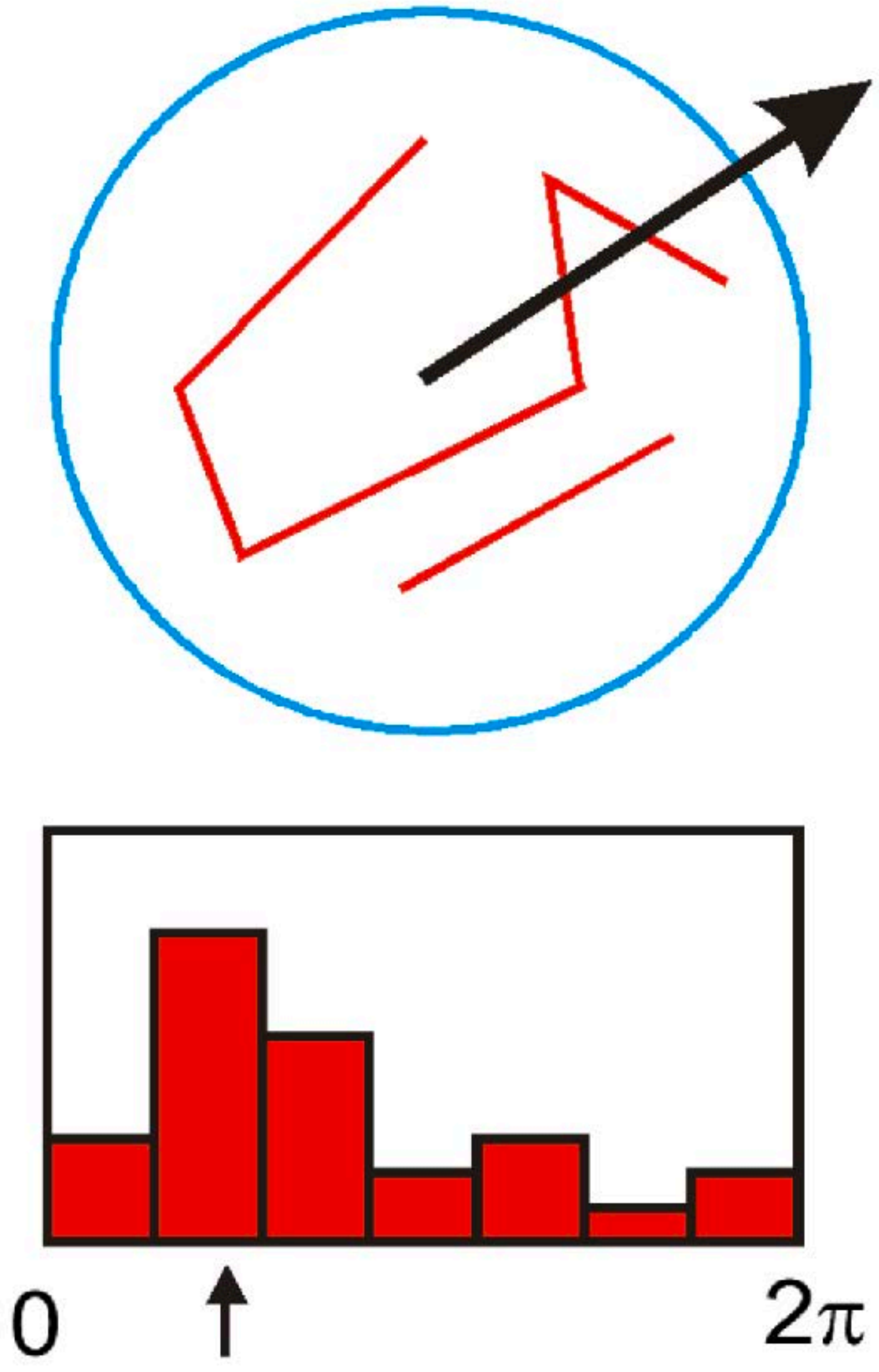
- (a) 233 × 189 image
- (b) 832 DOG extrema
- (c) 729 left after peak value threshold
- (d) 536 left after testing ratio of principal curvatures

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)

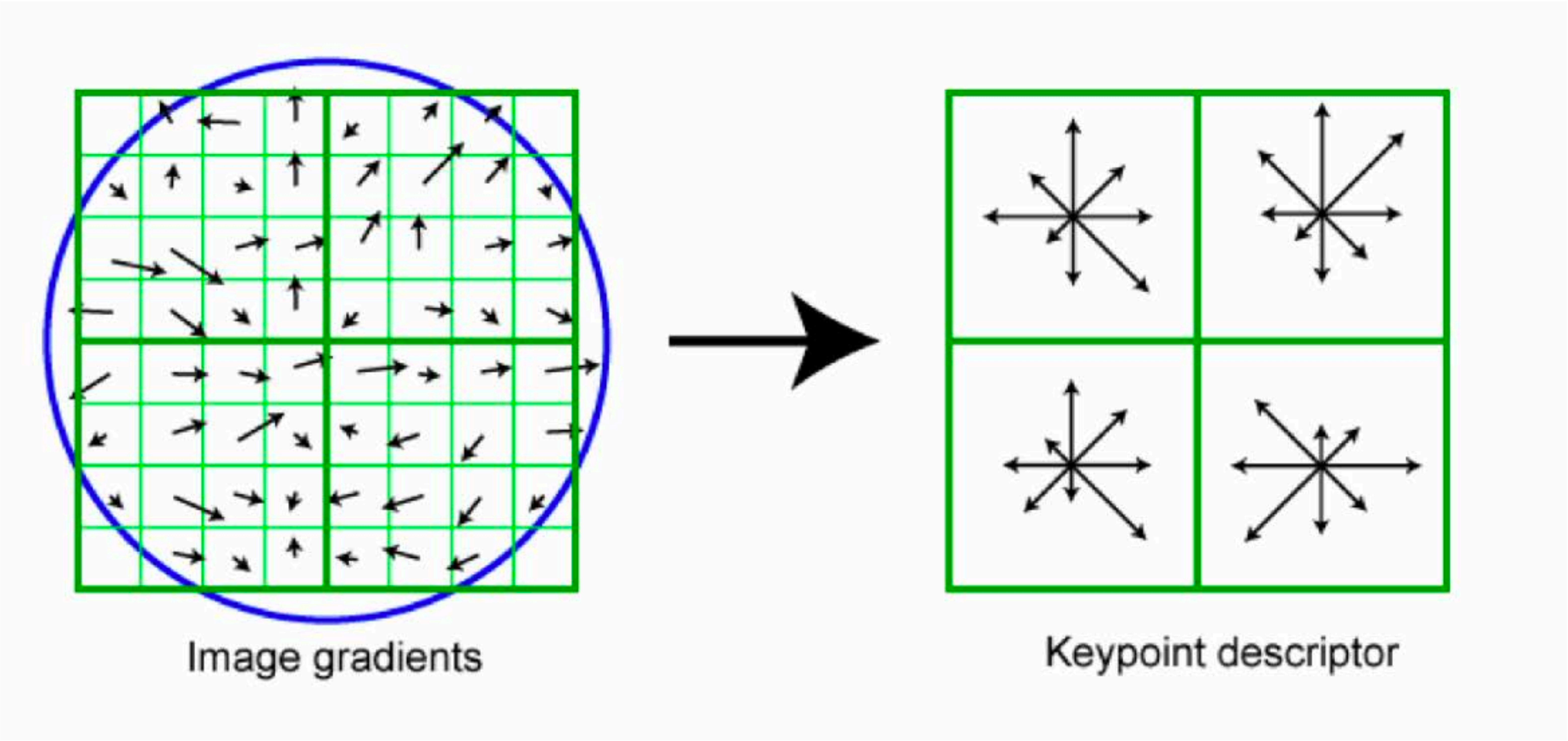


4. SIFT Descriptor

— 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

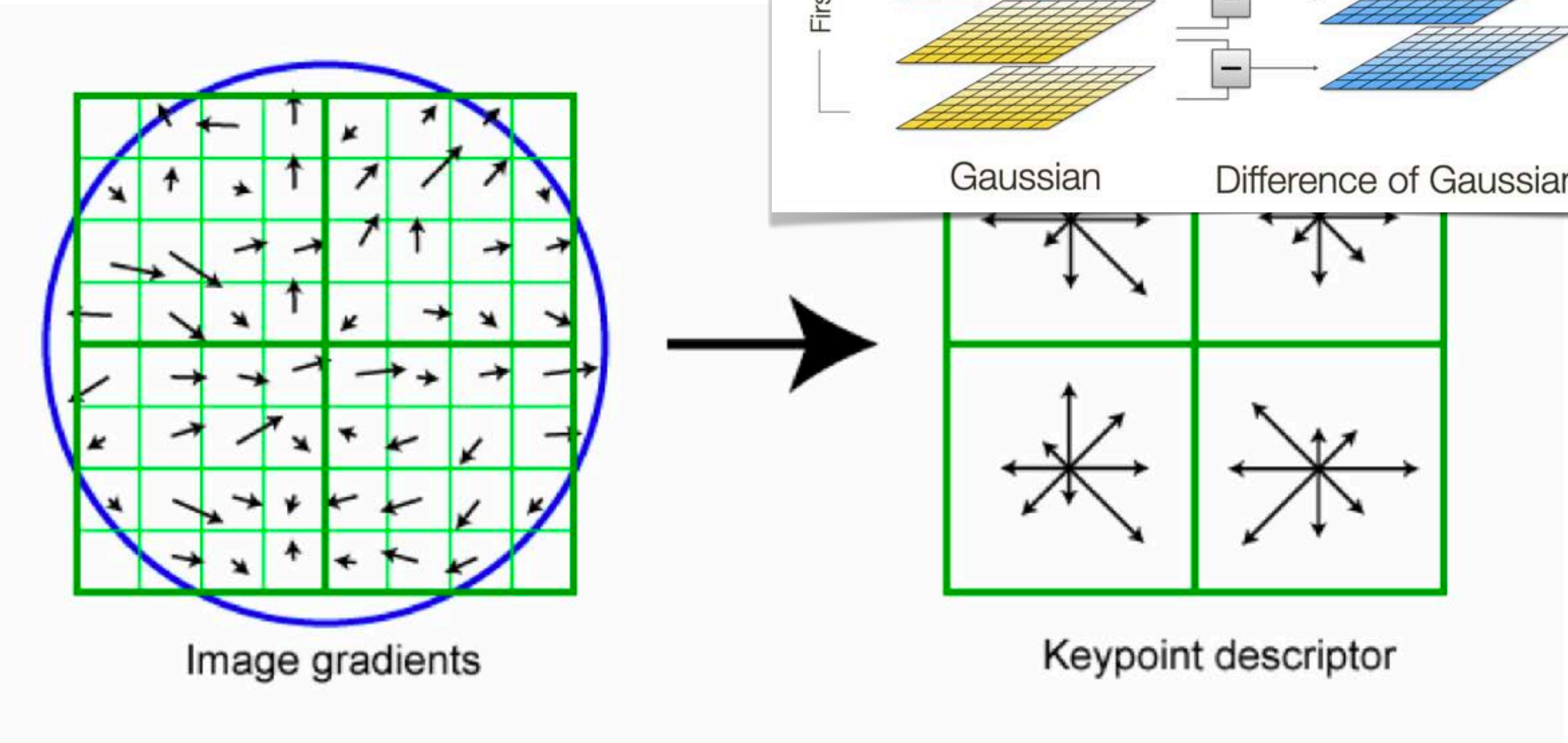


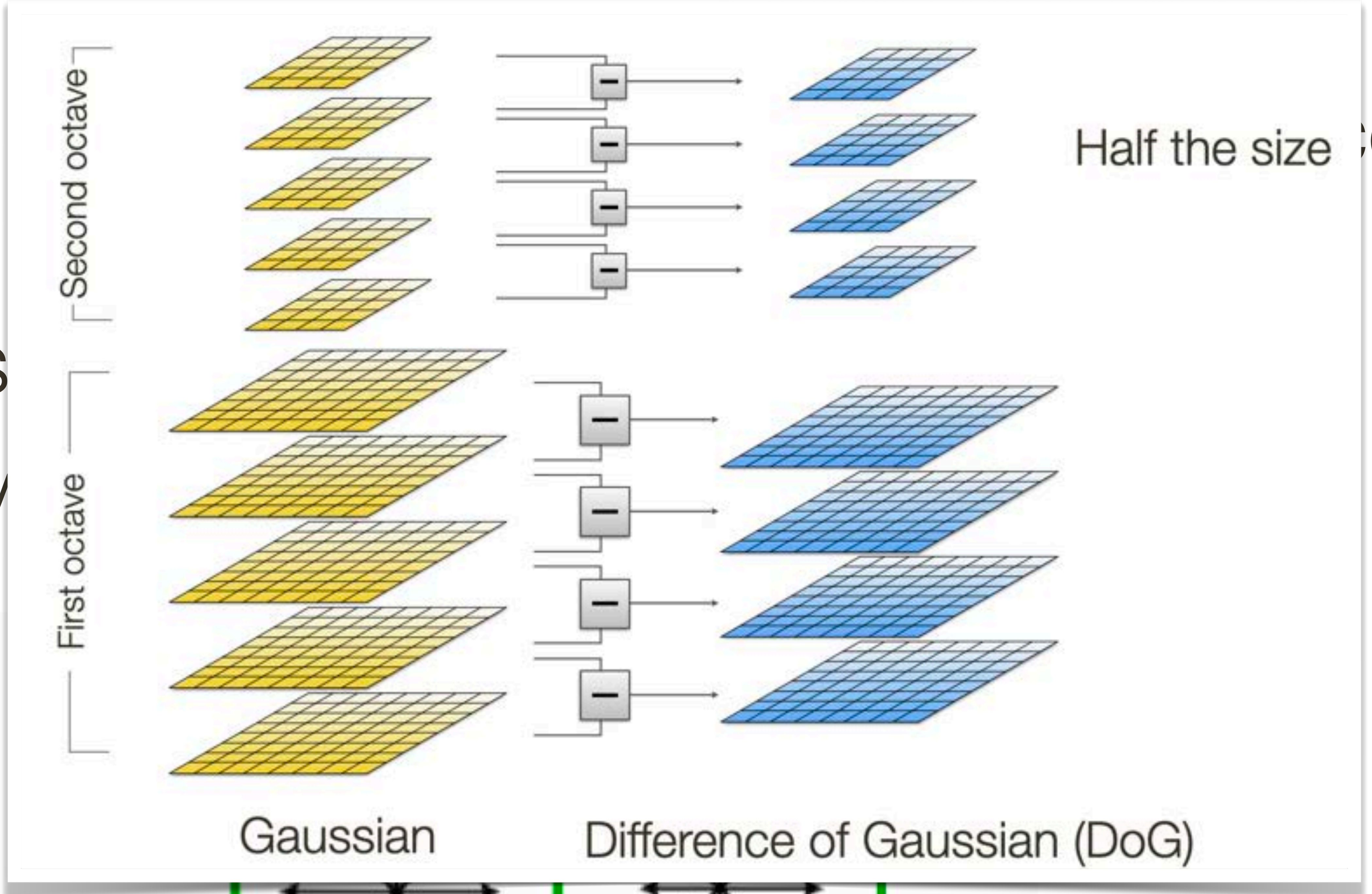
4. SIFT Descriptor

— Image gradients are sampled over 16 ... e (weighted by a Gaussian with sigma half ...

— Create array of orientation histograms

— 8 orientations × 4 × 4 histogram array





SIFT Matching

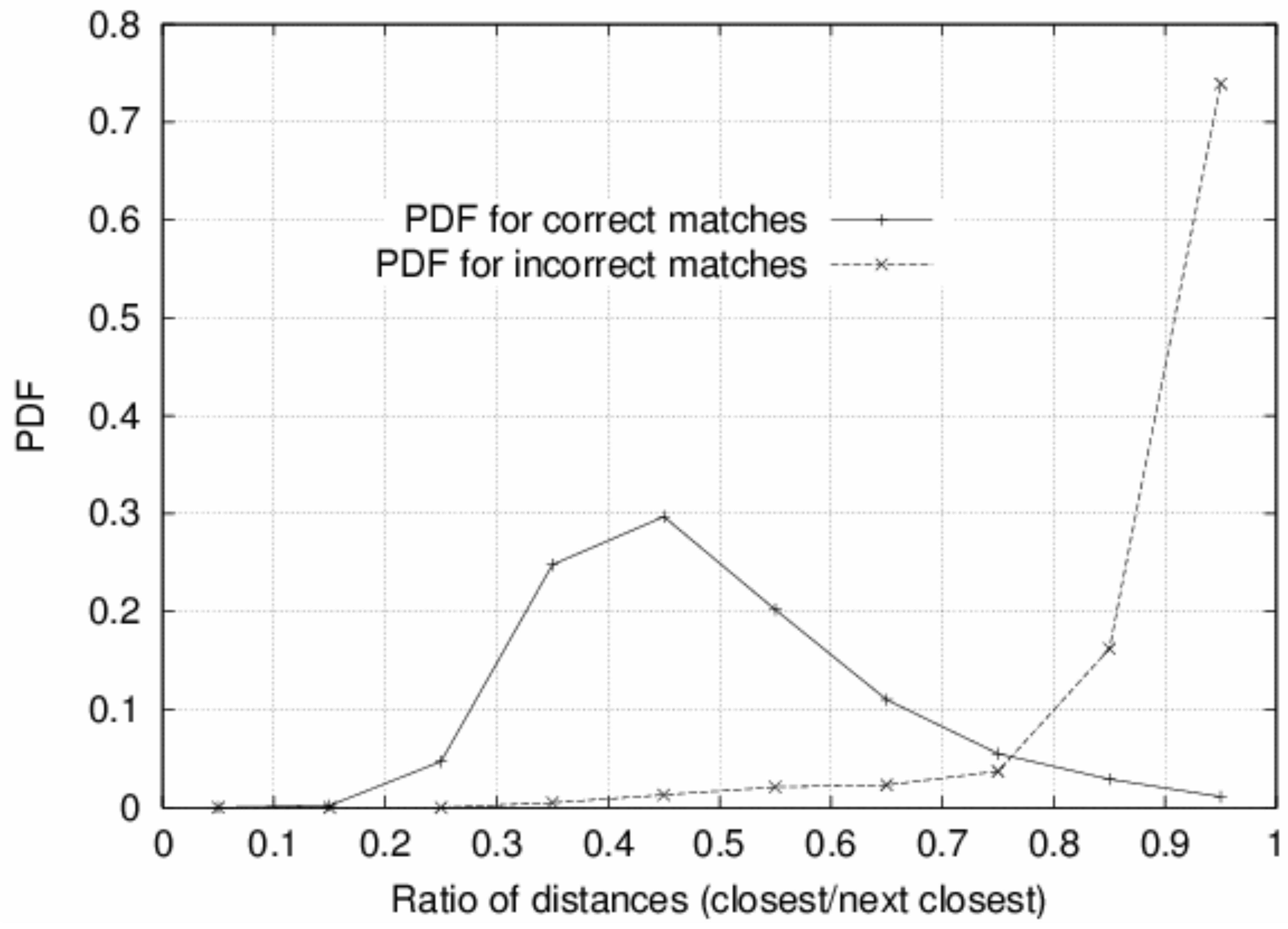
Extract features from the image …

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

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

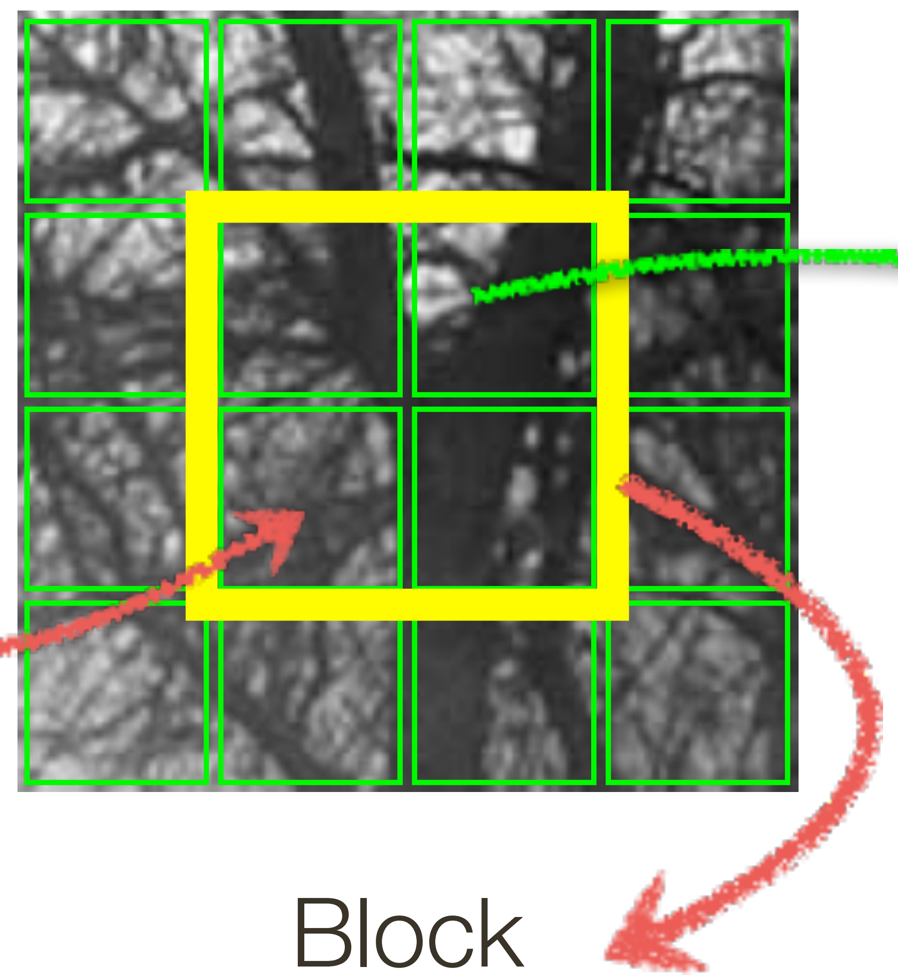


Any other ways to filter out matches?

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

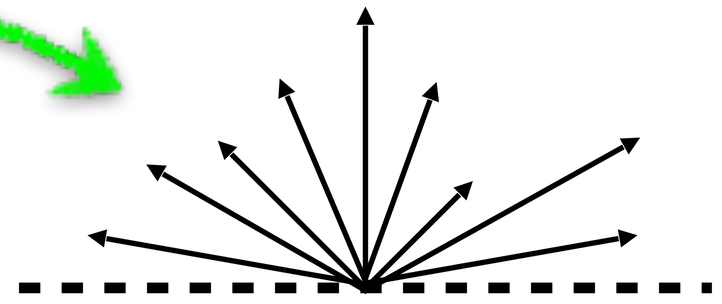
Dalal, Triggs. Histograms of Oriented Gradients for Human Detection. CVPR, 2005

Cell
(8x8 pixels)



Block
(2x2 cells)

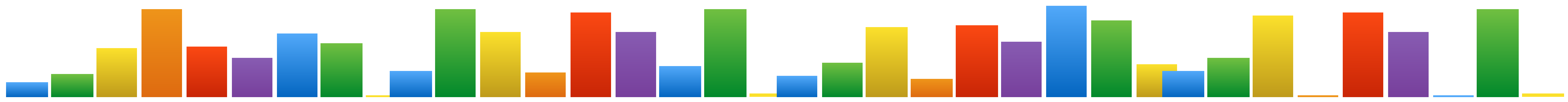
histogram of
'unsigned'
gradients



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

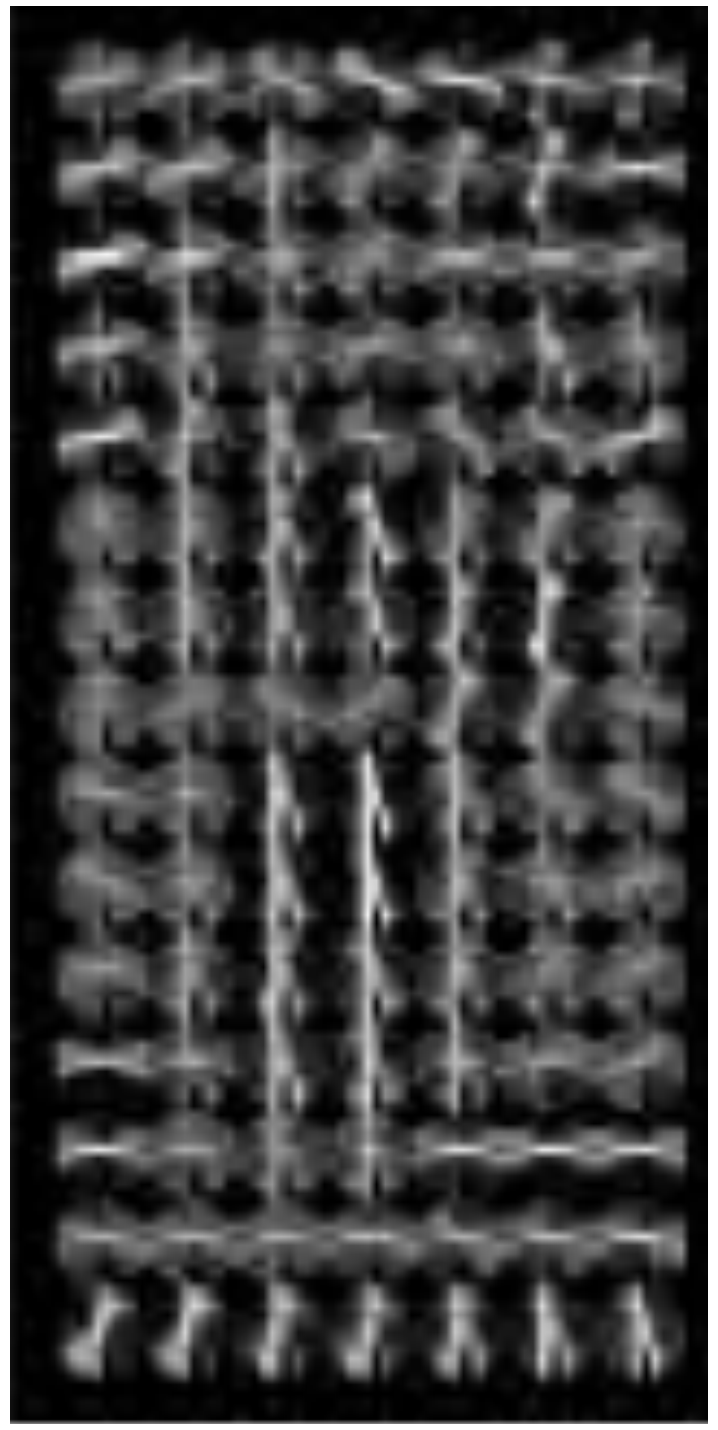
1 cell step size

128 pixels
16 cells
15 blocks

64 pixels
8 cells
7 blocks

15 x 7 x 4 x 9 =
3780

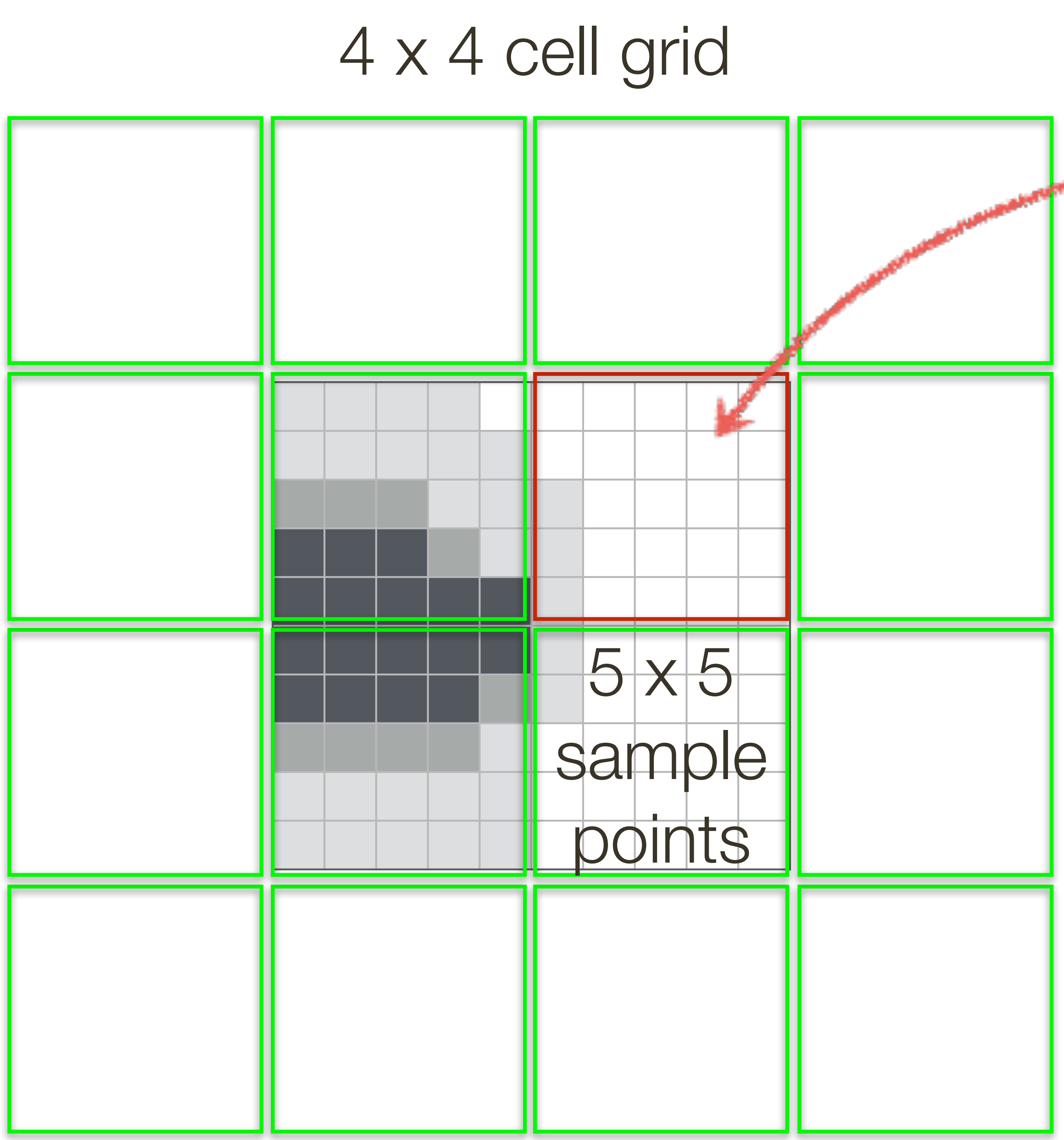
visualization



Redundant representation due to overlapping blocks

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

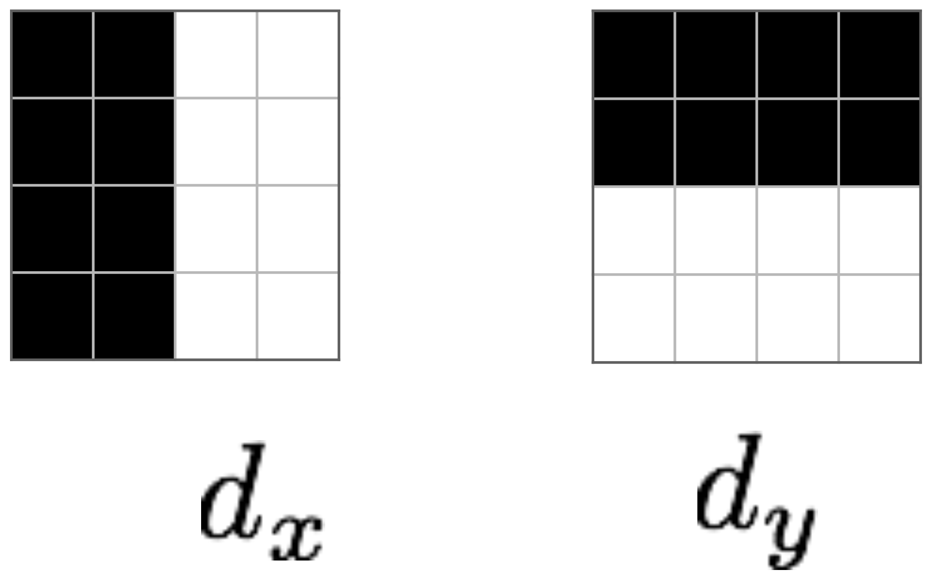
'Speeded' Up Robust Features (**SURF**)



Each cell is represented by 4 values:

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

Haar wavelets filters

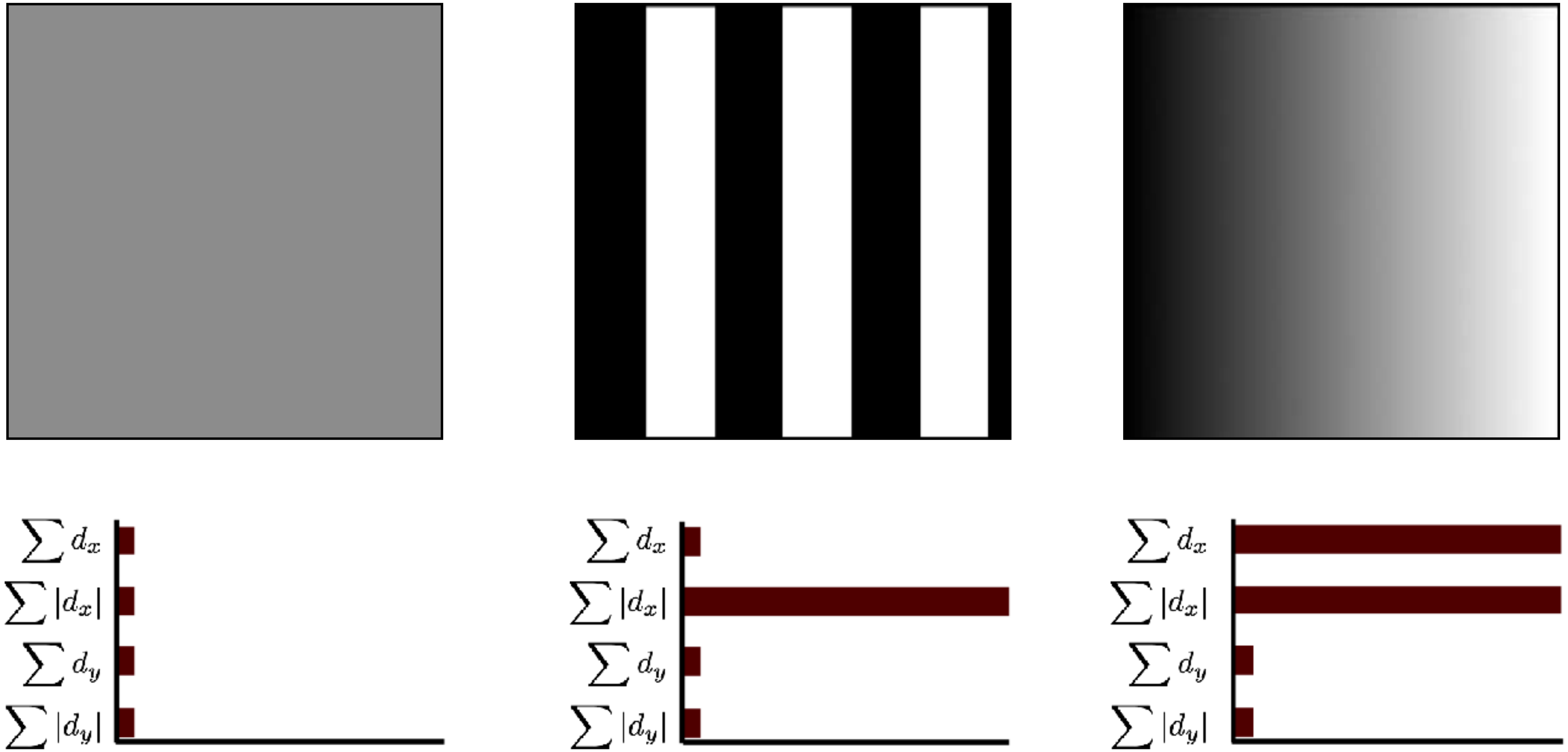


How big is the SURF descriptor?

64 dimensions

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

- Harris
- Blob (Laplacian)
- SIFT

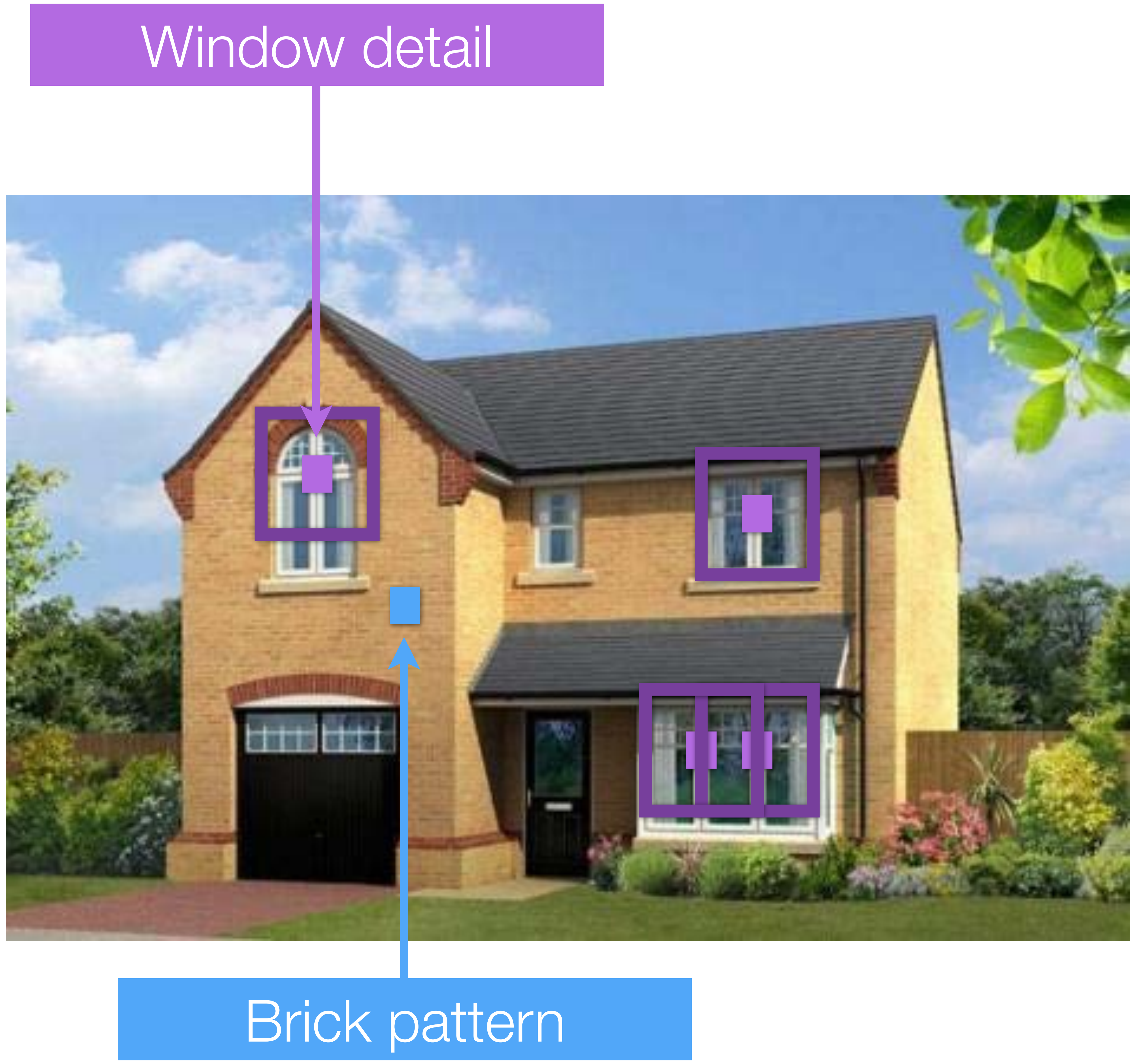
- SIFT
- HoG
- SURF

Failure Case: **Repetitive** Structures

Repetitive structures cause problems for feature matching

Multiple locations in an image provide good matches and have similar matching scores

They are particularly common in man-made environments



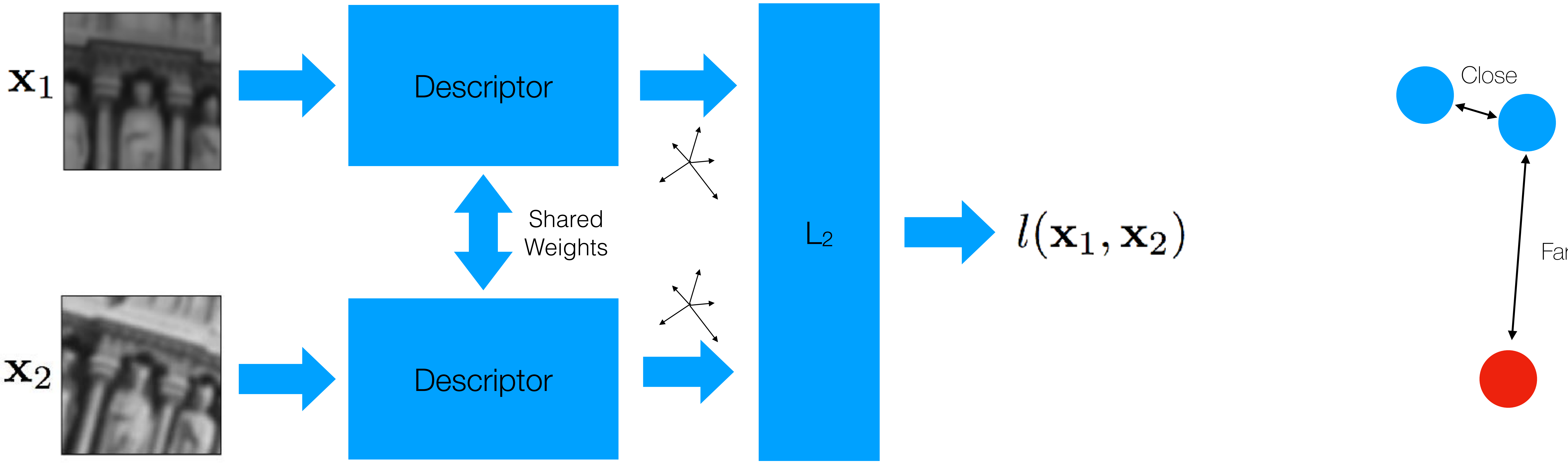
Learning Descriptors

Descriptor design as a learning (embedding) problem

[Winder Brown 2007]

DeepDesc [ICCV 2015]

Learning an “embedding”

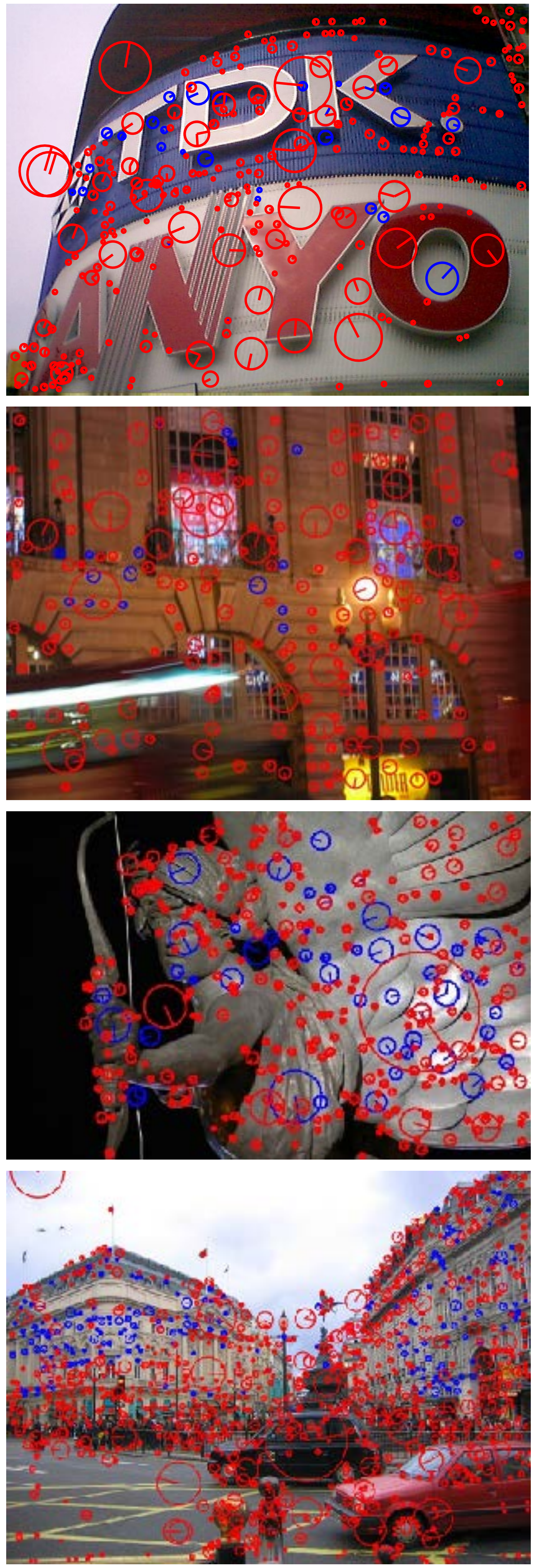


Minimize the distance for corresponding matches.

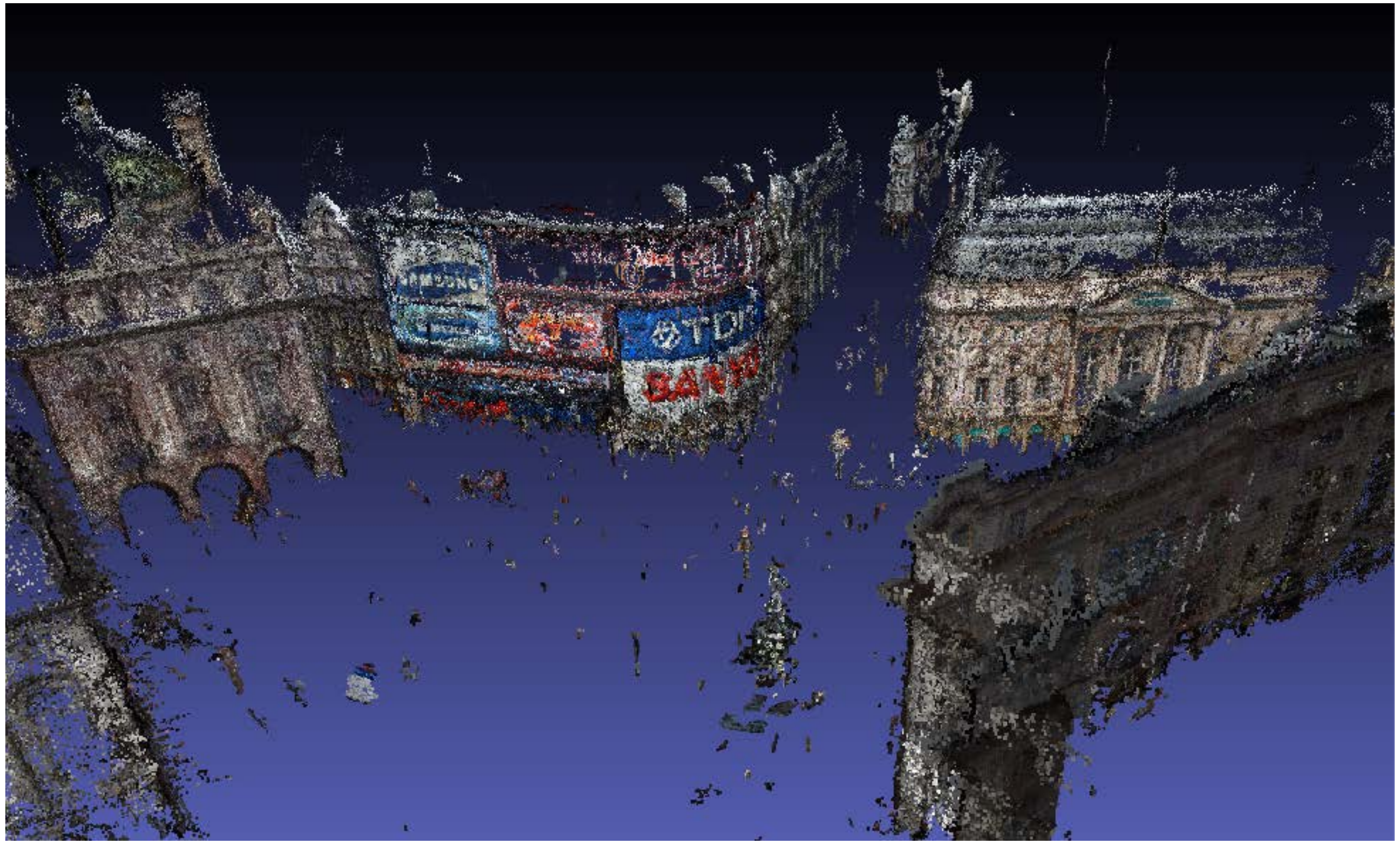
Maximize it for non-corresponding patches.

Slide credits, Eduard Trulls

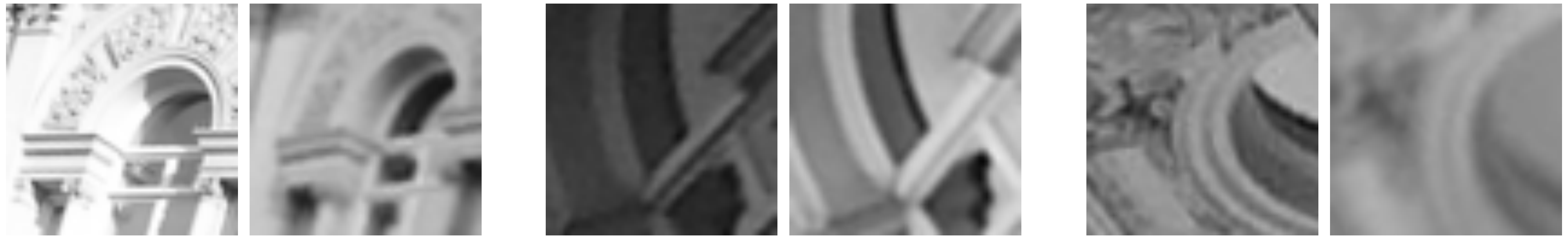
Learning with SfM dataset



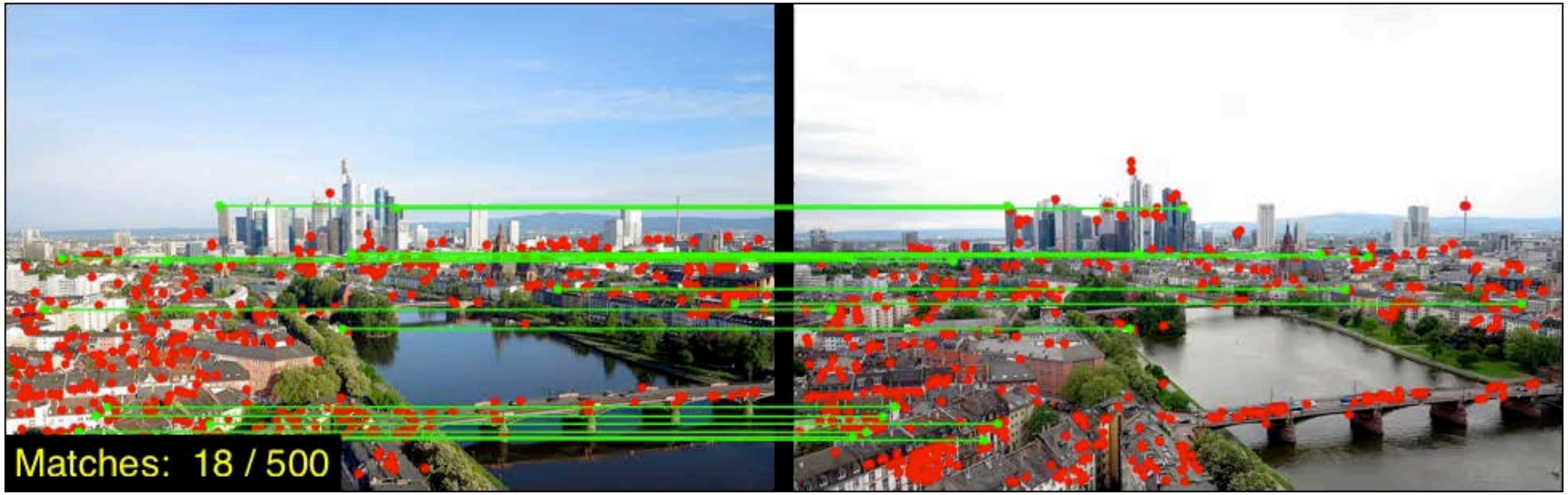
Training set #1:



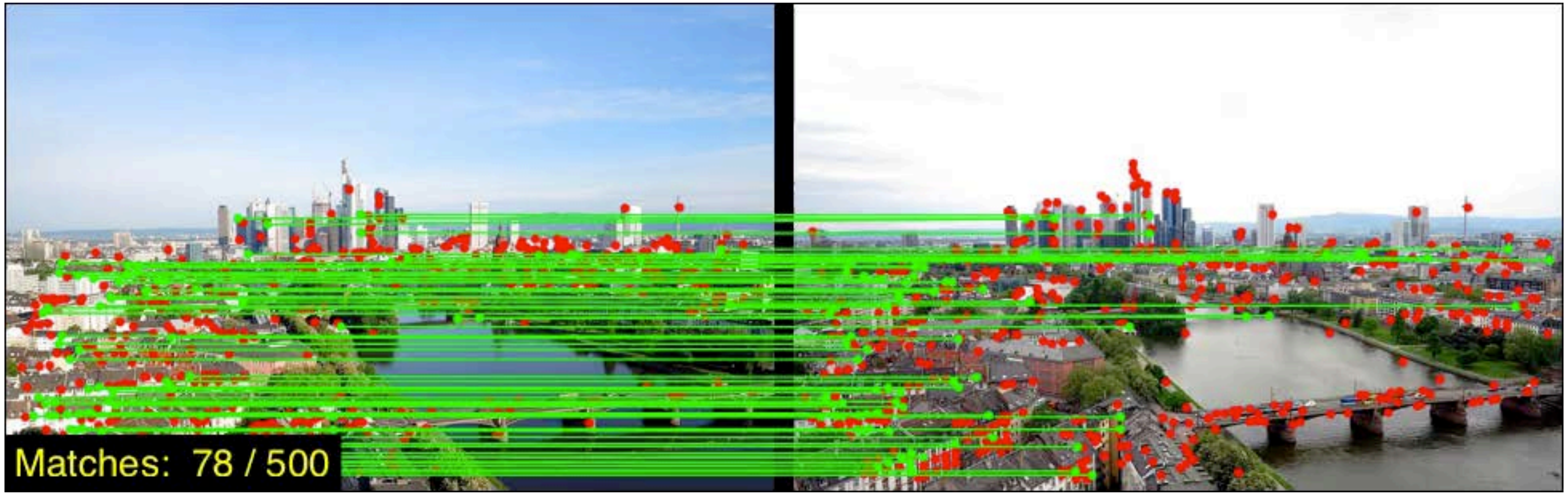
3k images, 59k unique points, 380k



Learned vs SIFT

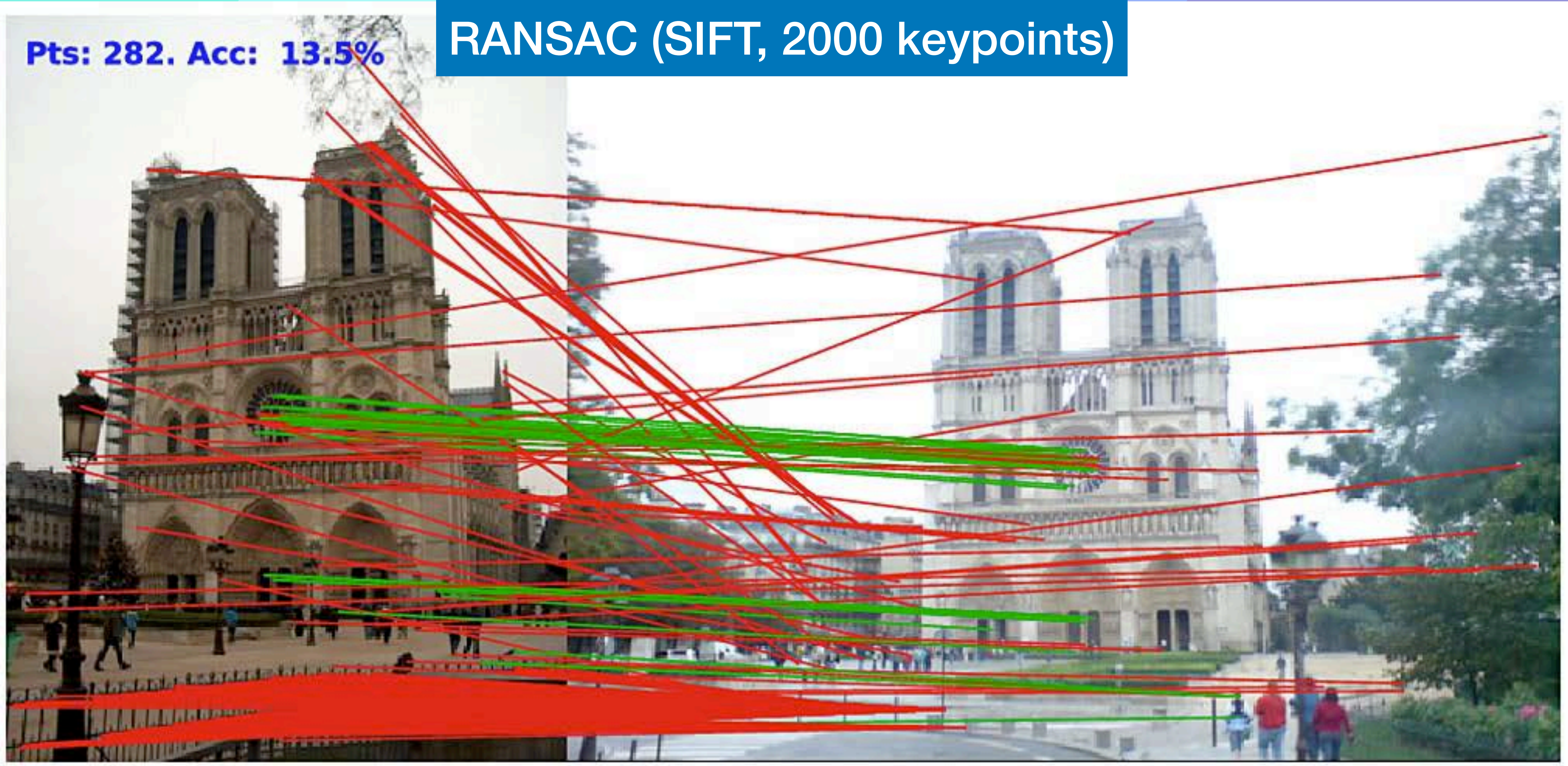


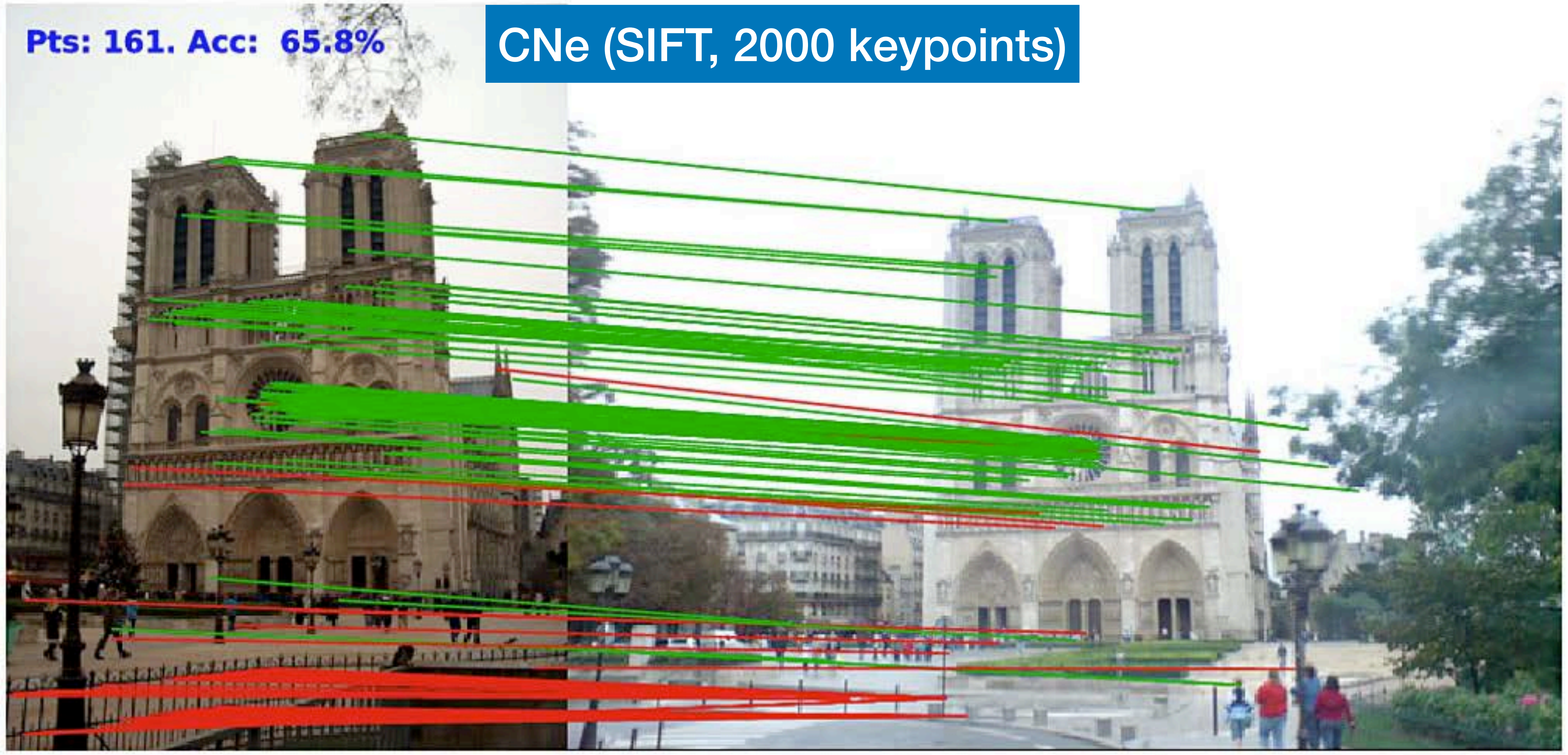
SIFT. Average: **23.1** matches



LIFT. Average: **60.6** matches

Learning to Filter





With COTR, we find where the four corners of the first frame went.
We visualize the results by augmenting another painting on top.

Image 1

Image 2

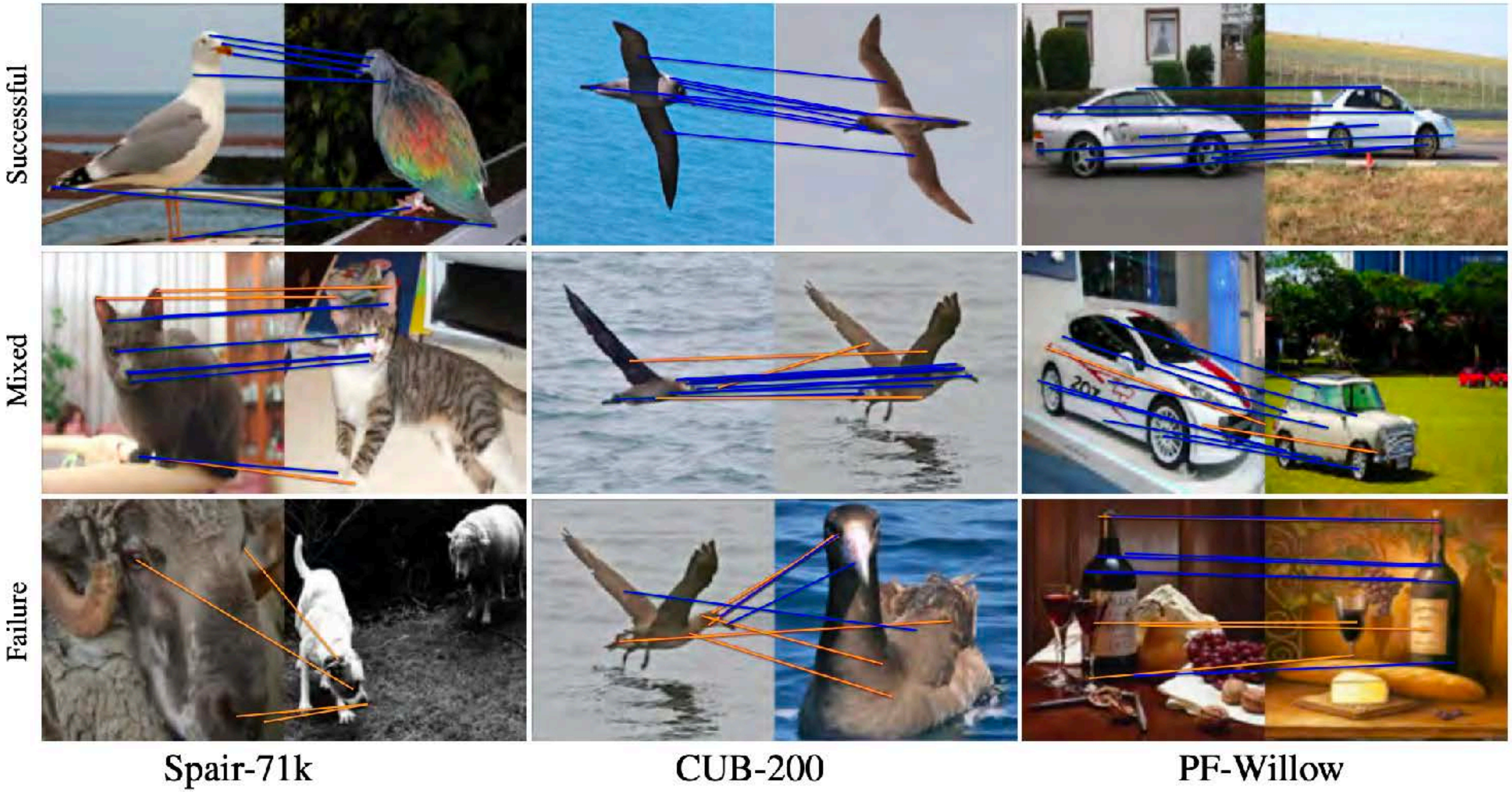
With COTR, we find dense correspondences, which we can reconstruct a dense 3D model from just two calibrated views.


https://youtu.be/OaLxTz1Yw7M
Facial landmarks can also be tracked by using COTR easily
by finding correspondences with the first frame.
NOTE that COTR was never trained on faces or deformable surfaces.

https://youtu.be/_-cjO3KqJ_w

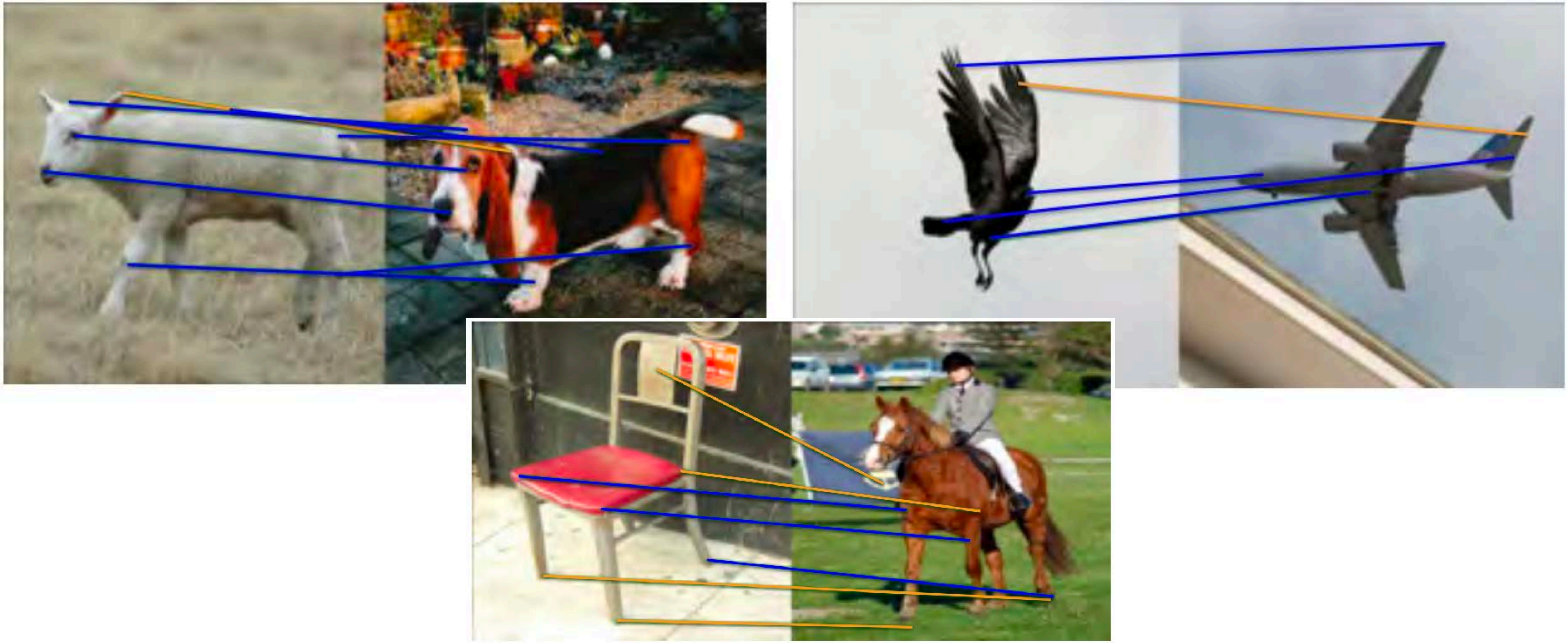
Even with the crazy transformations that we never trained COTR for, it finds good correspondences amazingly well.

“semantic” correspondences



 [Hedlin, Sharma, Mahajan, Isack, Kar, Tagliasacchi, Yi, NeurIPS, 2023]

“semantic” correspondences



 [Hedlin, Sharma, Mahajan, Isack, Kar, Tagliasacchi, Yi, NeurIPS, 2023]

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)