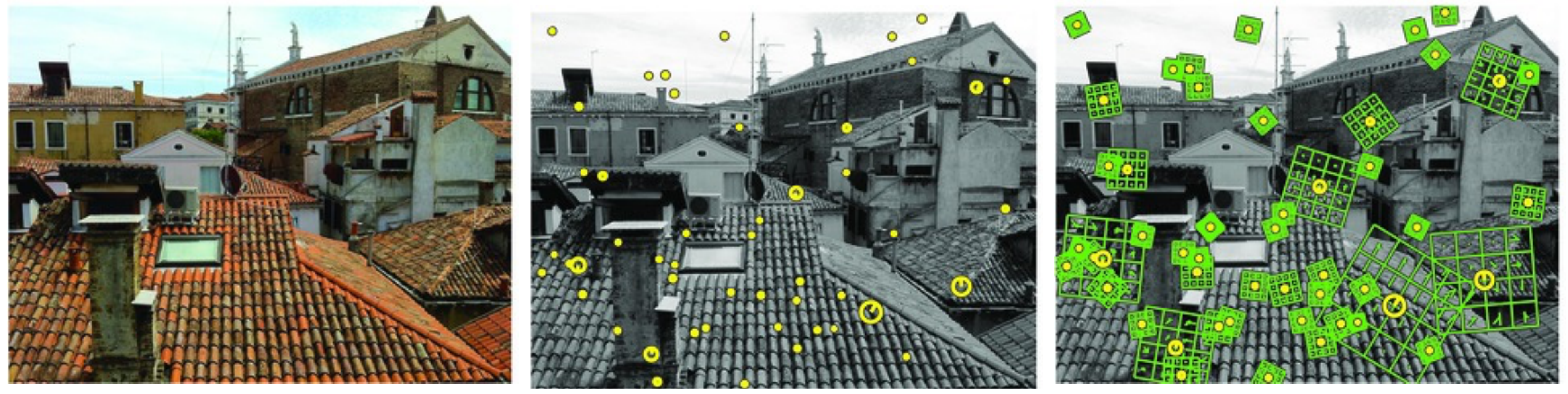
THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision



Lecture 15: Scale Invariant Features (SIFT)

Menu for Today (**February 19, 2018**)

Topics:

- Scale Invariant Feature Transform (SIFT)
- SIFT detector, descriptor
- HOG, SURF descriptors
- Object detection with SIFT

Redings:

- **Today's** Lecture: Forsyth & Ponce (2nd ed.) 5.4
  "Distinctive Image Features for Scale-Invariant Keypoints
- **Next** Lecture: Forsyth & Ponce (2nd ed.) 10.4.2, 10.1, 10.2

Reminders:

- **Assignment 3**: Texture Syntheis is **out,** due on **October 29th**

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

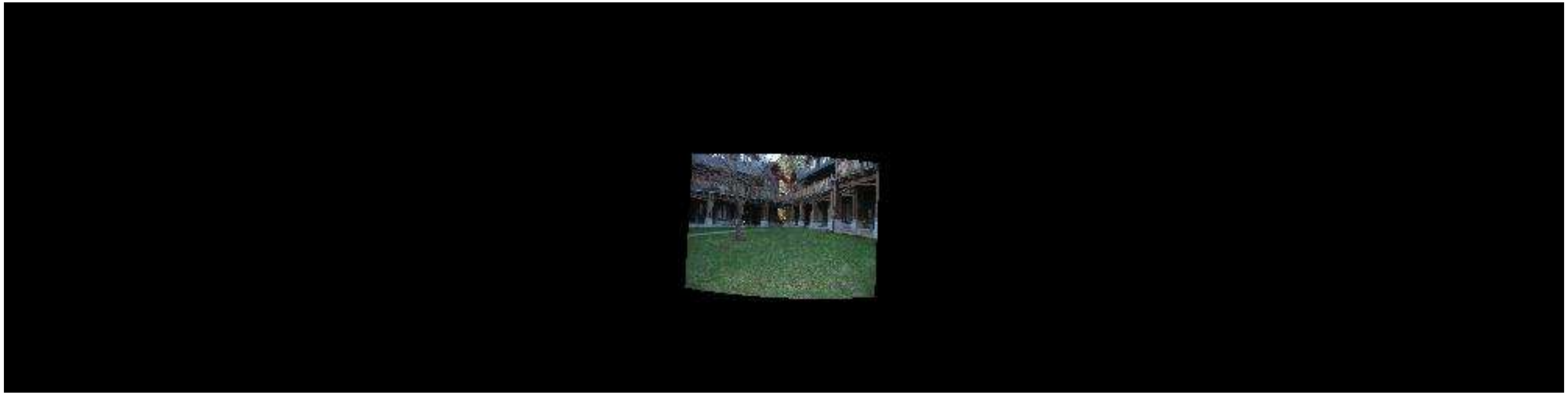


Figure Credit: Matthew Brown and David Lowe

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

Lecture 14: Re-cap

— Human **colour perception**

  — colour matching experiments

  — additive and subtractive matching

  — principle of trichromacy

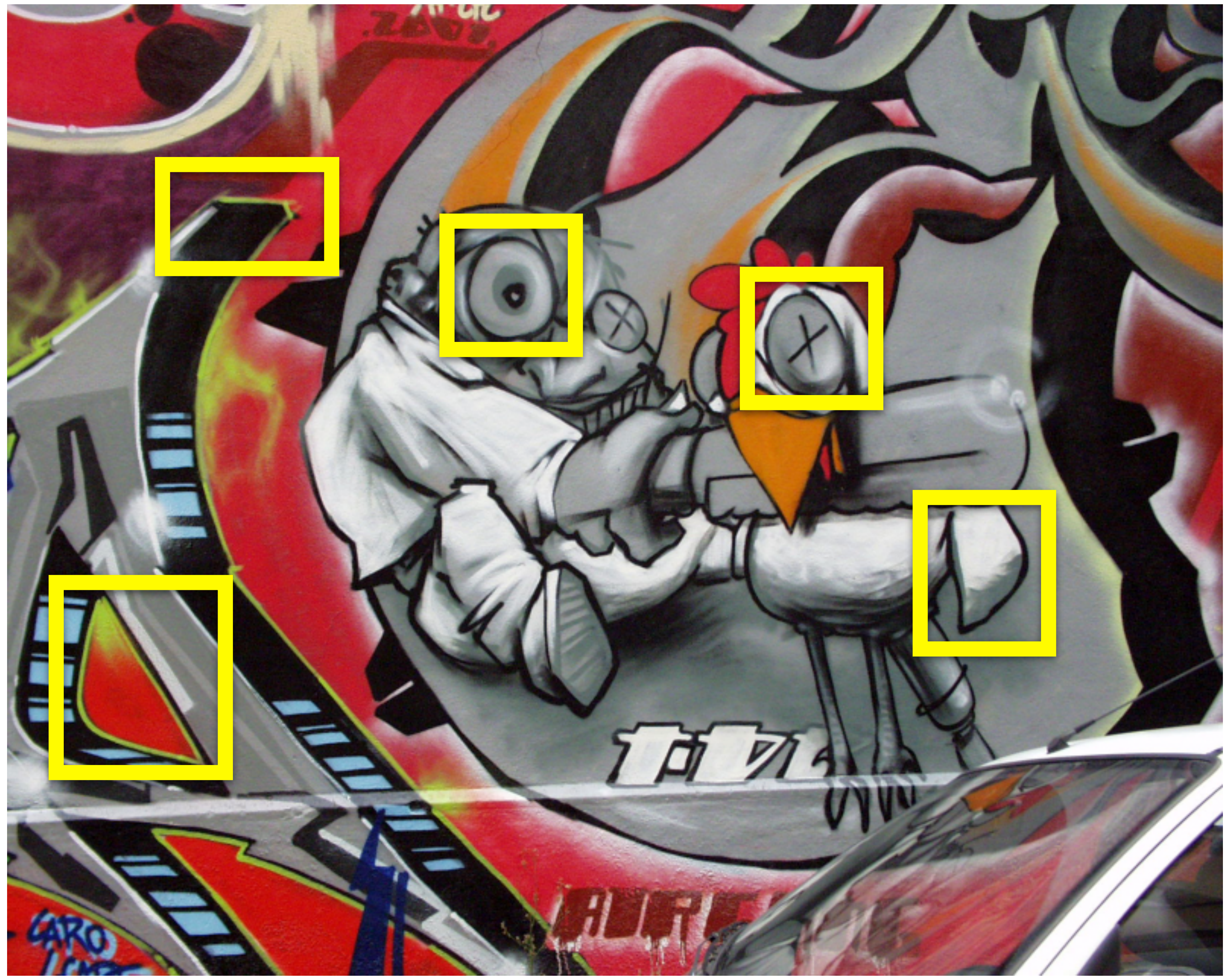
— **RGB** and **CIE XYZ** are linear colour spaces

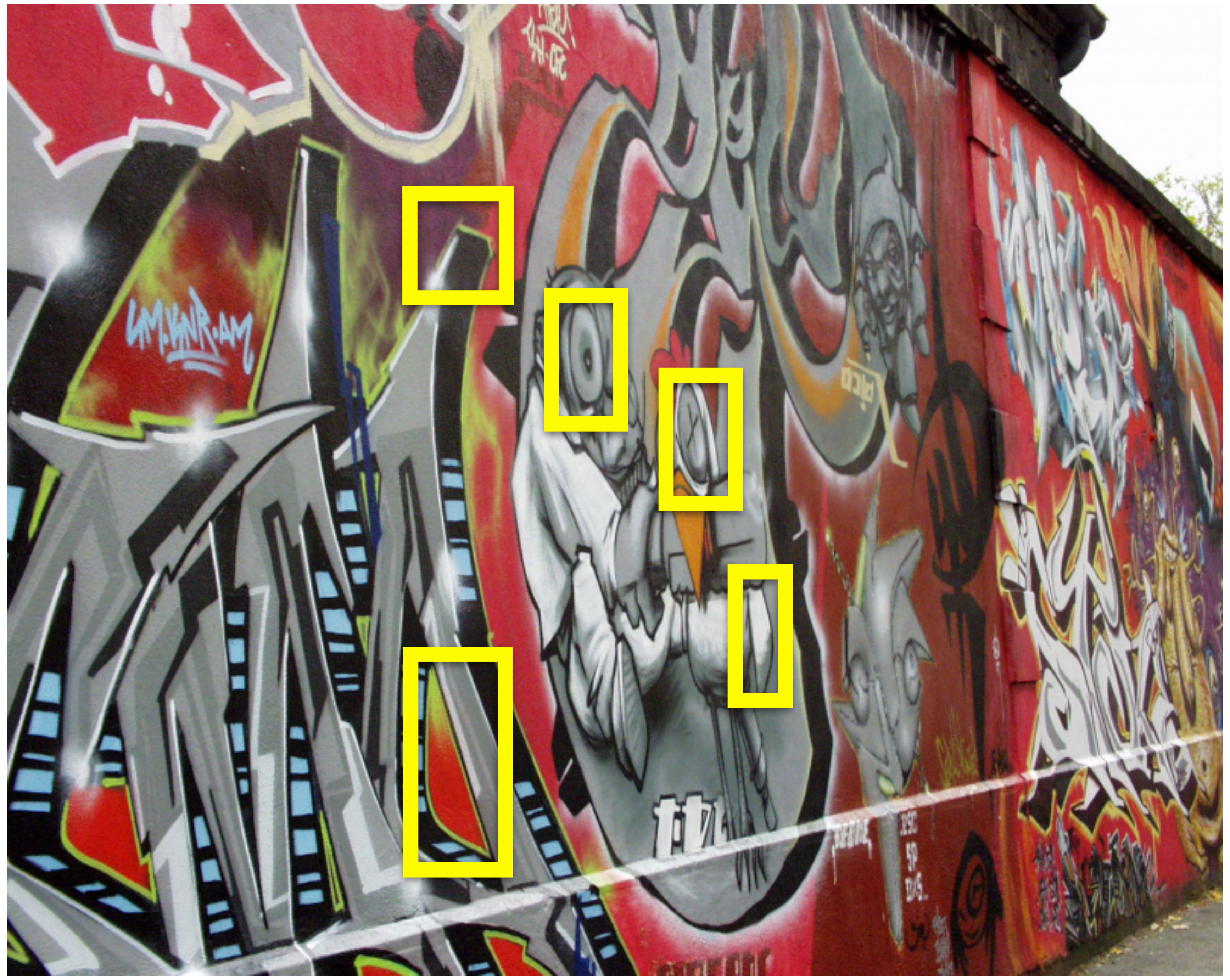
— **Uniform colour space**: differences in coordinates are a good guide to differences in perceived colour

— **HSV** colour space: more intuitive description of colour for human interpretation

— (Human) **colour constancy**: perception of intrinsic surface colour under different colours of lighting

Back to **Good Local Features**





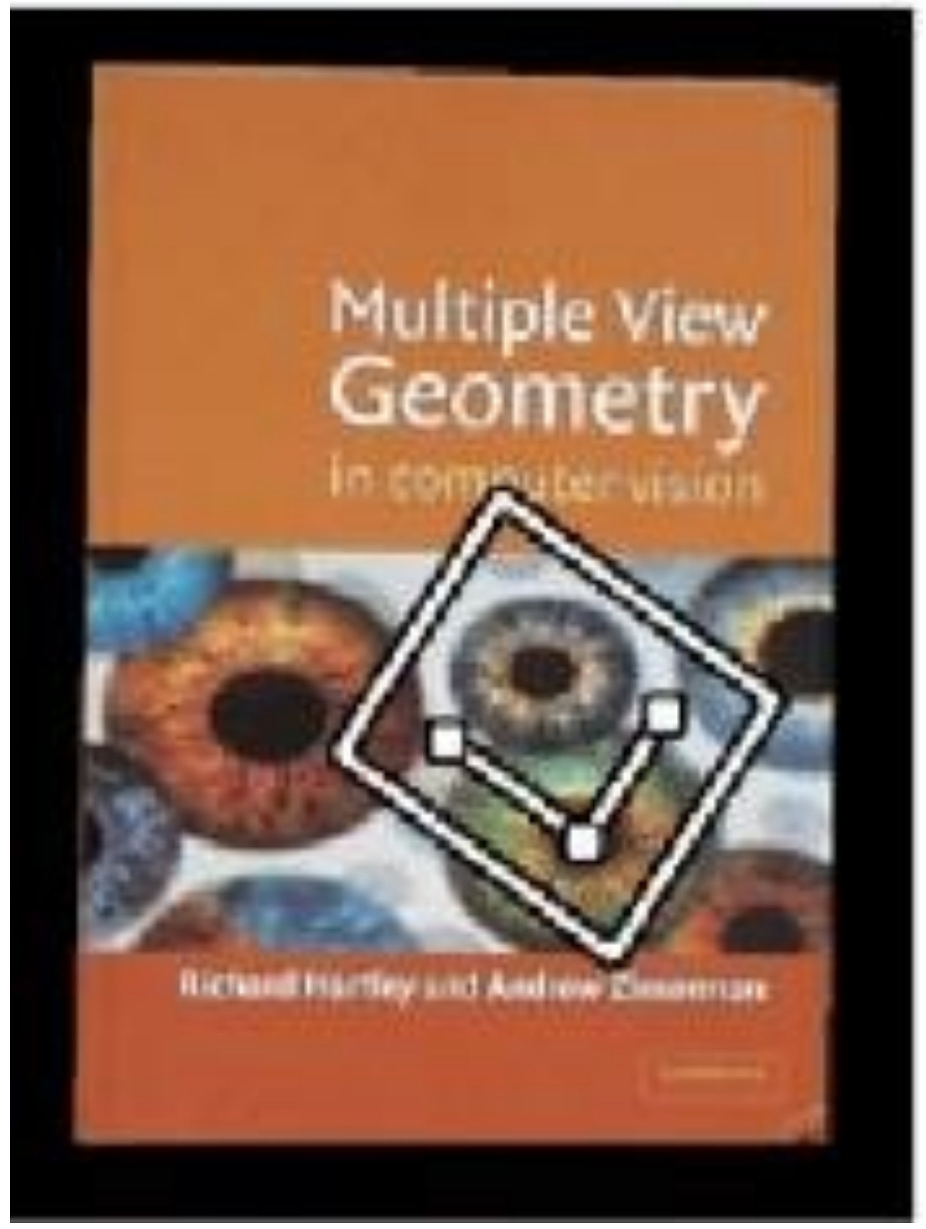
Where are the good features, and
how do we match them?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Photometric Transformations

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Geometric Transformations



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?

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)

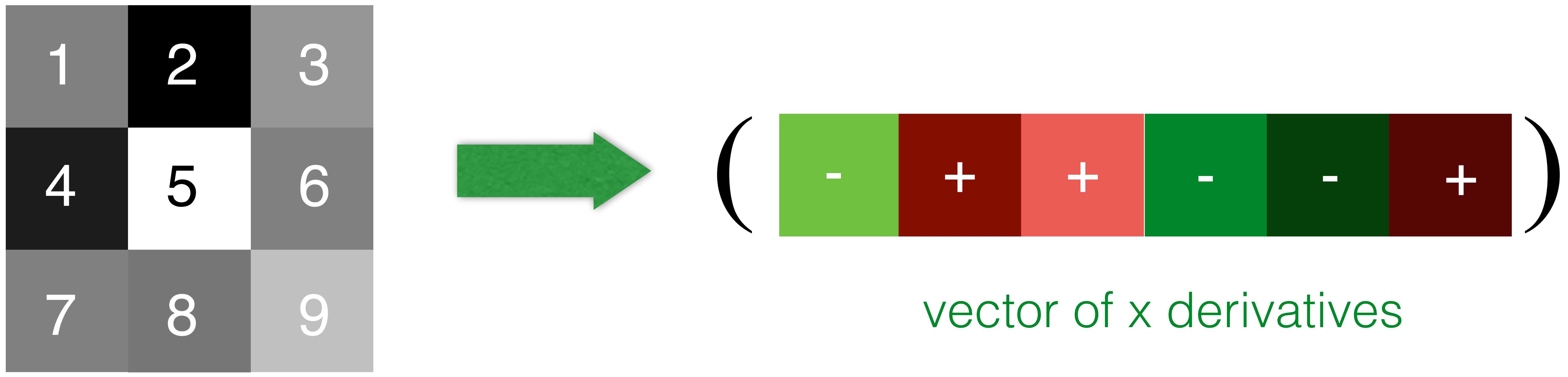
What are the problems?

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

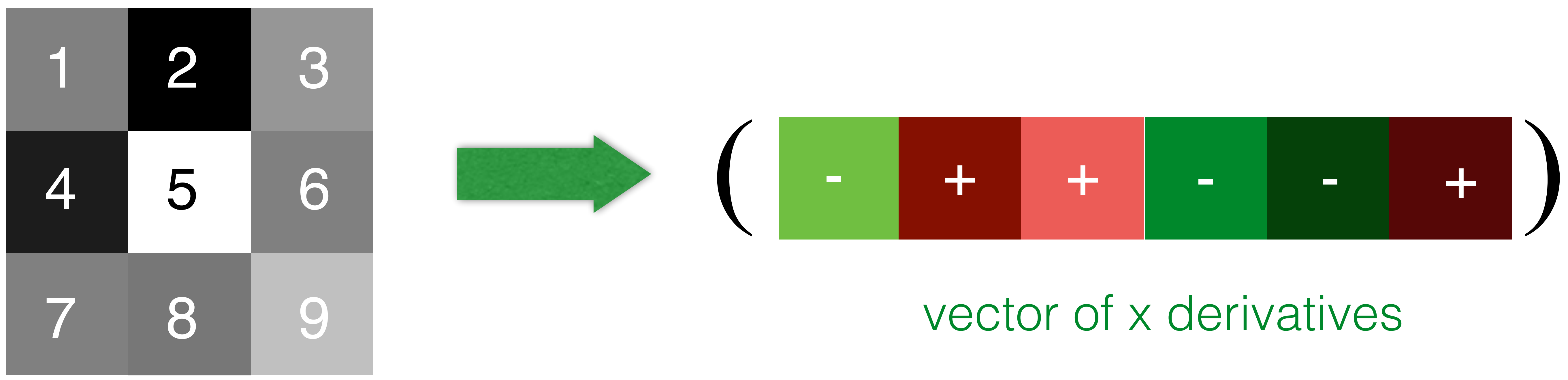
Feature is invariant to absolute intensity values

What are the problems?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

Use pixel differences



vector of x derivatives

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)

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"	sparse (0D)	locally distinct features	Harris, SIFT

Object **Recognition** with Invariant Features

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

Applications:

— Industrial automation and inspection

— Mobile robots, toys, user interfaces

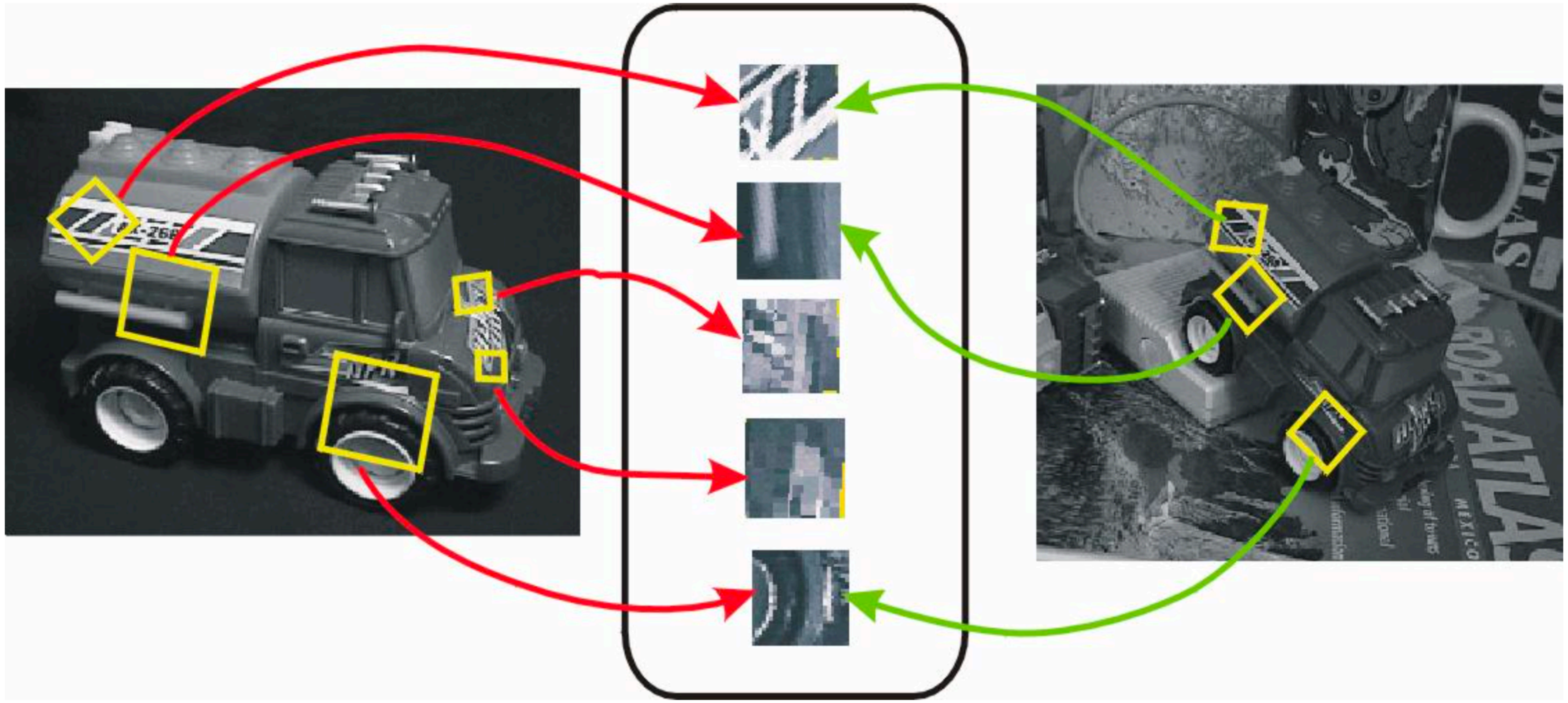
— Location recognition

— 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

Advantages of Invariant Local Features

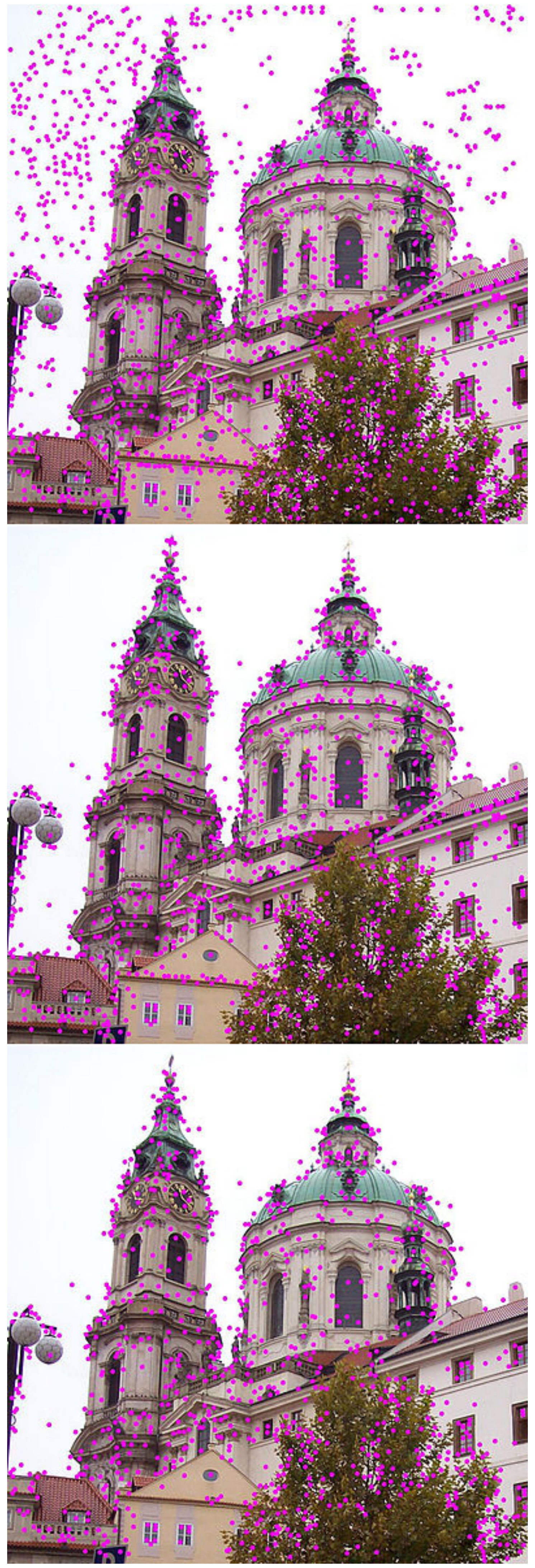
Locality: features are local, so robust to occlusion and clutter (no prior segmentation)

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**)

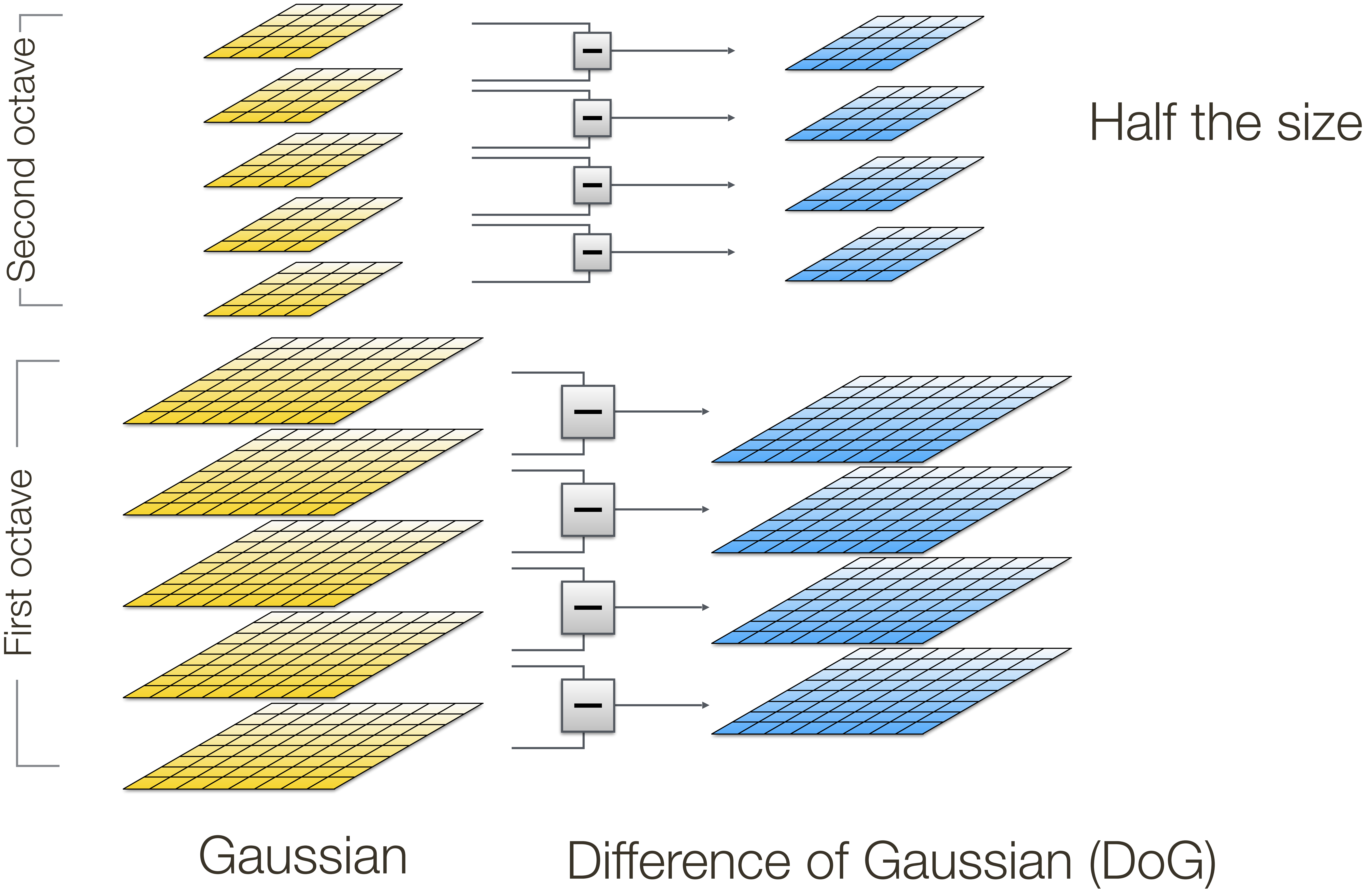


SIFT describes both a **detector** and **descriptor**

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

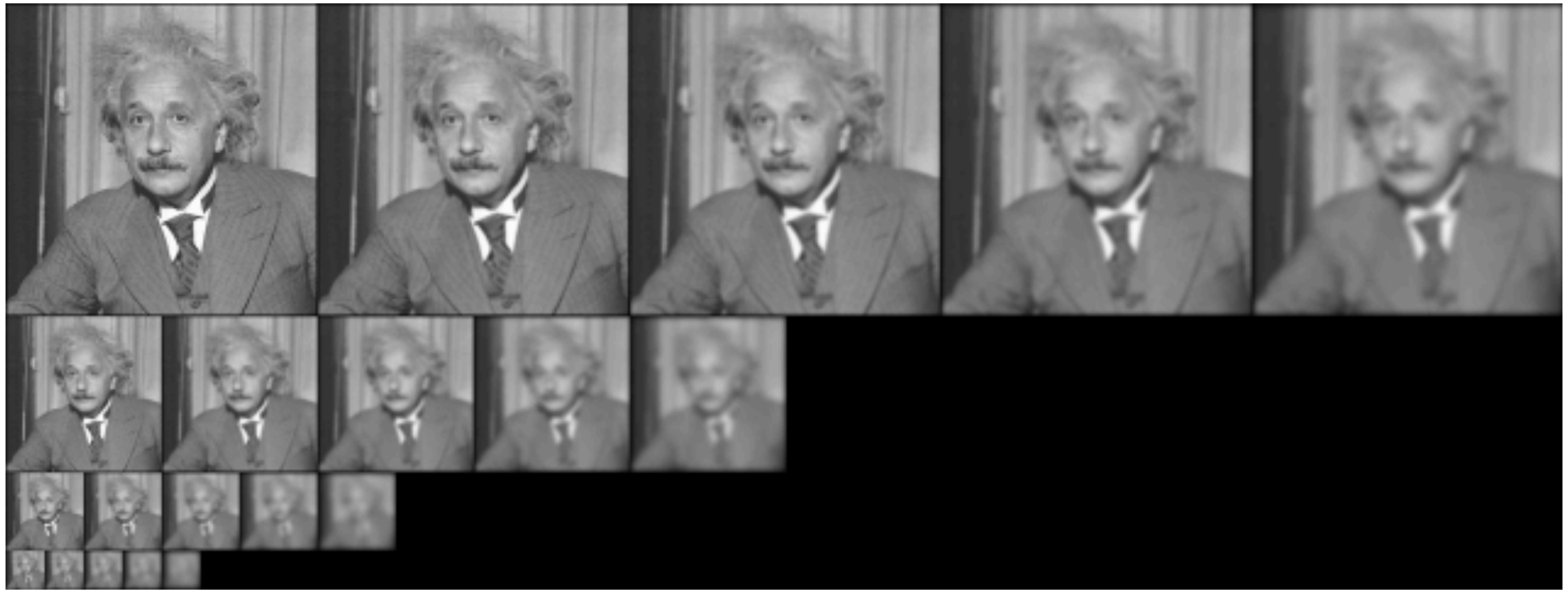
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

1. Multi-scale Extrema Detection



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

1. Multi-scale Extrema Detection



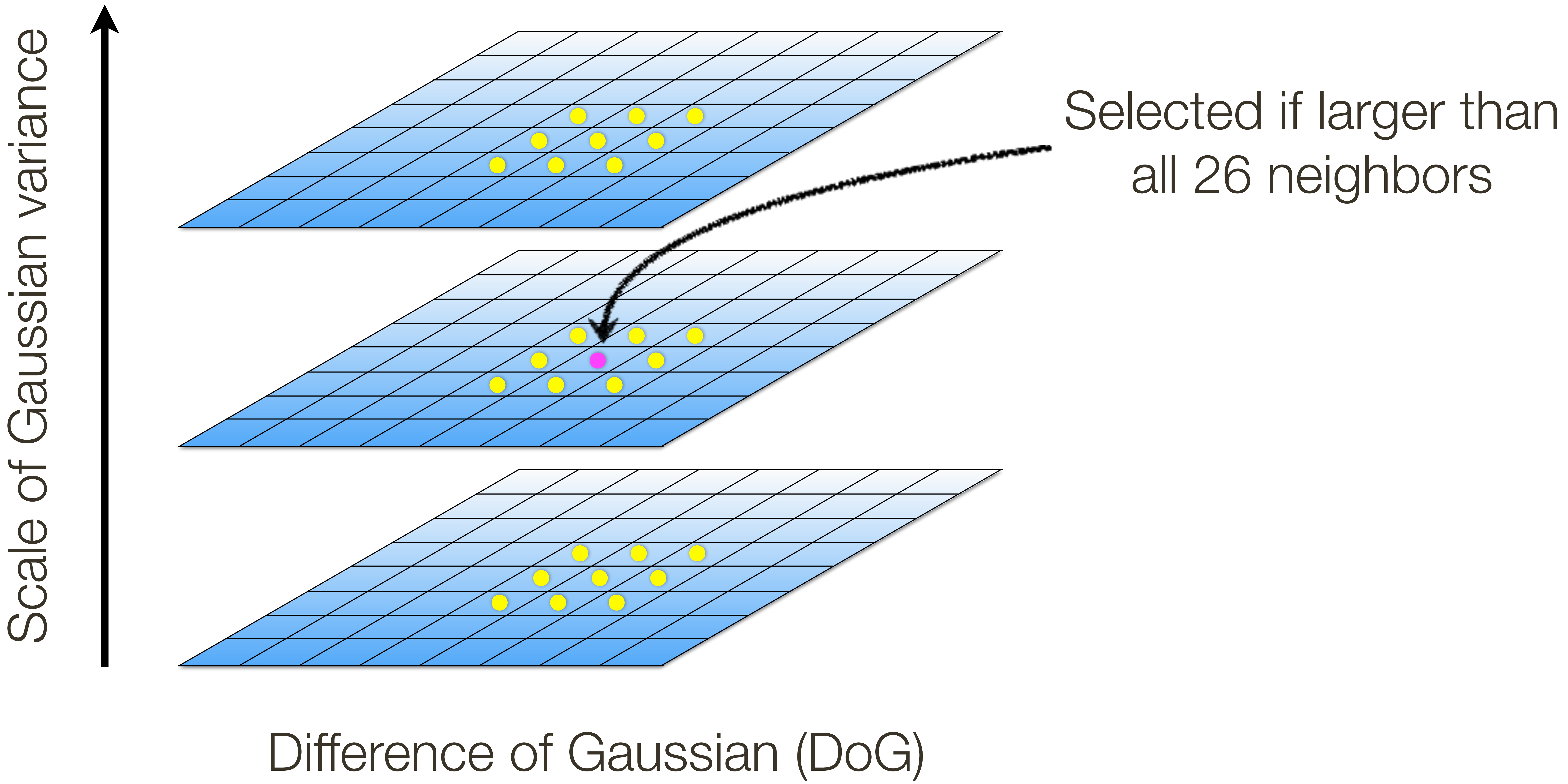
Gaussian

Laplacian

1. Multi-scale Extrema Detection

Detect maxima and minima of Difference of Gaussian in scale space

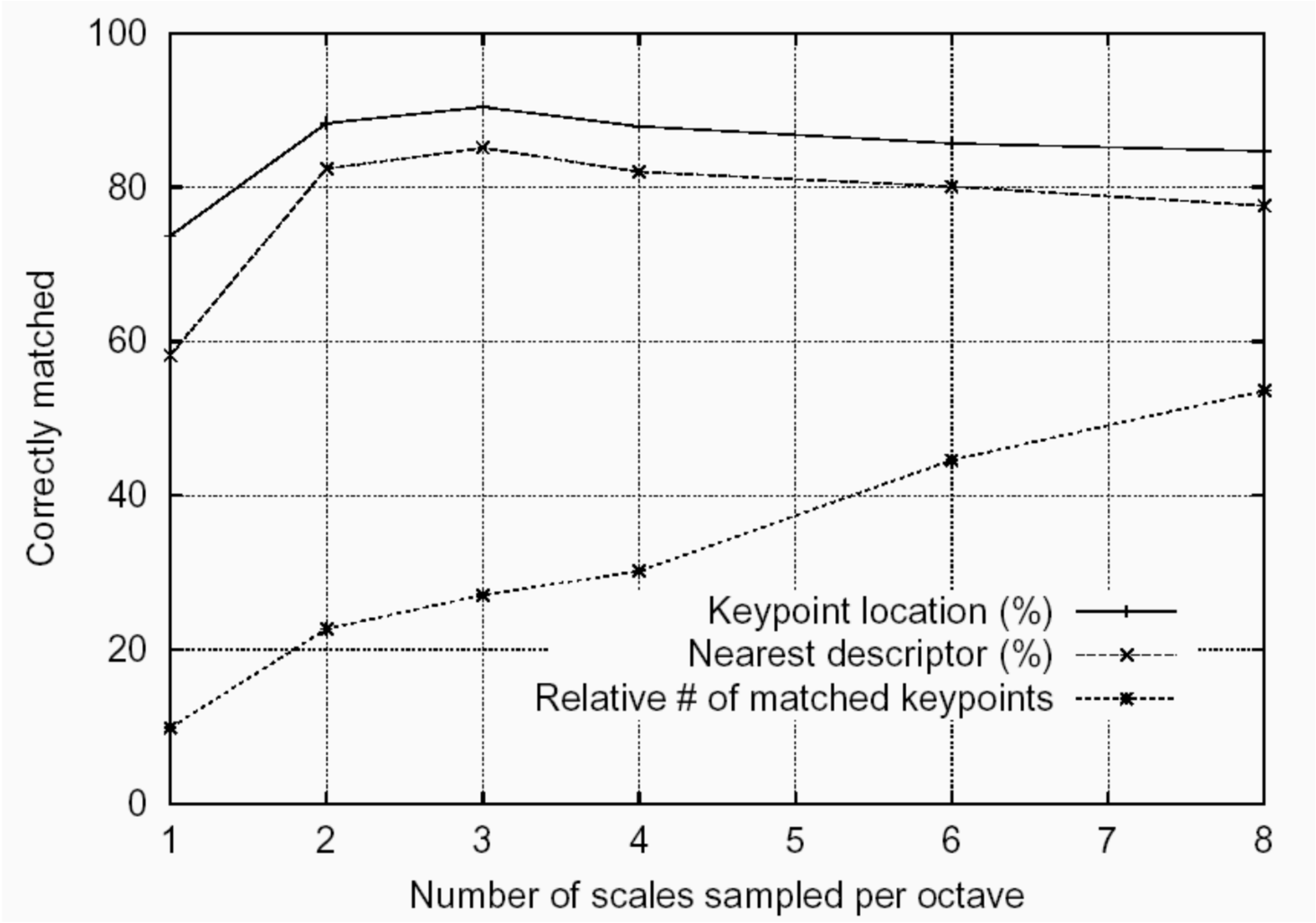
$$\sigma = 2^{1/s}$$



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



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

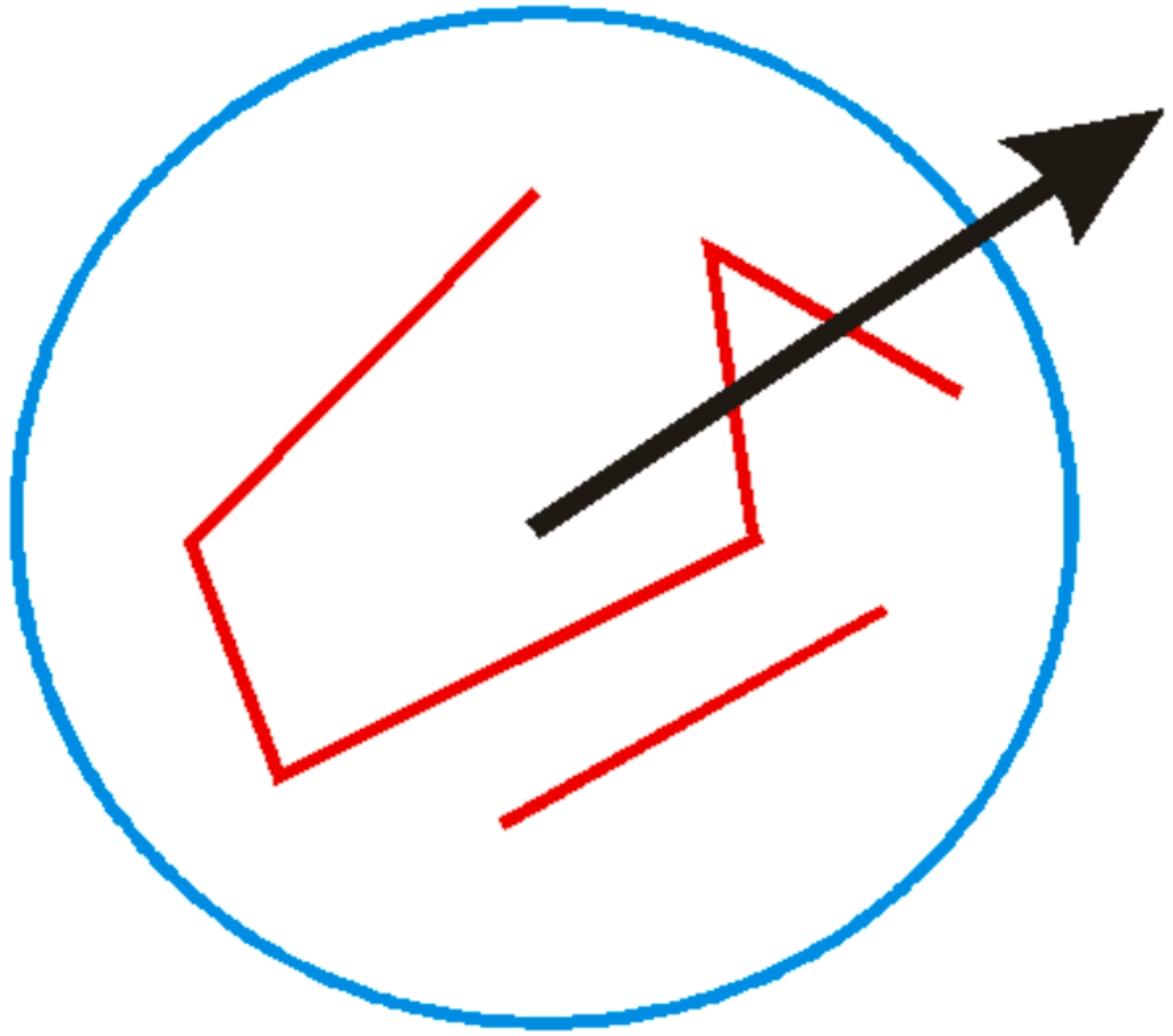
(d) 536 left after testing ratio of principal curvatures

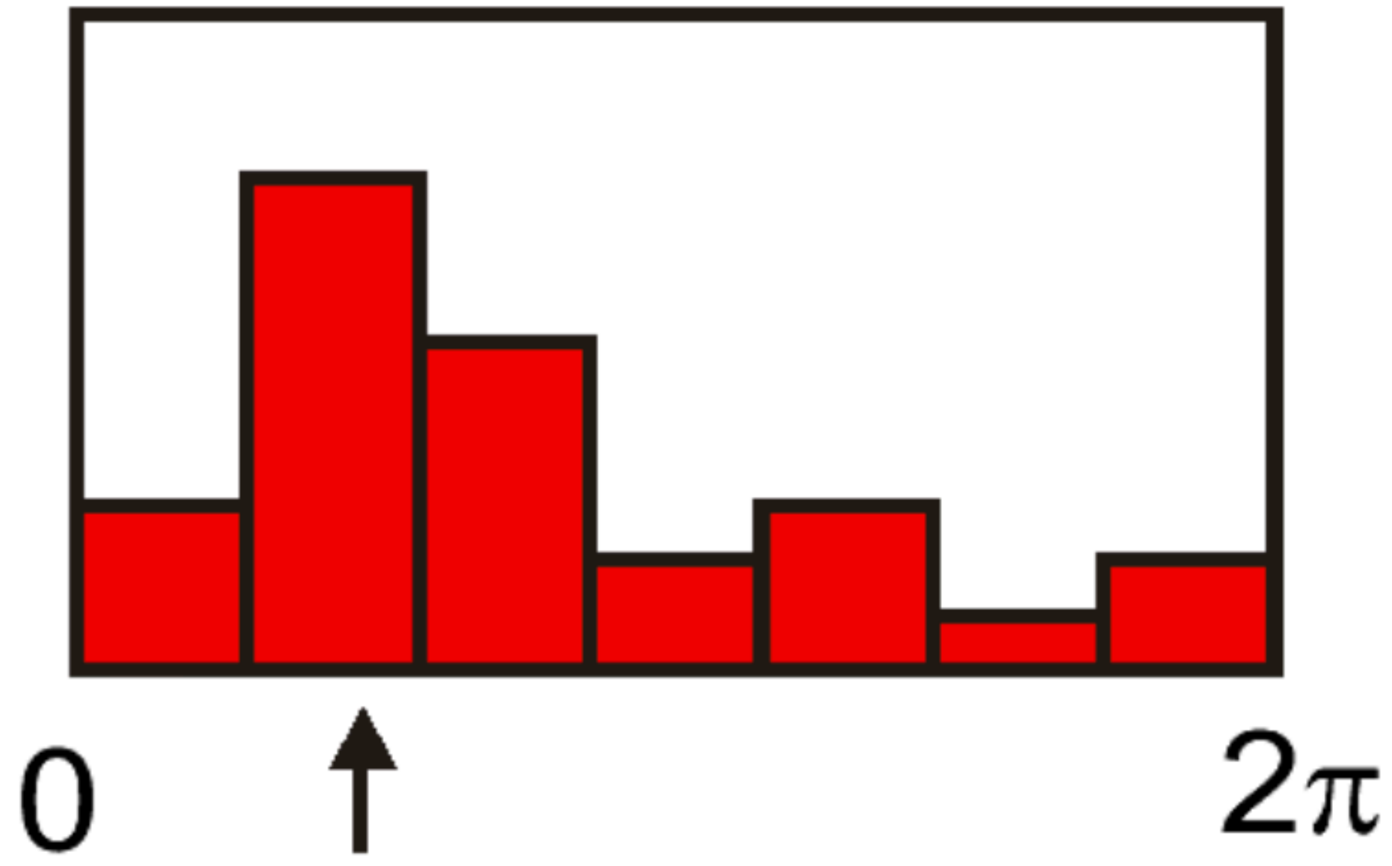
3. Orientation Assignment

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

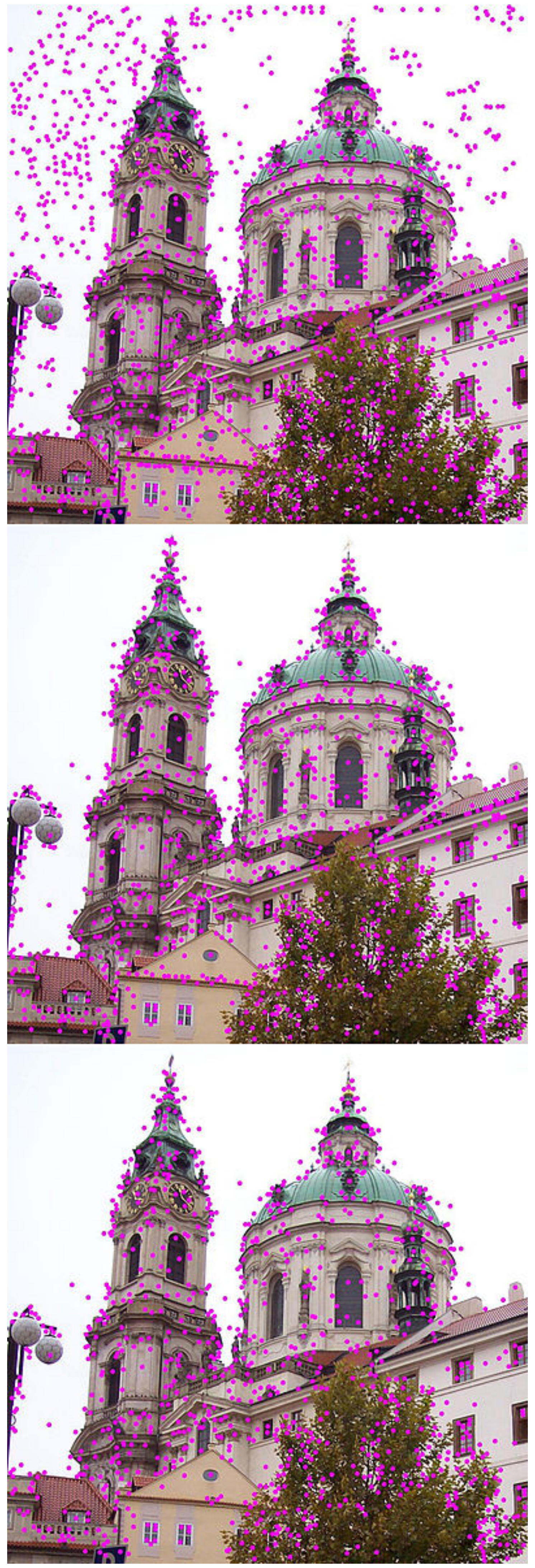
— Assign **canonical orientation** at peak of smoothed histogram

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





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

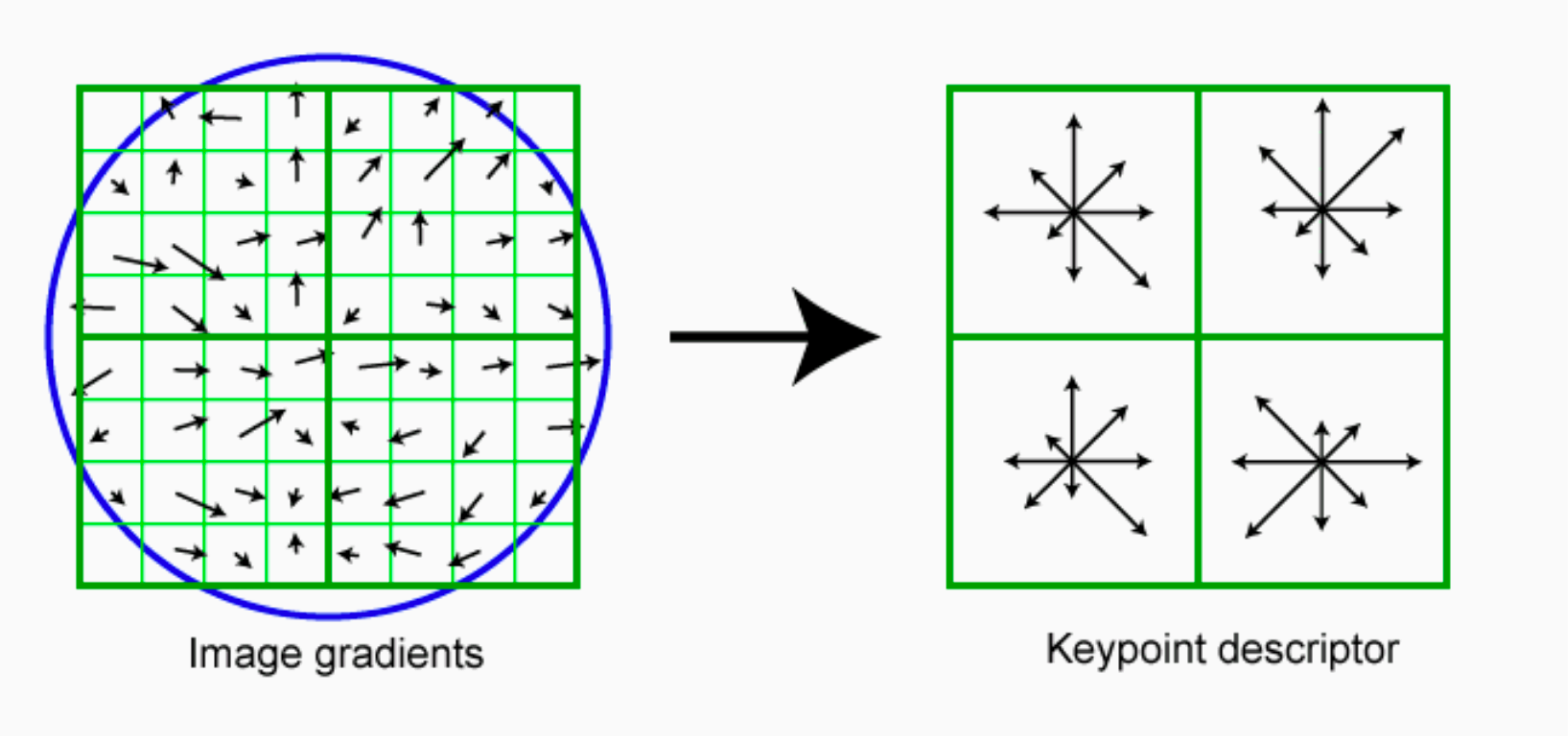
— Keypoint detection is not the same as keypoint description, e.g. some applications skip keypoint detection and extract SIFT descriptors on a regularly spaced grid

4. SIFT Descriptor

— Thresholded 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

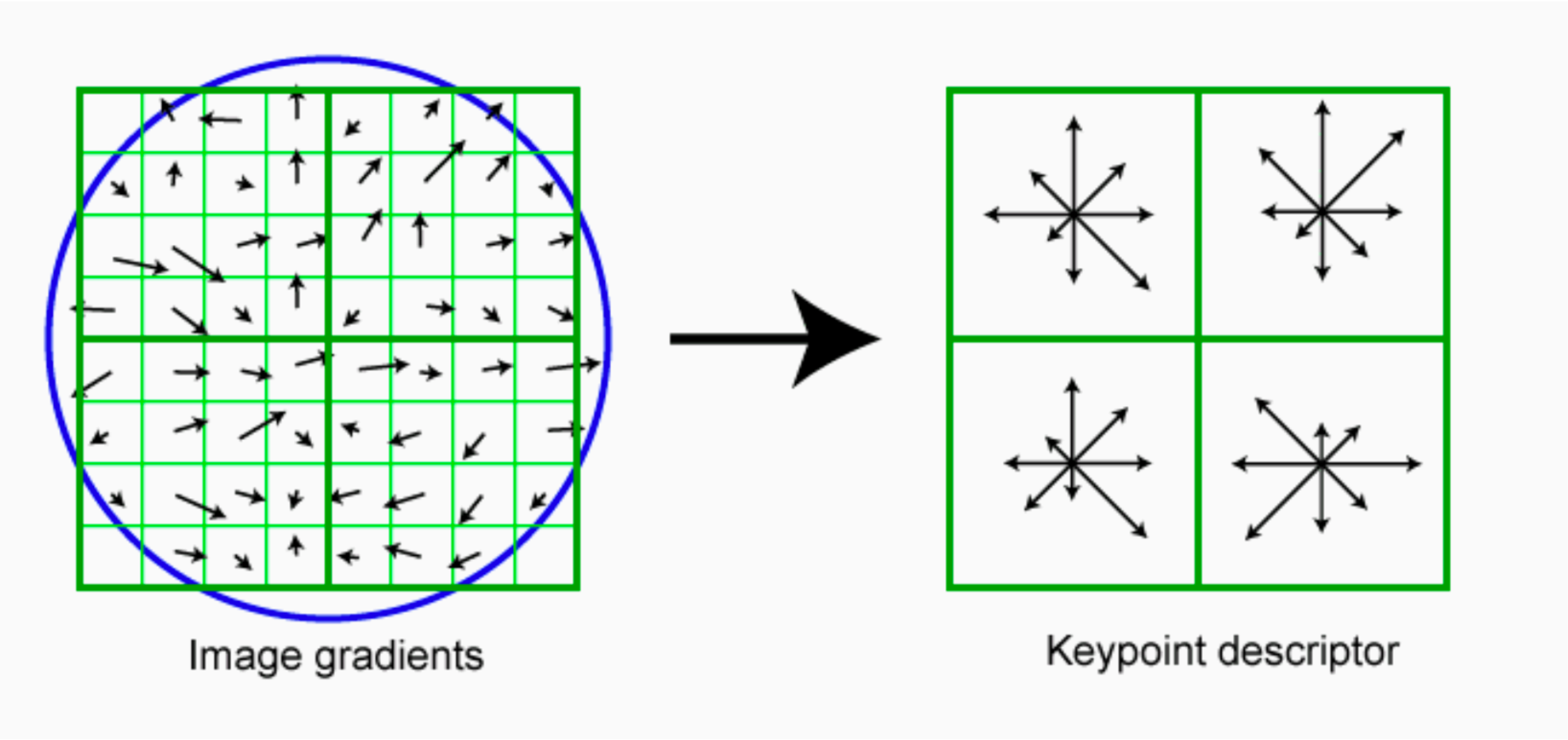
— 8 orientations × 4 × 4 histogram array



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)



Demo

4. SIFT Descriptor

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

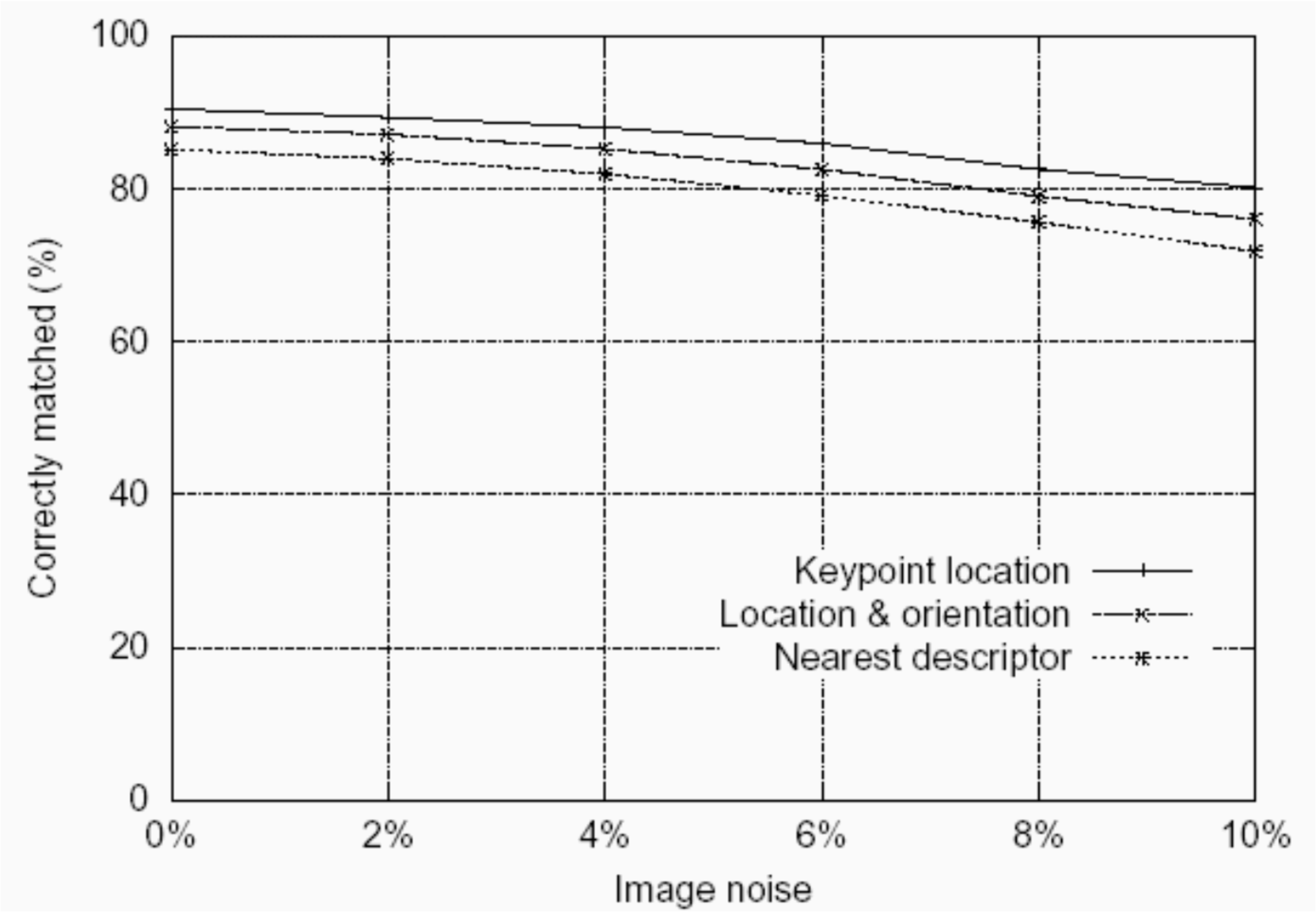
— if brightness values are scaled (multiplied) by a constant, the gradients are scaled by the same constant, and the normalization cancels the change

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

Feature Stability to **Noise**

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

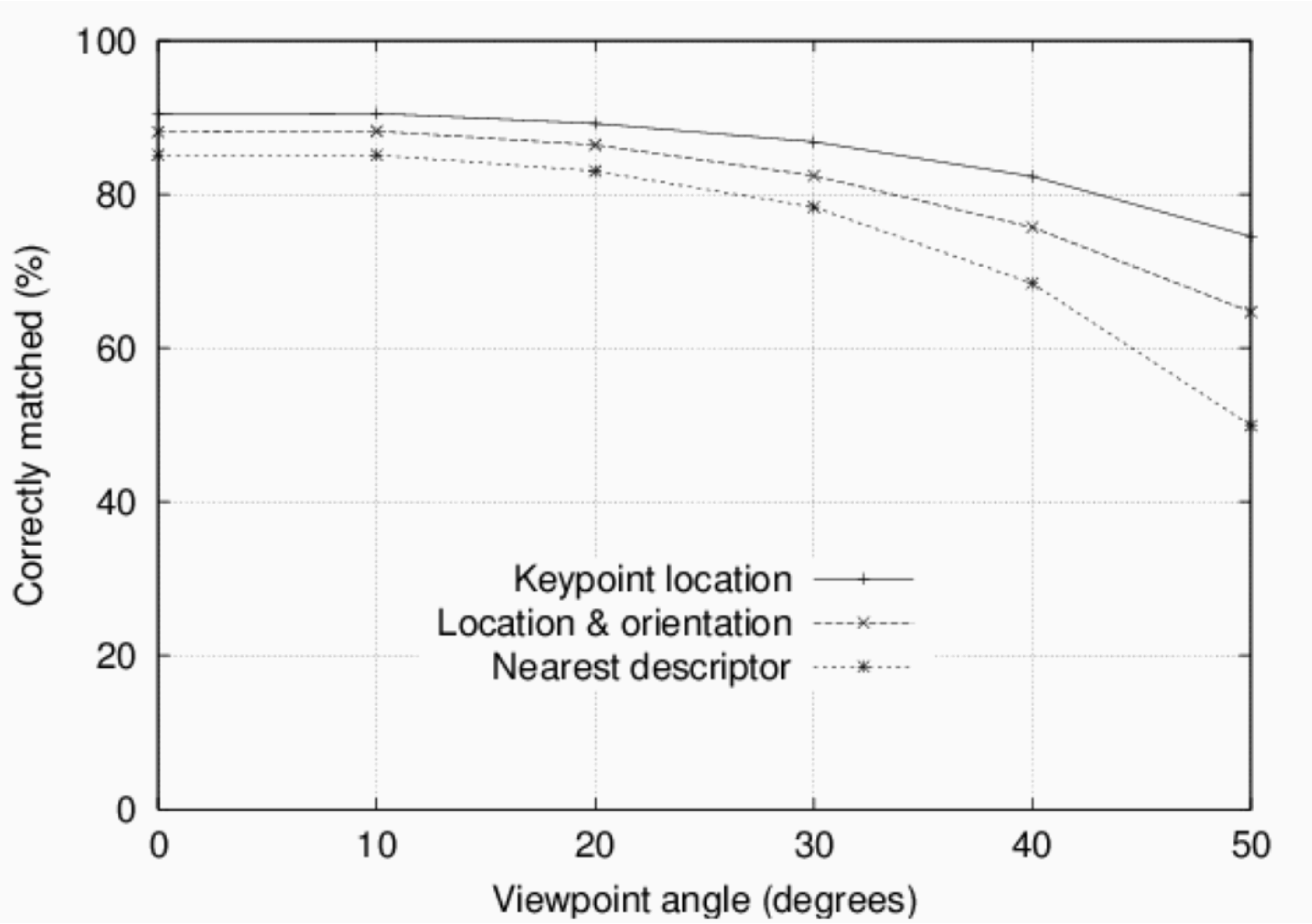
Find nearest neighbour in database of 30,000 features



Feature Stability to **Affine Change**

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

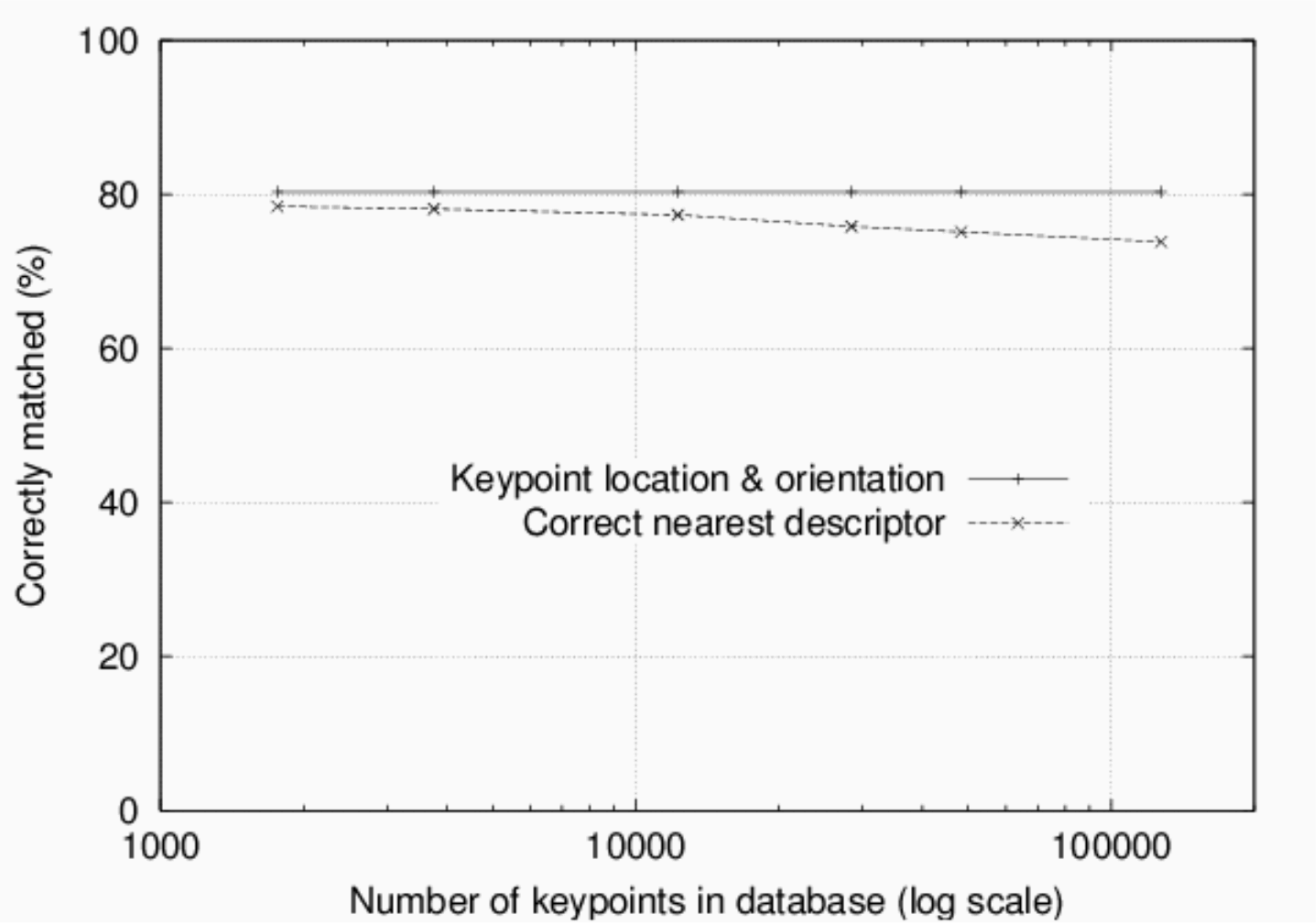
Find nearest neighbour in database of 30,000 features



Distinctiveness of Features

Vary size of database of features, with 30 degree affine change, 2% image noise

Measure % correct for single nearest neighbour match



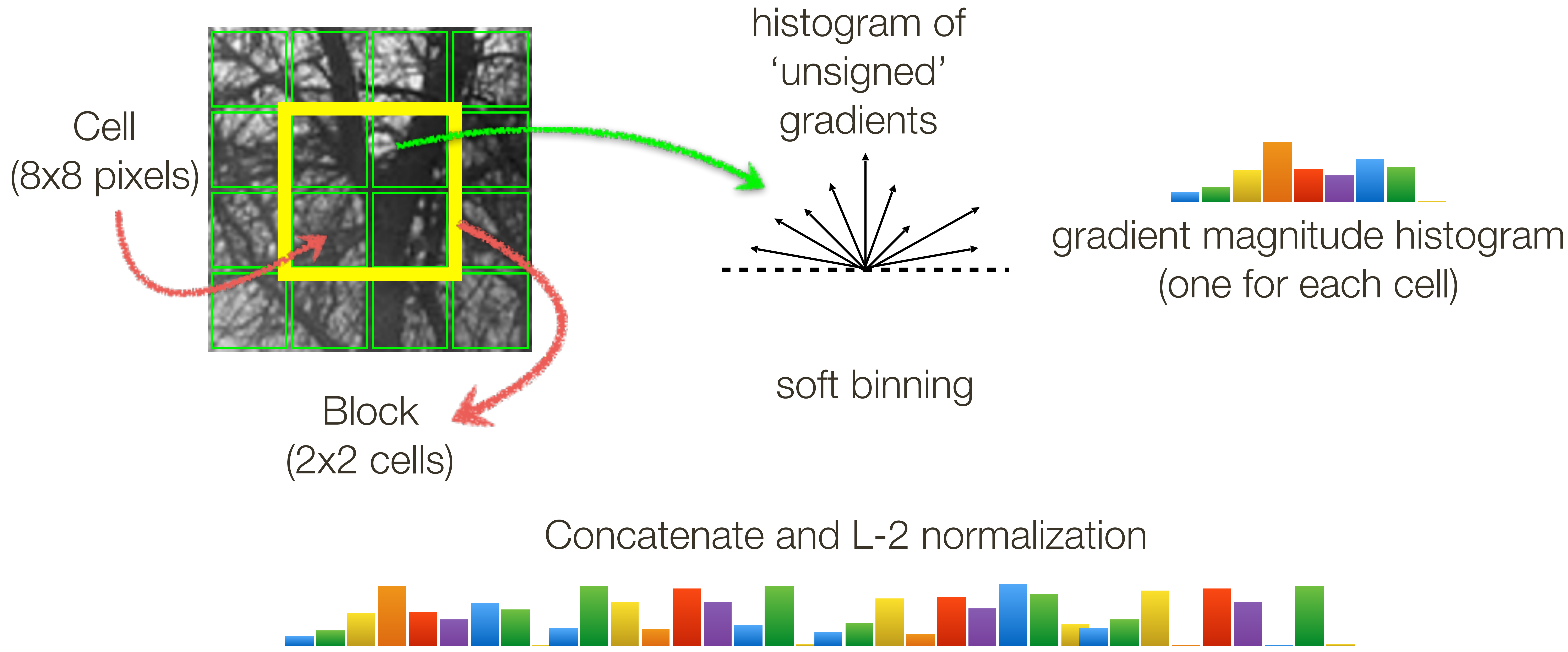
Summary

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



Single scale, no dominant orientation

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

Pedestrian detection

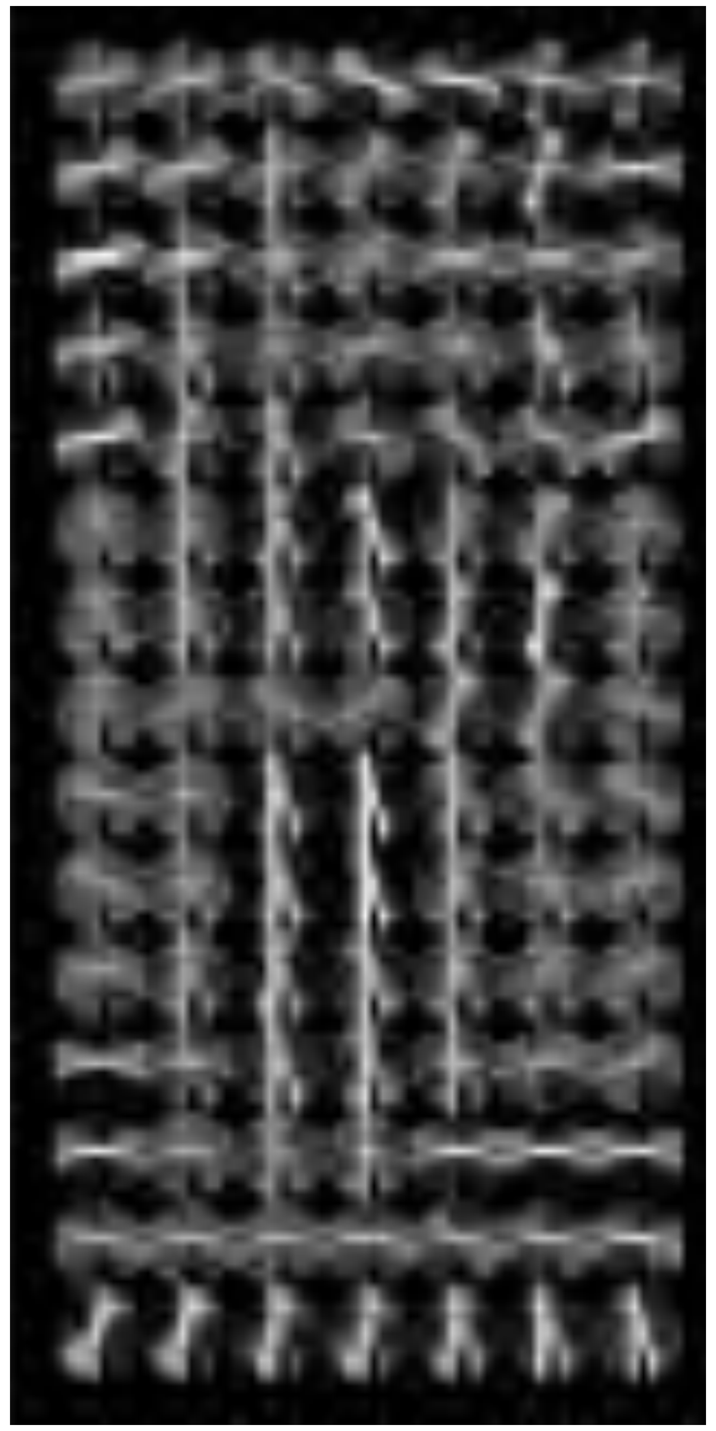
1 cell step size

128 pixels
16 cells
15 blocks

64 pixels
8 cells
7 blocks

15 x 7 x 4 x 36 =
3780

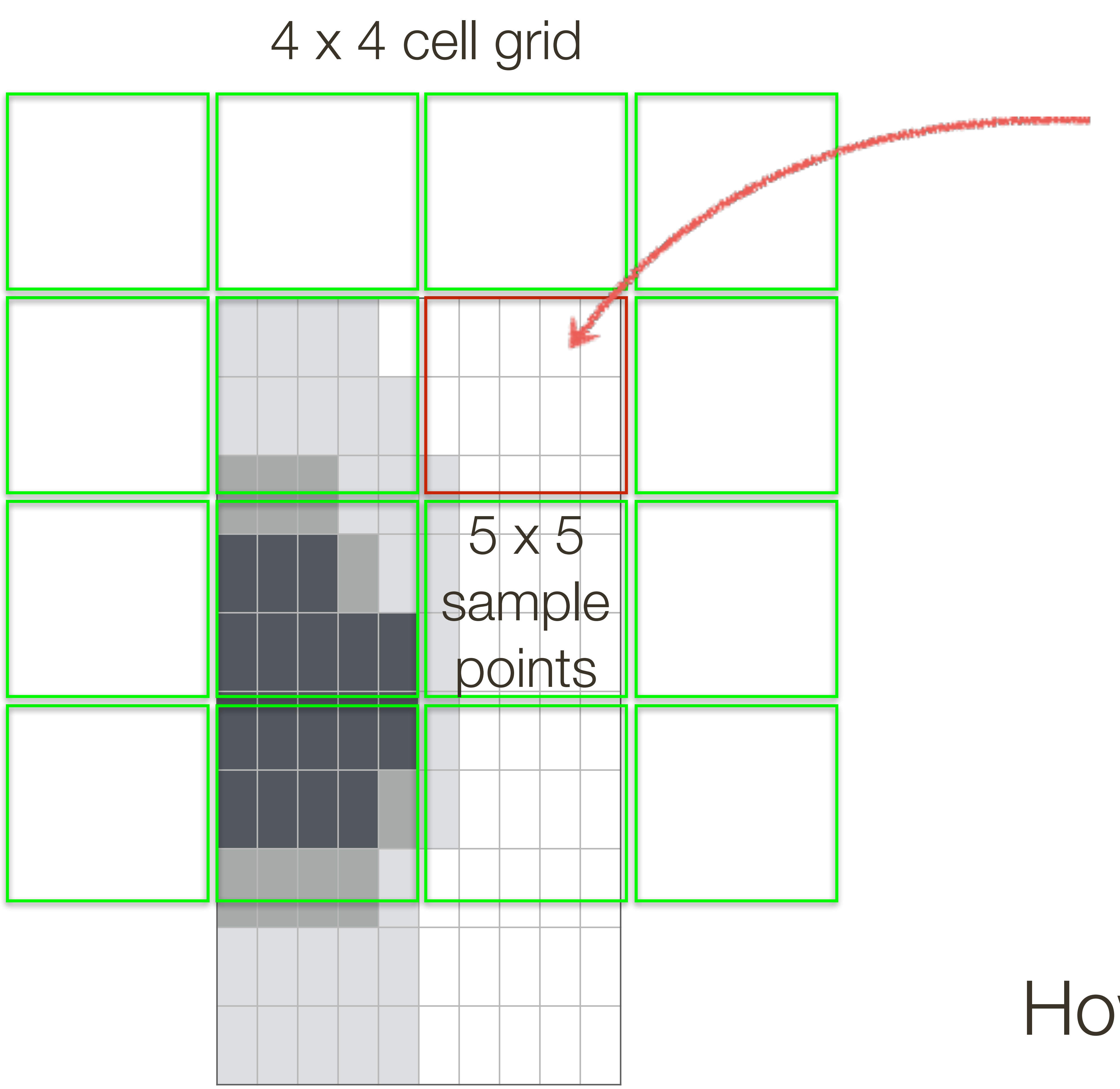
visualization



Redundant representation due to overlapping blocks

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

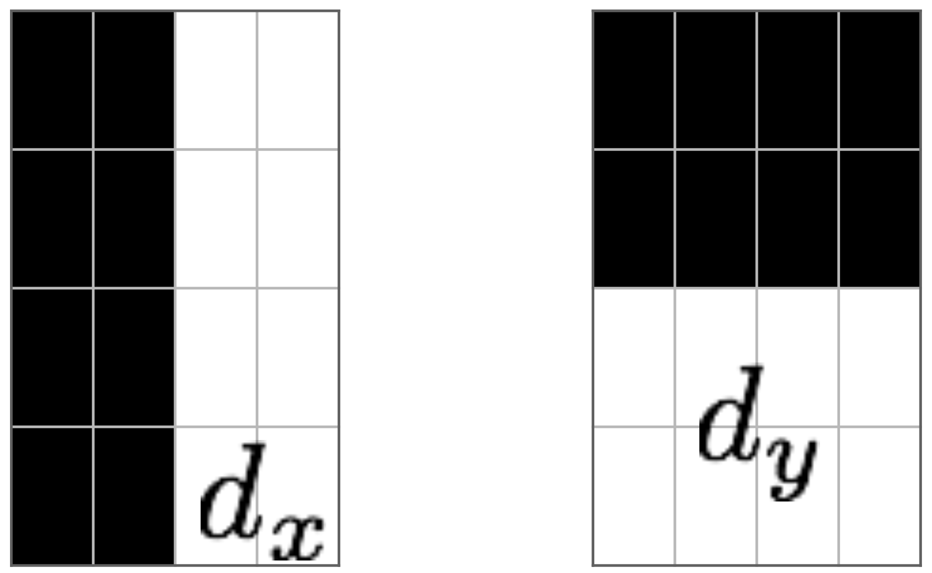
‘Speeded’ Up Robust Features (**SURF**)



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)

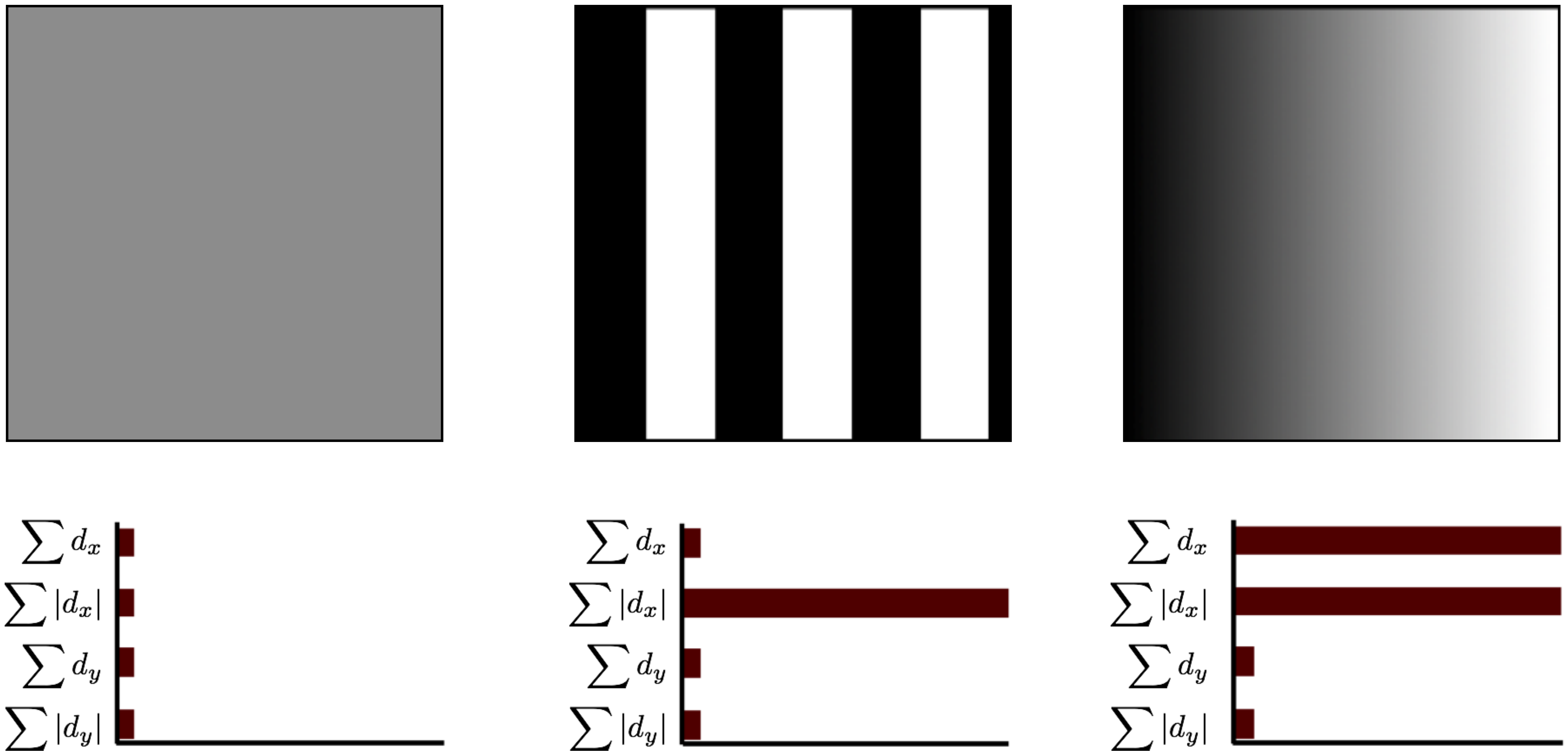


How big is the SURF descriptor?

64 dimensions

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

SIFT and **Object Recognition**

Object recognition requires us to first match each keypoint independently to the database of keypoints

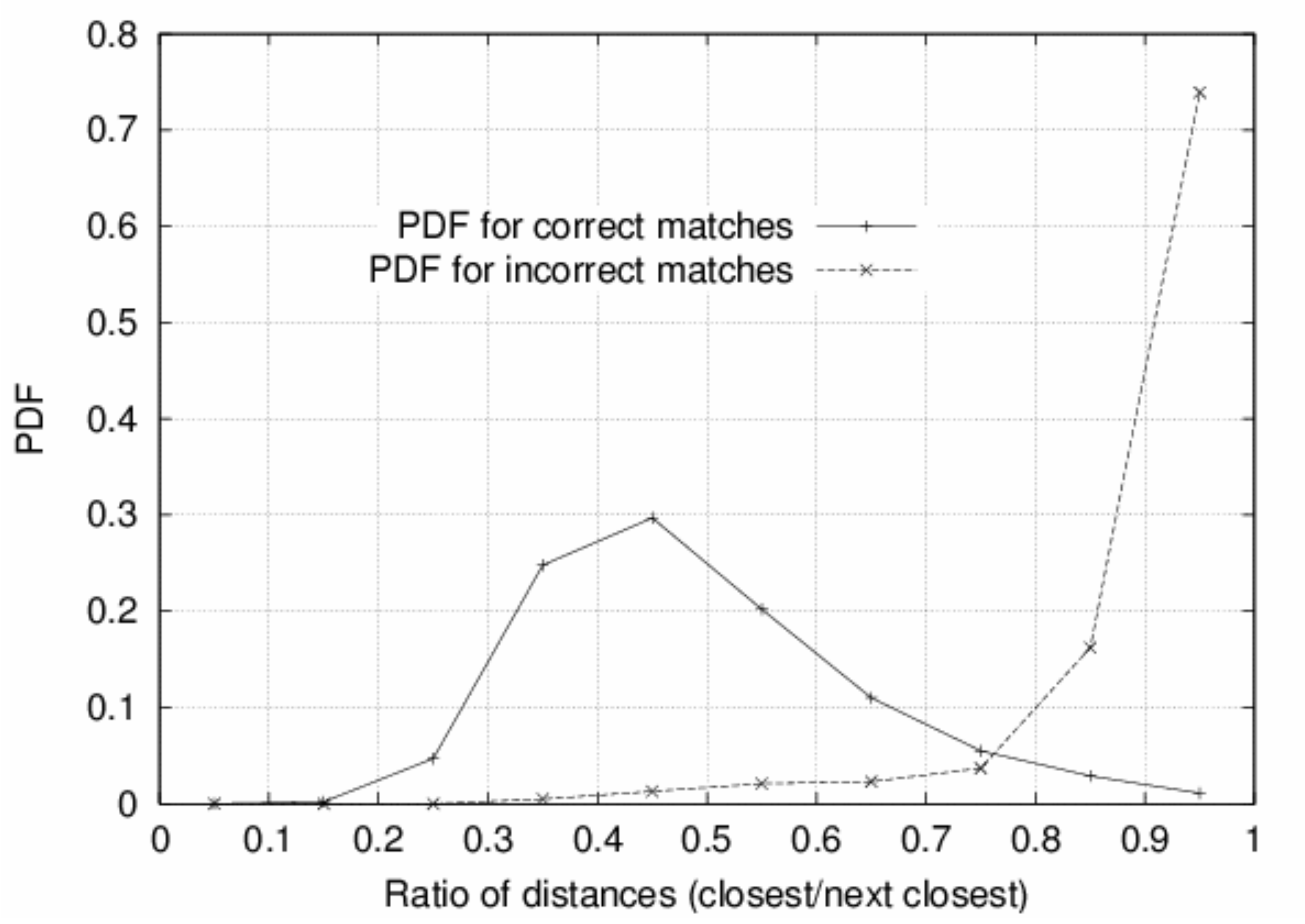
Many features will not have any correct match in the database because they arise from background clutter

It would be useful to have a way to **discard features** that do not have any good match

Probability of **Correct** Match

Compare ratio of distance of **nearest** neighbour to **second** nearest neighbour (from different object)

Threshold of 0.8 provides excellent separation



Nearest-Neighbor Matching to Feature Database

Hypotheses are generated by **approximate nearest neighbour** matching of each feature to vectors in the database

— Use best–bin–first (Beis & Lowe, 97) modification to k-d tree algorithm

— Use heap data structure to identify bins in order by their distance from query point

Result: Can give speedup by factor of 1,000 while finding nearest neighbour (of interest) 95% of the time

Identifying **Consistent** Features

We have matched keypoints to a database of known keypoints extracted from training images

Next we identify **clusters of at least 3 features** that agree on an object and its pose

— a typical image contains 2,000+ features → detecting less than 1% inliers among 99% outliers!

Lowe's solution uses the generalized **Hough transform**

— vote for each potential match according to model ID and pose

— insert into multiple bins to allow for error in similarity approximation

— (more on Hough transforms later)

Model **Verification**

1. Examine all clusters with at least 3 features

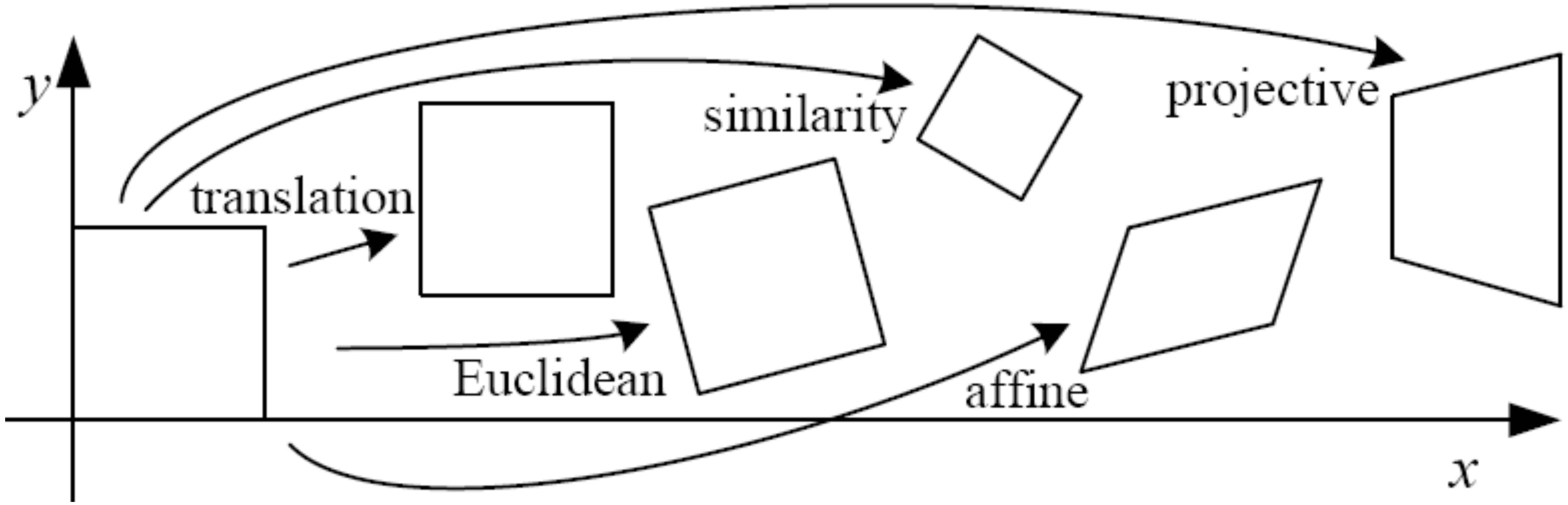
2. Perform least-squares affine **fit to model**

3. **Discard outliers** and perform top-down check for additional features

4. Evaluate probability that match is correct

   — Use Bayesian model, with probability that features would arise by chance if object was not present (Lowe, CVPR 01)

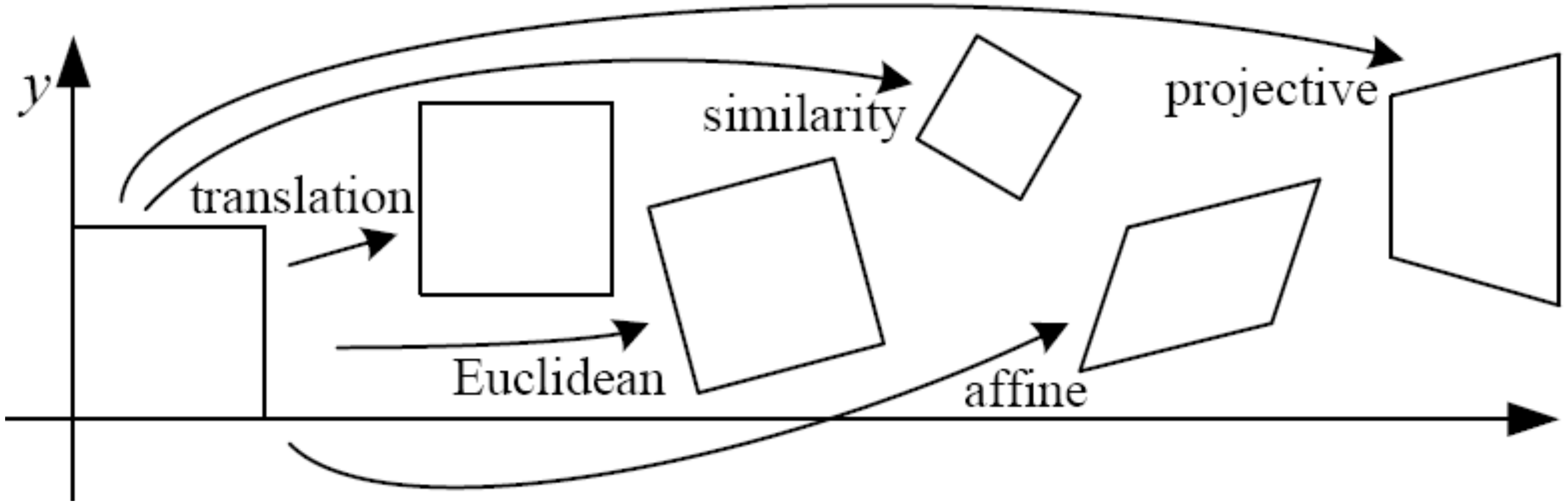
Aside: Classification of 2D Transformations



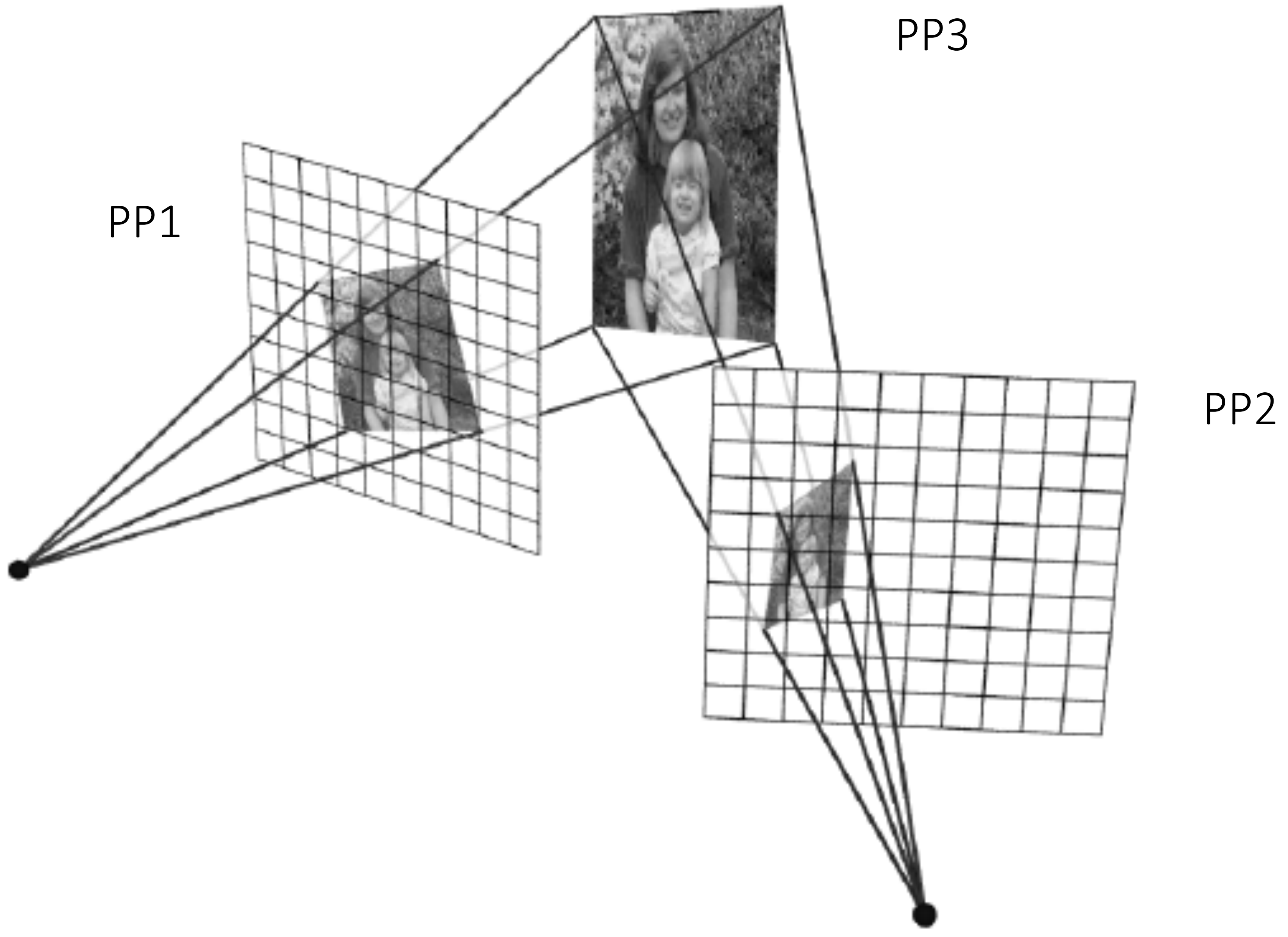
Name	Matrix	# D.O.F.
translation	$\left[\, \boldsymbol{I} \mid \boldsymbol{t} \,\right]_{2\times3}$	2
rigid (Euclidean)	$\left[\, \boldsymbol{R} \mid \boldsymbol{t} \,\right]_{2\times3}$	3
similarity	$\left[\, s\boldsymbol{R} \mid \boldsymbol{t} \,\right]_{2\times3}$	4
affine	$\left[\, \boldsymbol{A} \,\right]_{2\times3}$	6
projective	$\left[\, \tilde{\boldsymbol{H}} \,\right]_{3\times3}$	8

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Aside: Classification of 2D Transformations

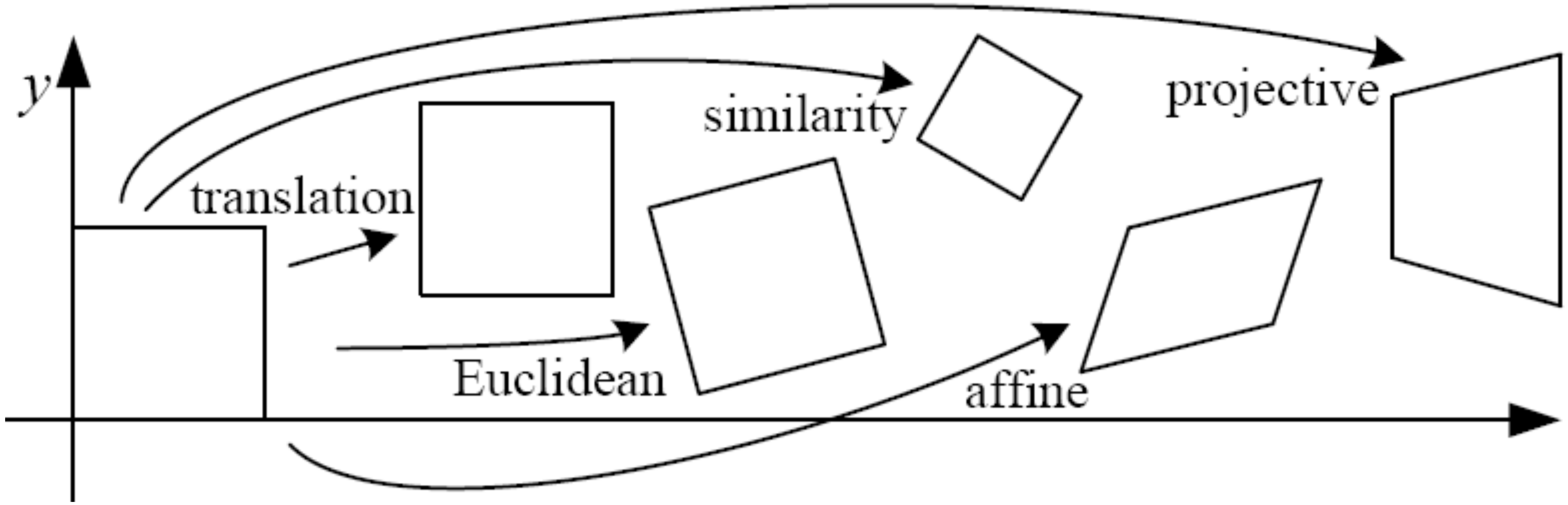


Which kind **transformation** is needed to warp projective plane 1 into projective plane 2?



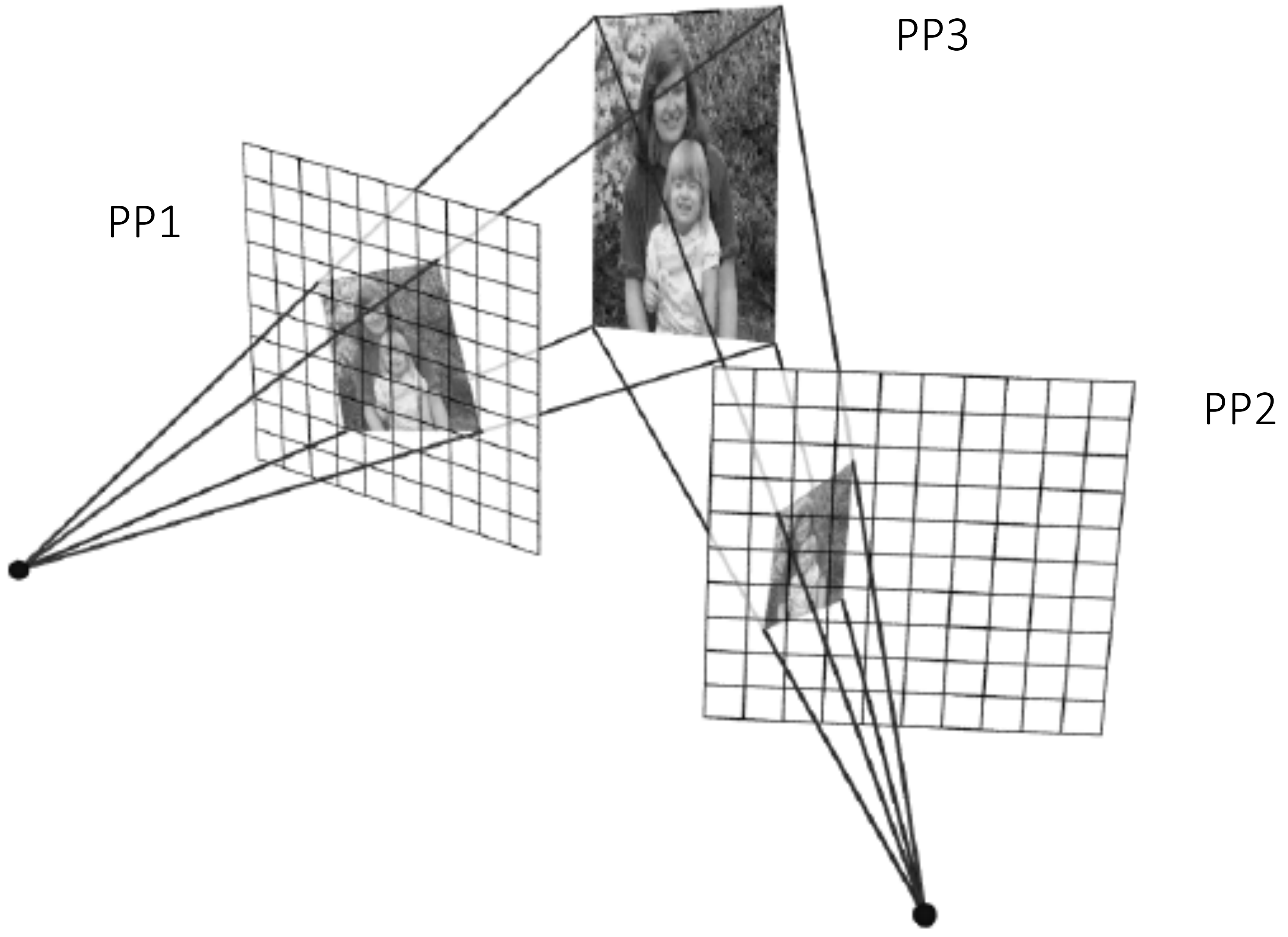
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Aside: Classification of 2D Transformations



Which kind **transformation** is needed to warp projective plane 1 into projective plane 2?

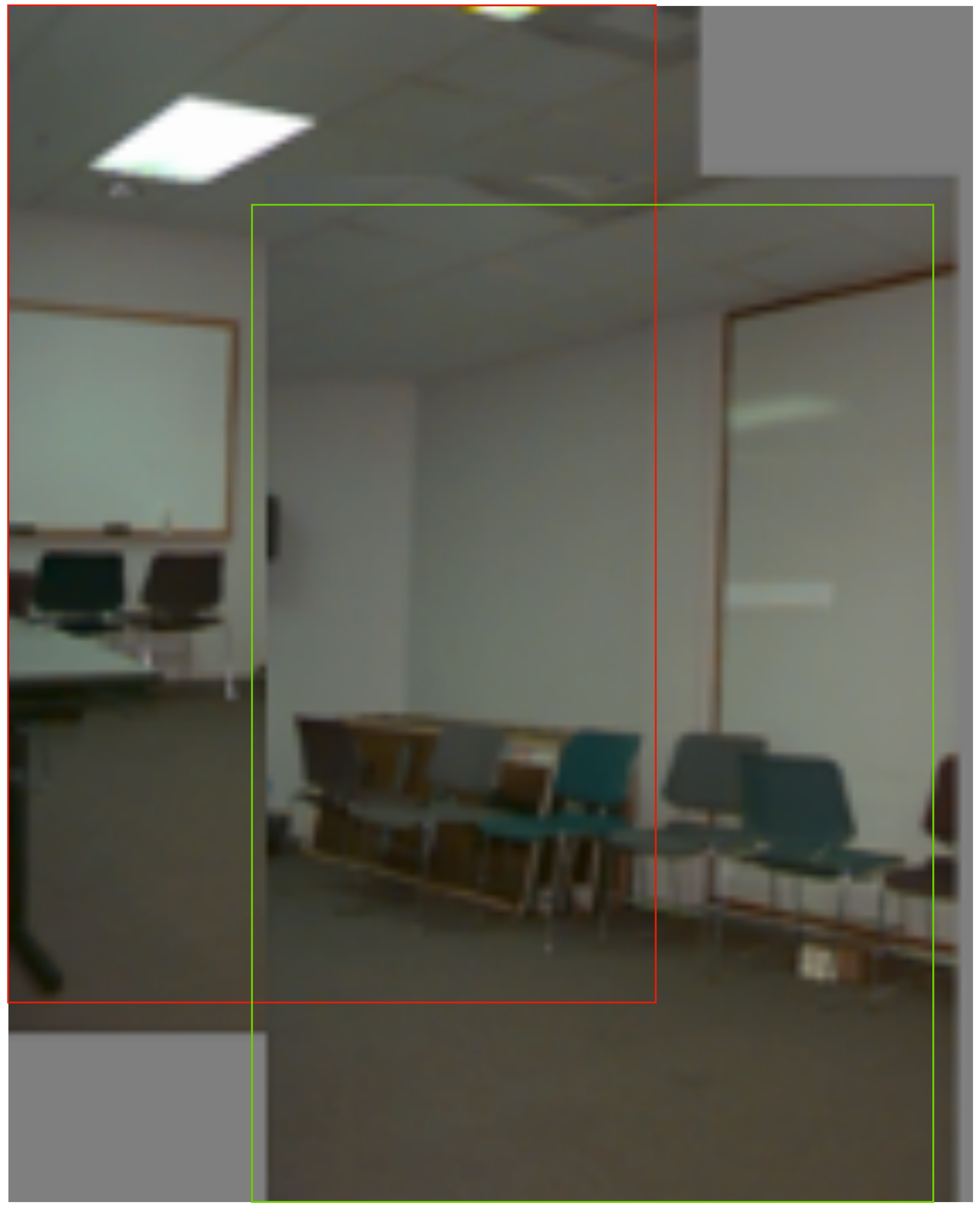
— A **projective** transformation (a.k.a. a homography).



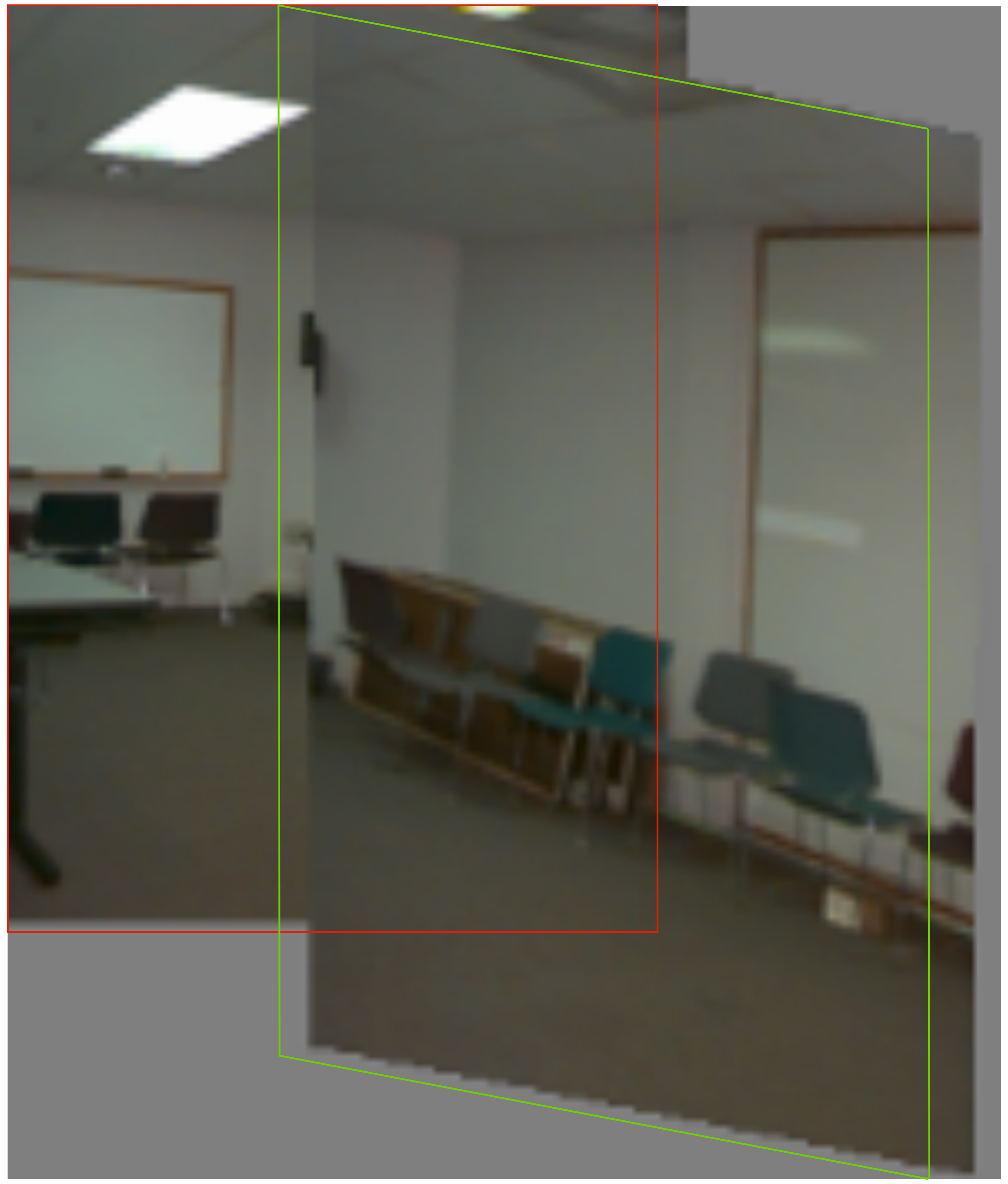
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Aside: Warping with Different Transformations

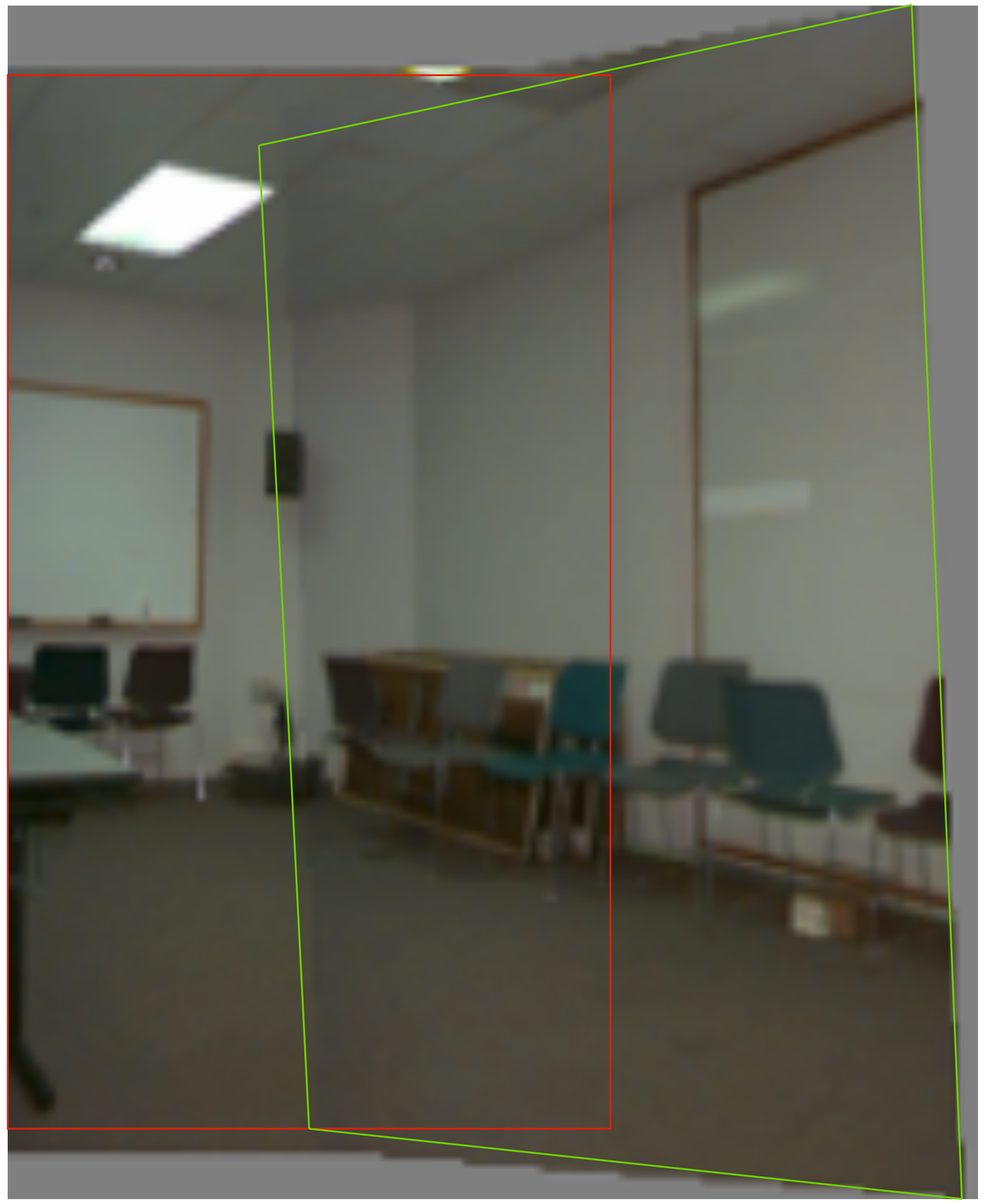
Translation



Affine



Projective (homography)



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Aside: We can use homographies when …

1.… the scene is planar; or

2.… the scene is very far or has small (relative) depth variation → scene is approximately planar

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Aside: We can use homographies when …

3.… the scene is captured under camera rotation only (no translation or pose change)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Solution for **Affine** Parameters

Affine transform of $[x, y]$ to $[u, v]$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Rewrite to solve for **transformation** parameters:

$$\begin{bmatrix} x_1 & y_1 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1 & y_1 & 0 & 1 \\ x_2 & y_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_2 & y_2 & 0 & 1 \\ & & \dots & \dots & & \\ & & \dots & \dots & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \dots \\ \dots \end{bmatrix}$$

(6 equations 6 unknowns)

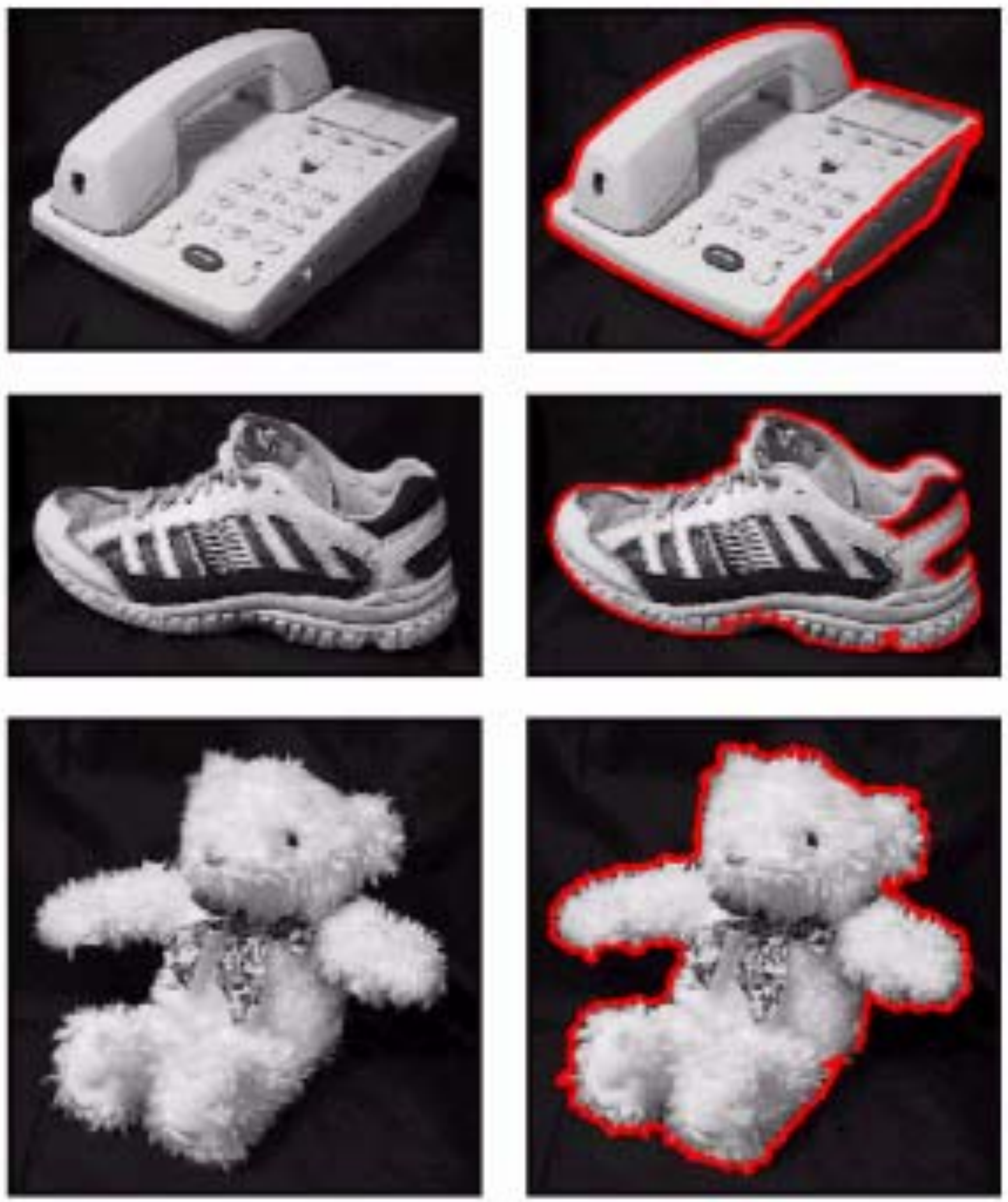
Solution for **Affine** Parameters

Suppose we have $k \geq 3$ matches, $[x_i, y_i]$ to $[u_i, v_i]$, $i = 1, 2, \cdots, k$

Then,

$$\begin{bmatrix} x_1 & y_1 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1 & y_1 & 0 & 1 \\ x_2 & y_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_2 & y_2 & 0 & 1 \\ & & \cdots & \cdots & & \\ & & \cdots & \cdots & & \\ x_k & y_k & 0 & 0 & 1 & 0 \\ 0 & 0 & x_k & y_k & 0 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \cdots \\ \cdots \\ u_k \\ v_k \end{bmatrix}$$

3D **Object Recognition**

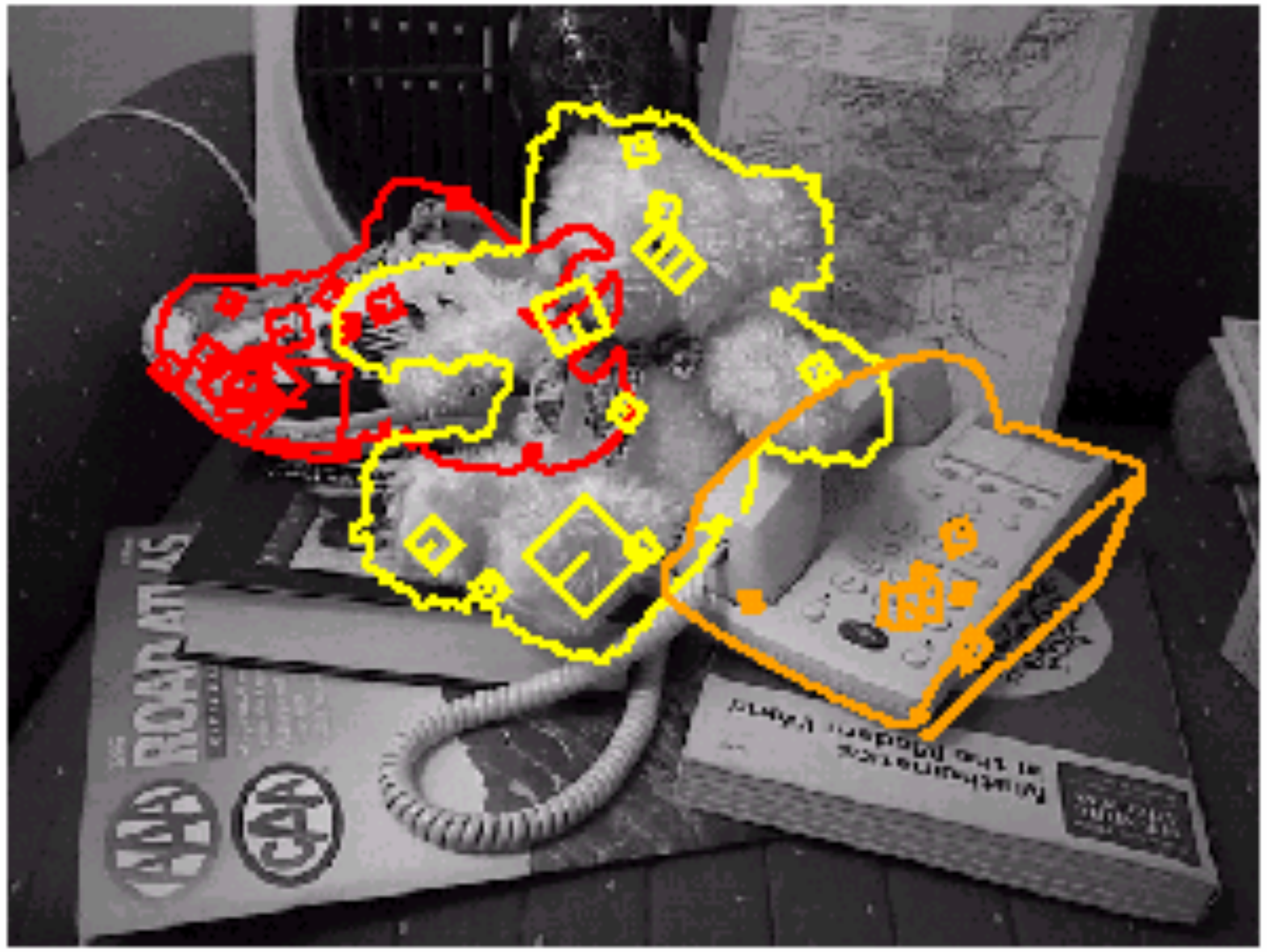


Extract outlines with background subtraction

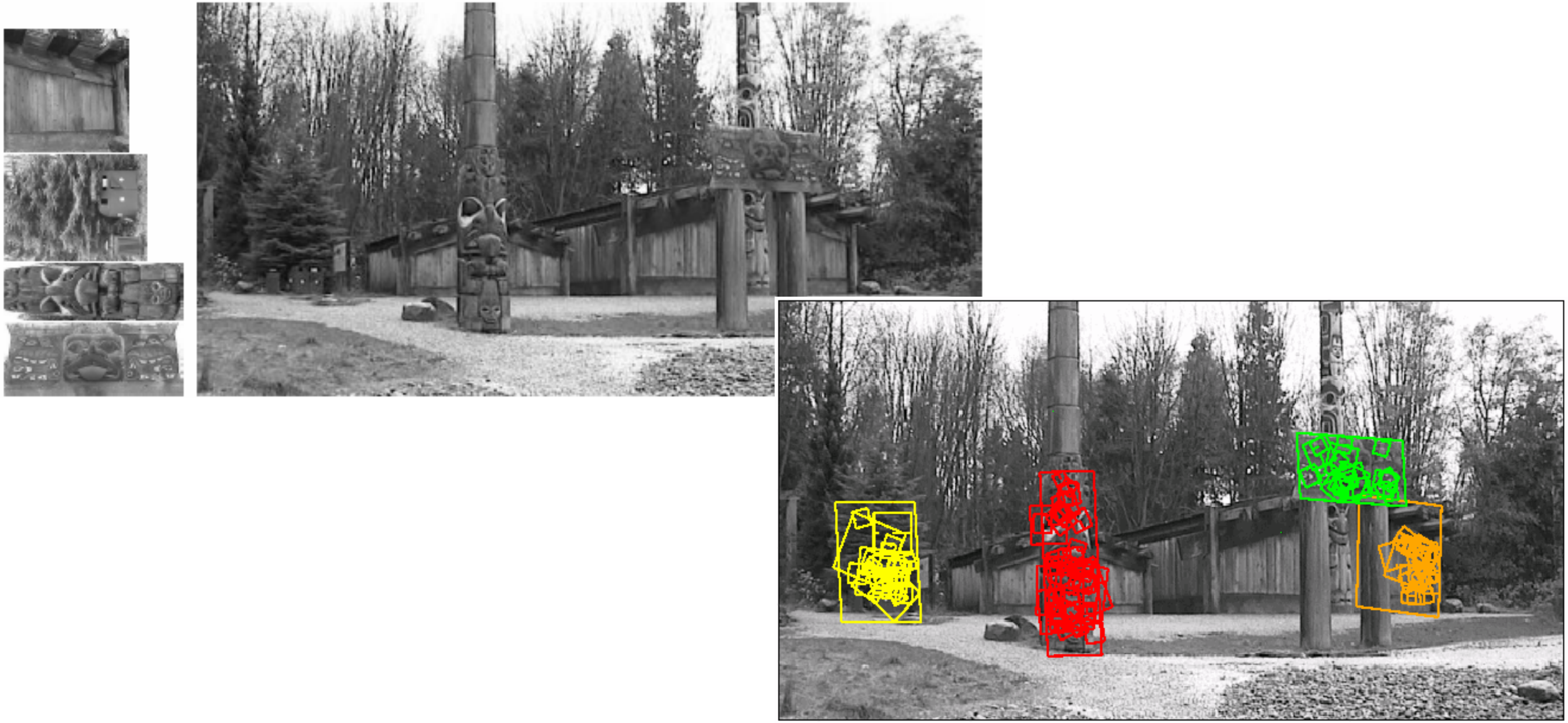
3D **Object Recognition**

Only 3 keys are needed for recognition, so extra keys provide robustness

Recognition Under **Occlusion**



Location Recognition



Example 1: Sony Aibo

SIFT Usage

— Recognize charging station

— Communicate with visual cards

Summary of Object Recognition with SIFT

Match each keypoint independently to database of known keypoints extracted from “training” examples

— use fast (approximate) nearest neighbour matching

— threshold based on ratio of distances to best and to second best match

Identify **clusters of (at least) 3 matches** that agree on an object and a similarity pose

— use generalized Hough transform

Check each cluster found by performing detailed geometric fit of affine transformation to the model

— accept/reject interpretation accordingly