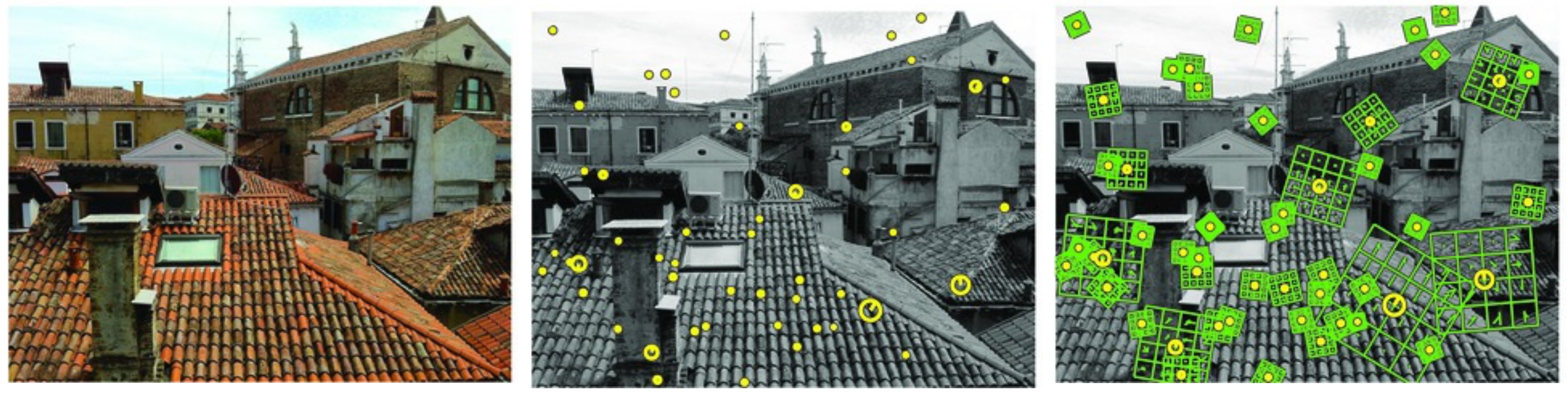
THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision



Lecture 19: Scale Invariant Features (SIFT) cont.

Menu for Today (**October 26, 2020**)

Topics:

— Scale Invariant Feature Transform (SIFT)

— SIFT detector, descriptor

Readings:

— **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

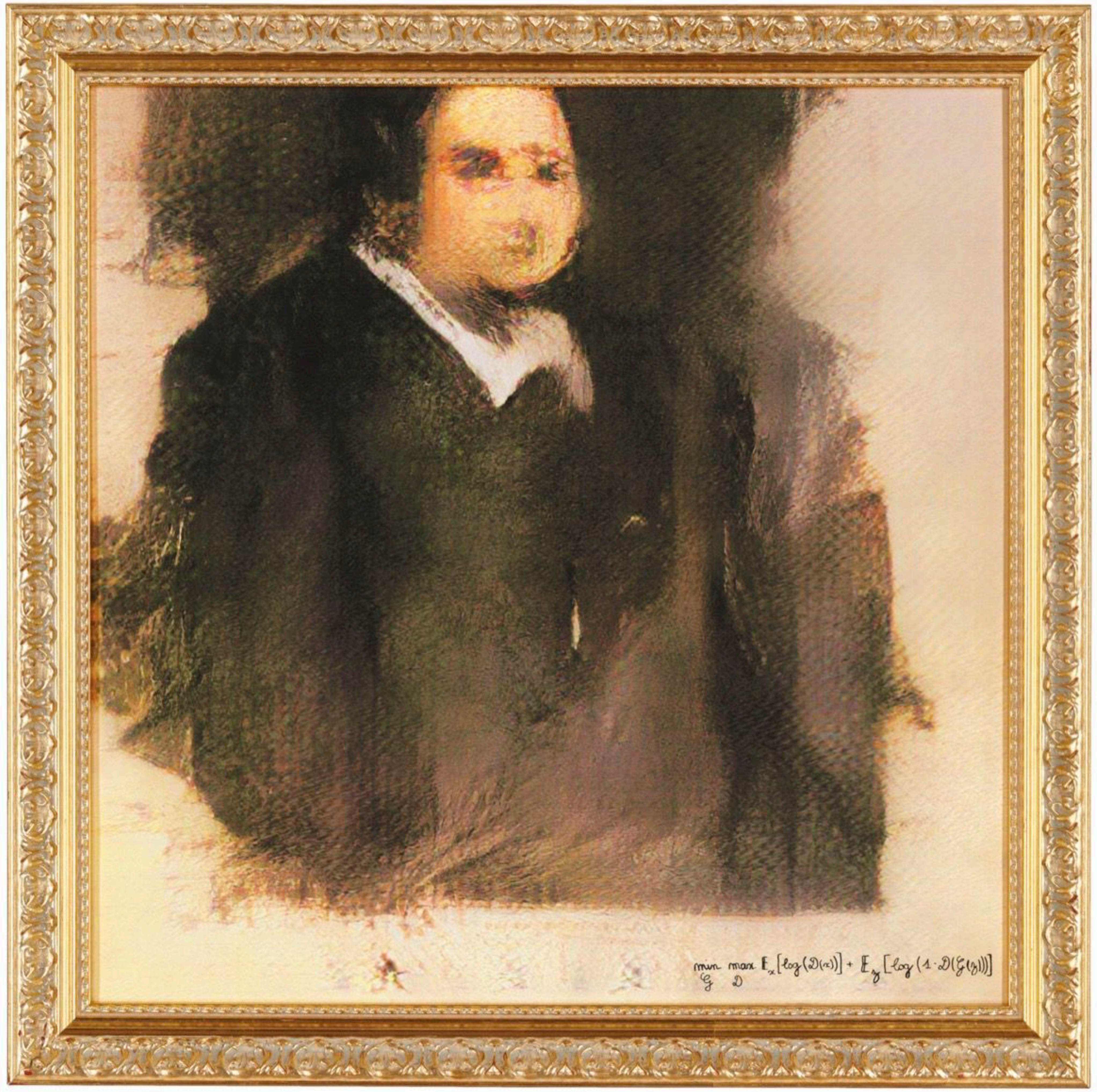
Reminders:

— **Assignment 3**: Texture Synthesis is due **today**

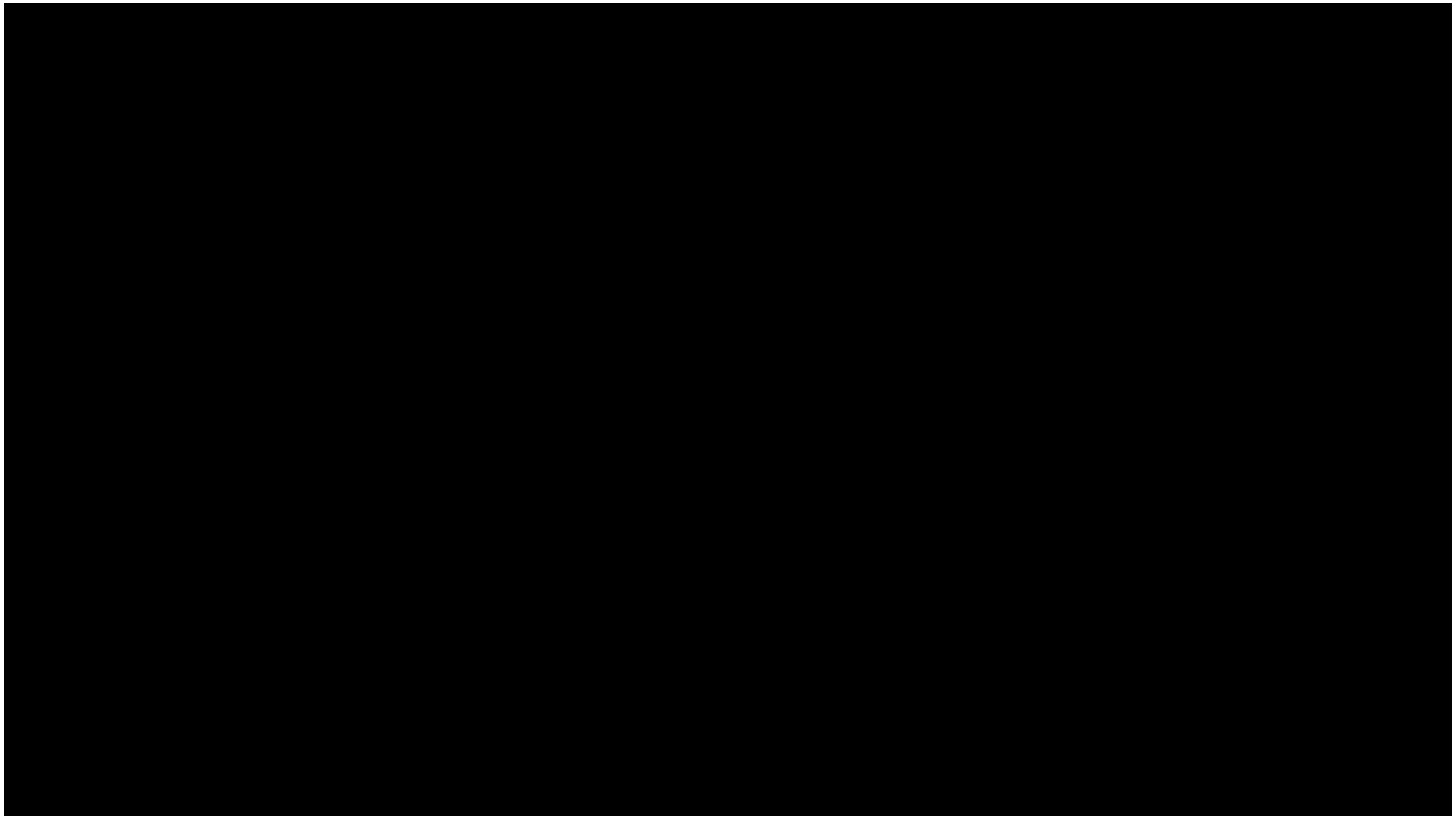
— **Assignment 4** is **out** now

Today's "**fun**" Example: AI Generated Portrait

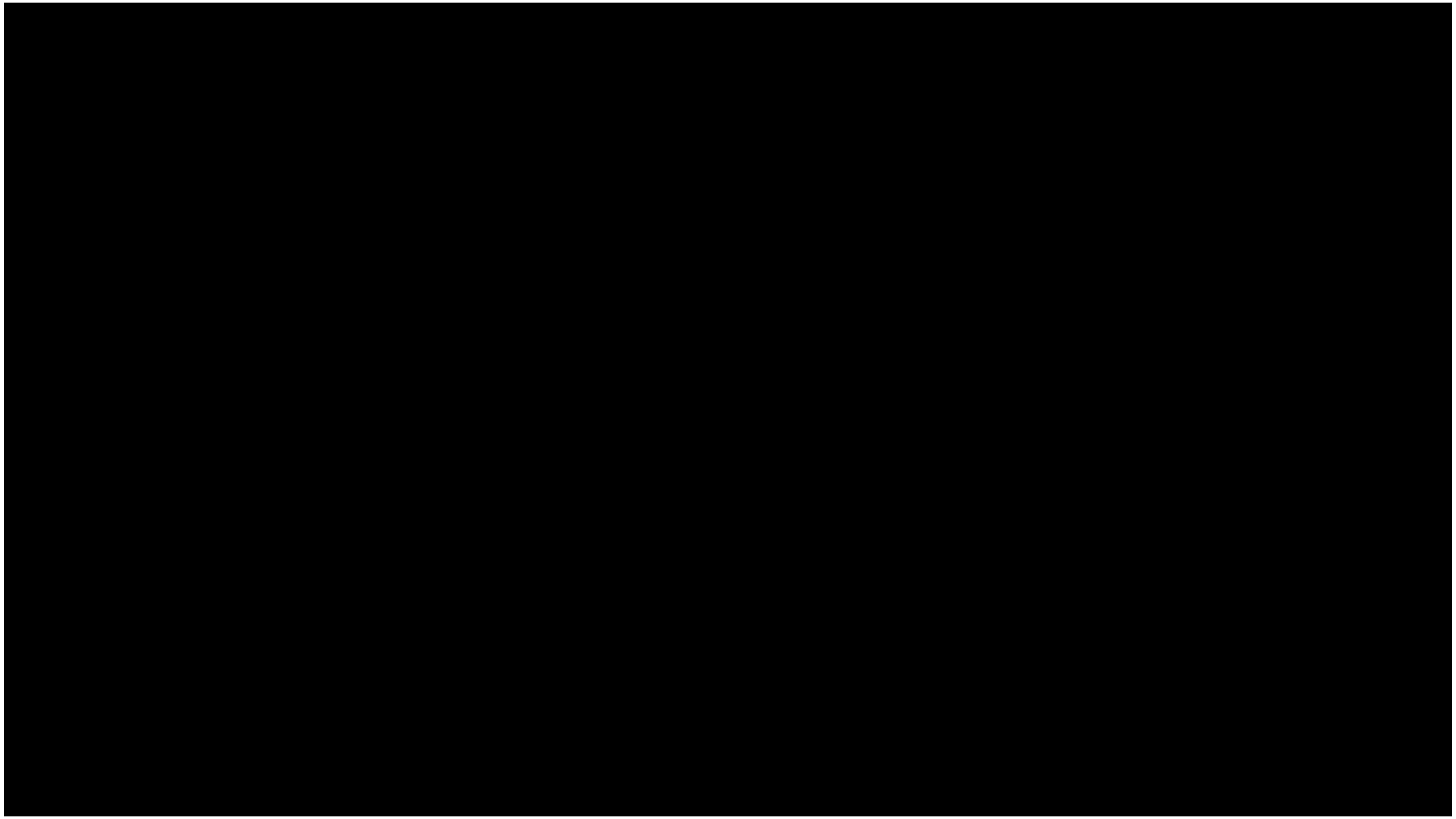
Sold in 2018 for $432,500 at British auction house



Today's "**fun**" Example: Sunspring



Today's "**fun**" Example: Sunspring



Today's "**fun**" Example: DopeLearning

DopeLearning: A Computational Approach to Rap Lyrics Generation*

Eric Malmi
Aalto University and HIIT
Espoo, Finland
eric.malmi@aalto.fi

Pyry Takala
Aalto University
Espoo, Finland
pyry.takala@aalto.fi

Hannu Toivonen
University of Helsinki and HIIT
Helsinki, Finland
hannu.toivonen@cs.helsinki.fi

Tapani Raiko
Aalto University
Espoo, Finland
tapani.raiko@aalto.fi

Aristides Gionis
Aalto University and HIIT
Espoo, Finland
aristides.gionis@aalto.fi

http://deepbeat.org

Lecture 18: Re-Cap

— We motivated SIFT for identifying locally distinct keypoints in an image (**detection**)

— SIFT features (**description**) are invariant to translation, rotation, and scale; robust to 3D pose and illumination

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

Lecture 18: Re-Cap

Keypoint is an image location at which a descriptor is computed

— Locally distinct points

— Easily localizable and identifiable

The feature **descriptor** summarizes the local structure around the key point

— Allows us to (hopefully) unique matching of keypoints in presence of object pose variations, image and photometric deformations

Note, for repetitive structure this would still not give us unique matches.

Lecture 18: Re-Cap

Keypoint is an image location at which a descriptor is computed

— Locally distinct points

— Easily localizable and identifiable

The feature **descriptor** summarizes the local structure around the key point

— Allows us to (hopefully) unique matching of keypoints in presence of object pose variations, image and photometric deformations

Note, for repetitive structure this would still not give us unique matches.

Lecture 18: Re-Cap

Keypoint is an image location at which a descriptor is computed

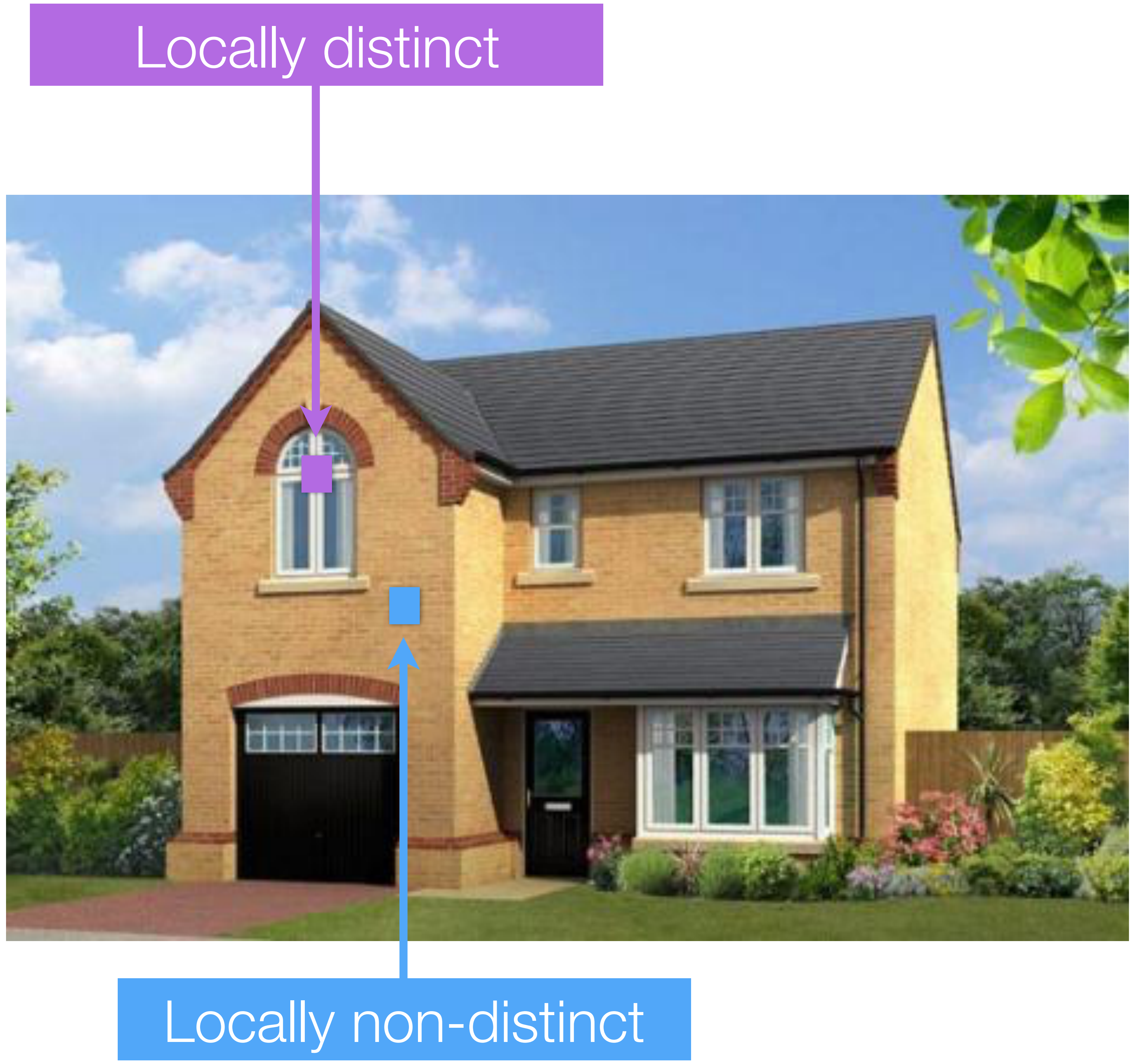
— Locally distinct points

— Easily localizable and identifiable

The feature **descriptor** summarizes the local structure around the key point

— Allows us to (hopefully) unique matching of keypoints in presence of object pose variations, image and photometric deformations

Note, for repetitive structure this would still not give us unique matches.



Lecture 18: Re-Cap

Keypoint is an image location at which a descriptor is computed

— Locally distinct points

— Easily localizable and identifiable

The feature **descriptor** summarizes the local structure around the key point

— Allows us to (hopefully) unique matching of keypoints in presence of object pose variations, image and photometric deformations

Note, for repetitive structure this would still not give us unique matches.

Lecture 18: Re-Cap

Keypoint is an image location at which a descriptor is computed

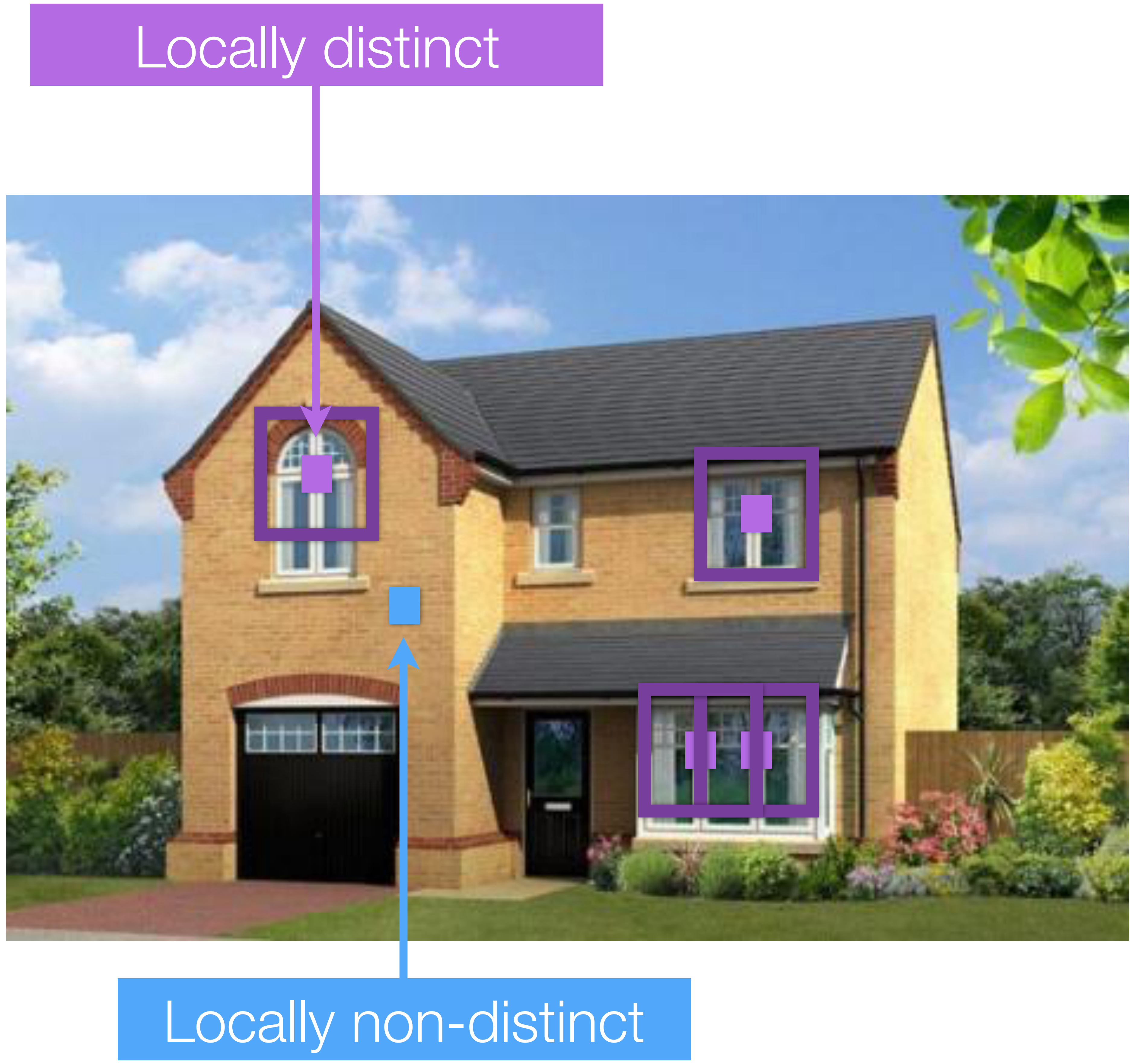
— Locally distinct points

— Easily localizable and identifiable

The feature **descriptor** summarizes the local structure around the key point

— Allows us to (hopefully) unique matching of keypoints in presence of object pose variations, image and photometric deformations

Note, for repetitive structure this would still not give us unique matches.



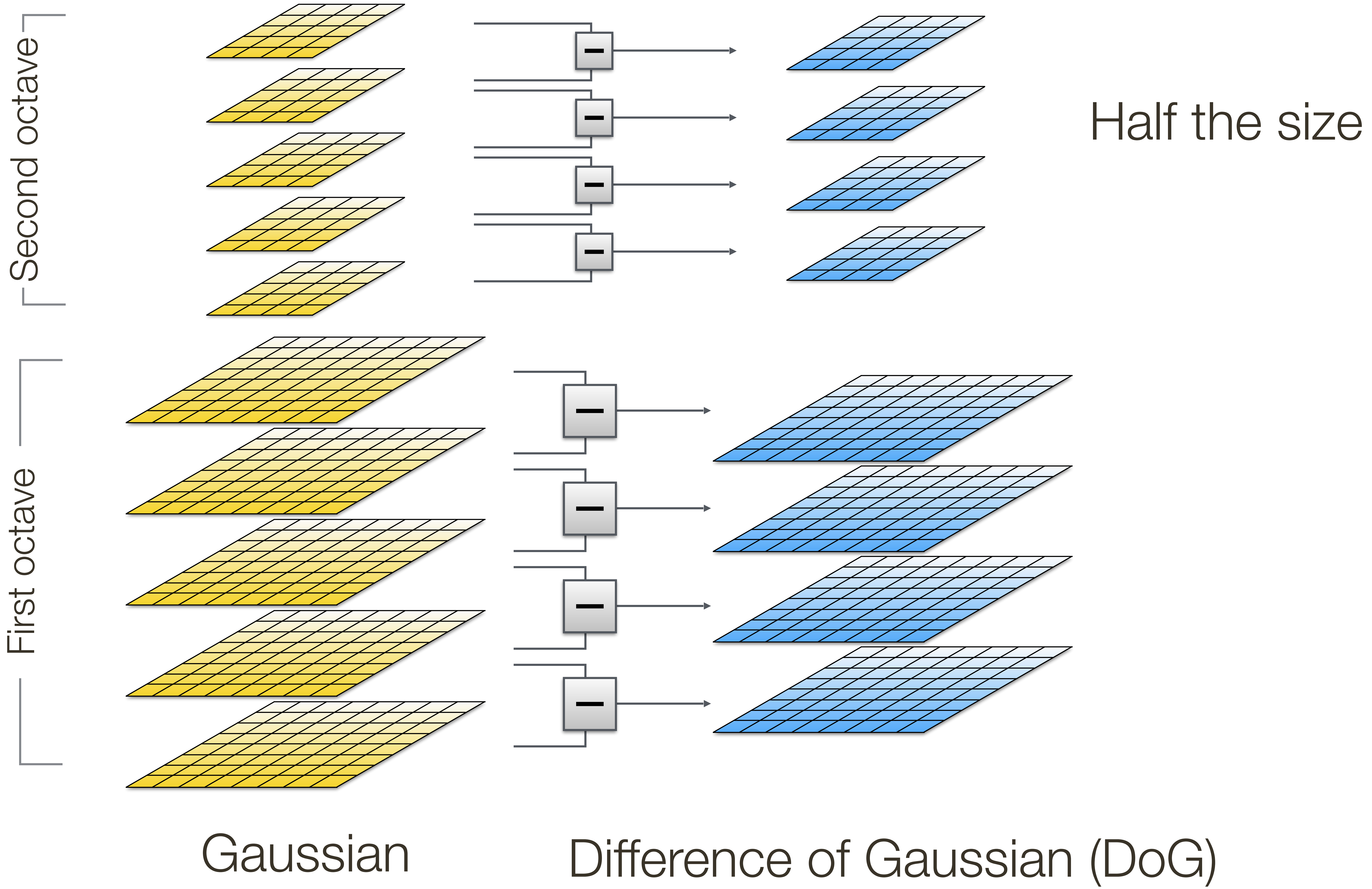
Lecture 18: Re-Cap

— We motivated SIFT for identifying locally distinct keypoints in an image (**detection**)

— SIFT features (**description**) are invariant to translation, rotation, and scale; robust to 3D pose and illumination

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

1. Multi-scale Extrema Detection

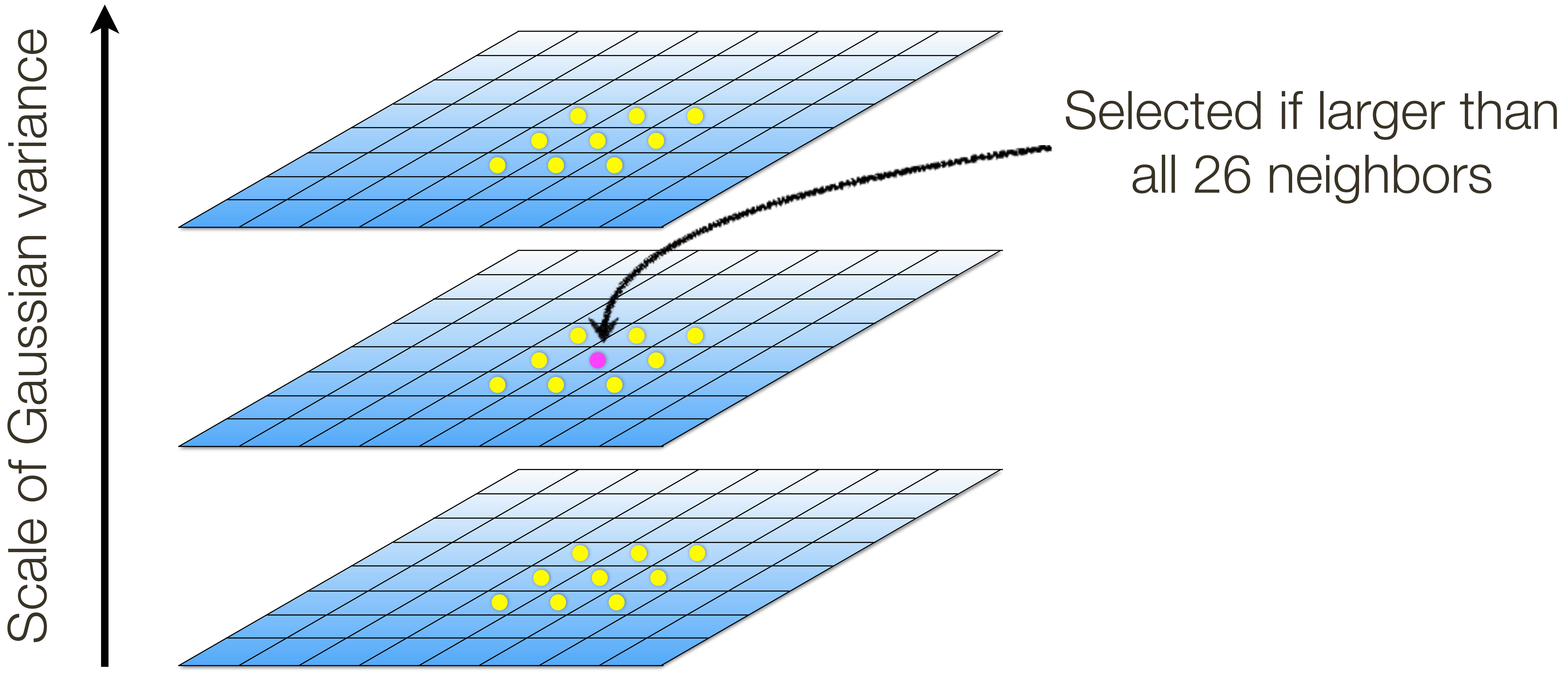


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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)

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?

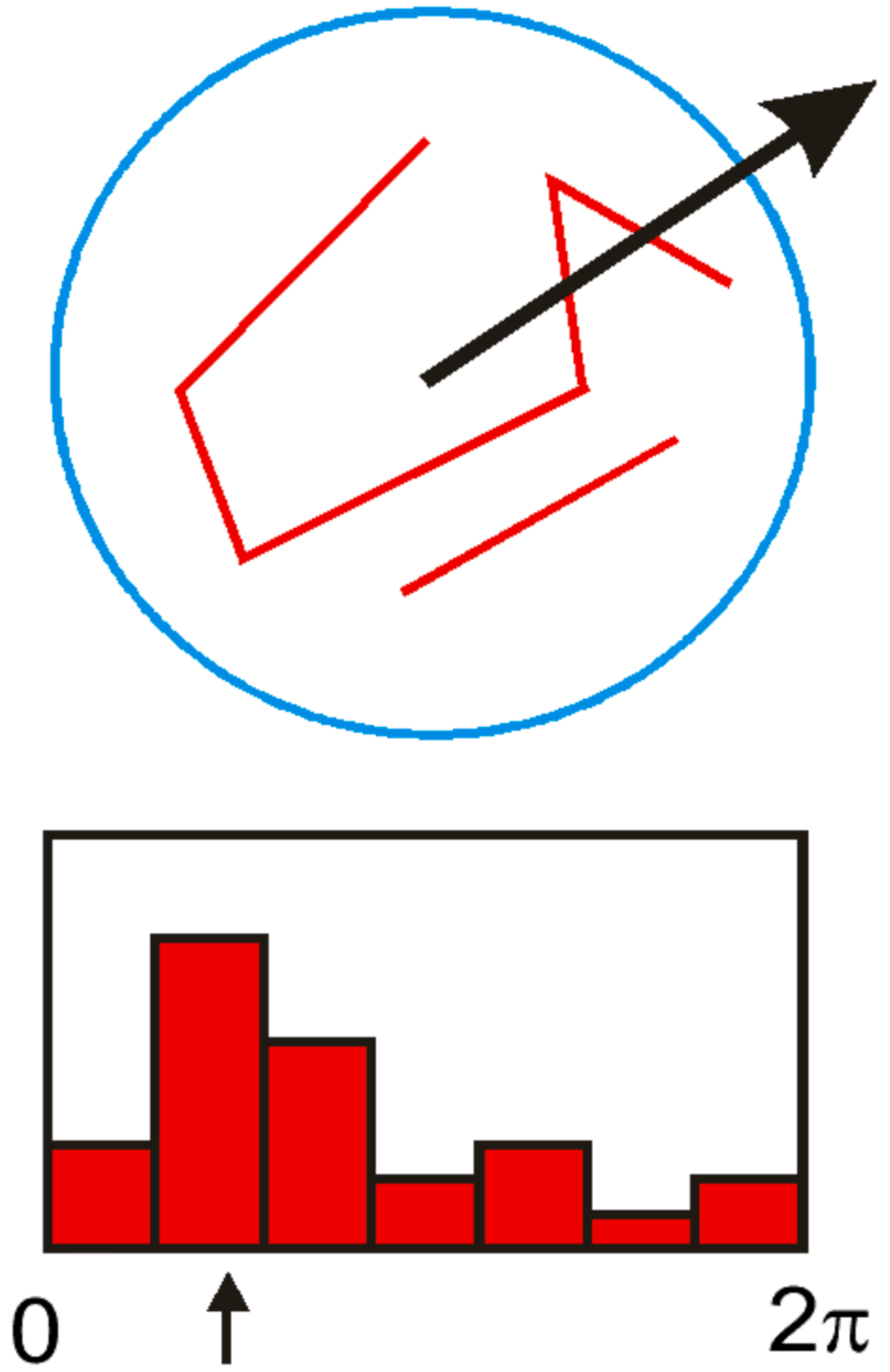
$$C = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

3. Orientation Assignment

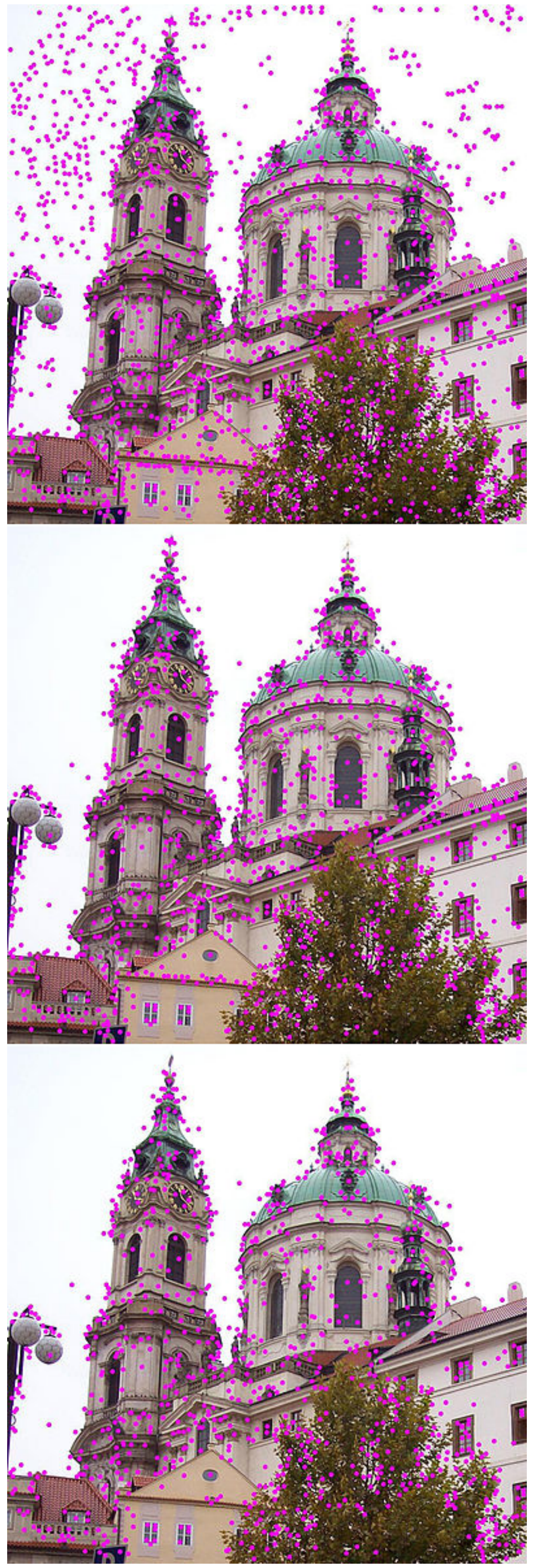
— Create **histogram** of local gradient directions computed at selected scale

— 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

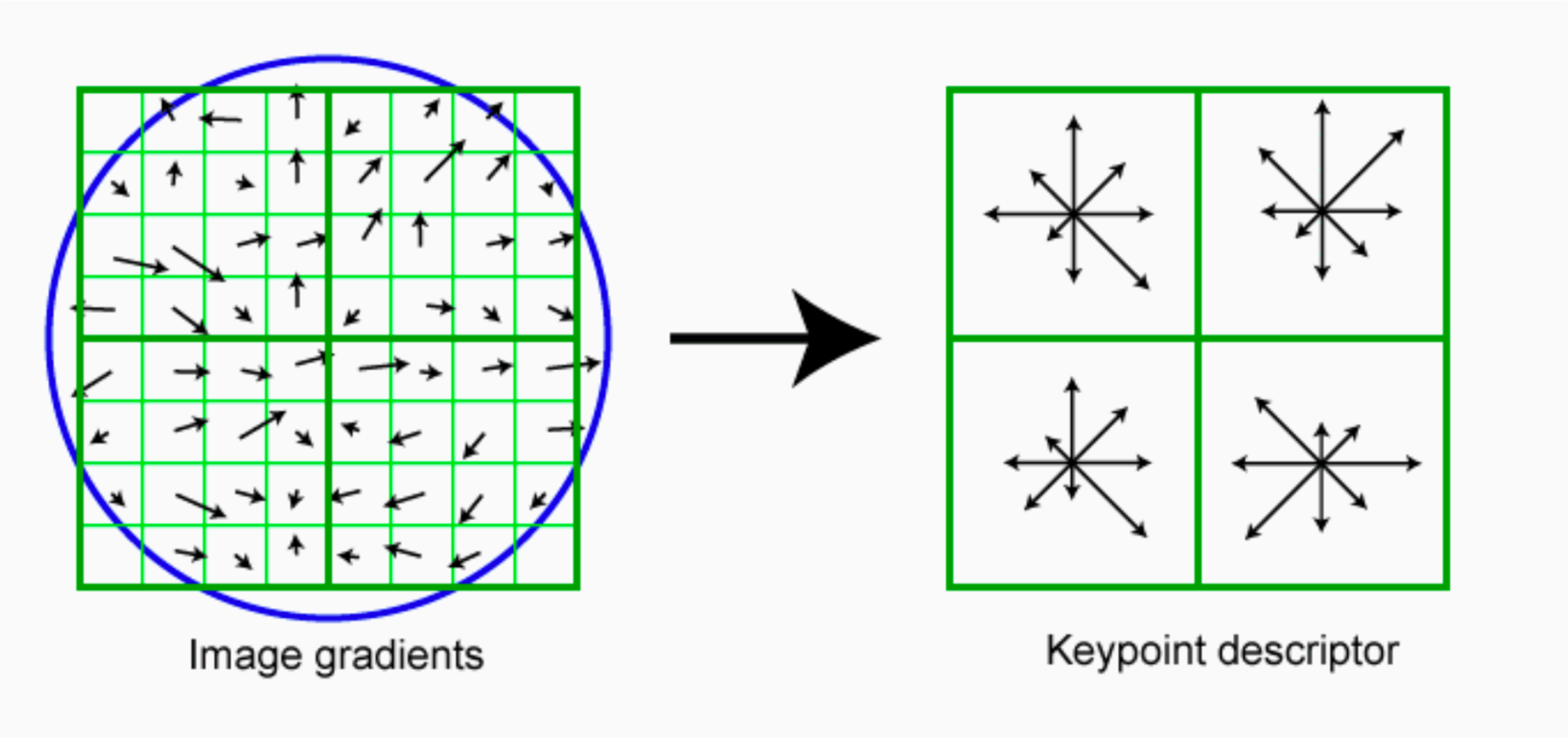
— Keypoint detection is not the same as keypoint description, e.g. some applications skip keypoint detection and extract SIFT descriptors on a regularly spaced grid

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

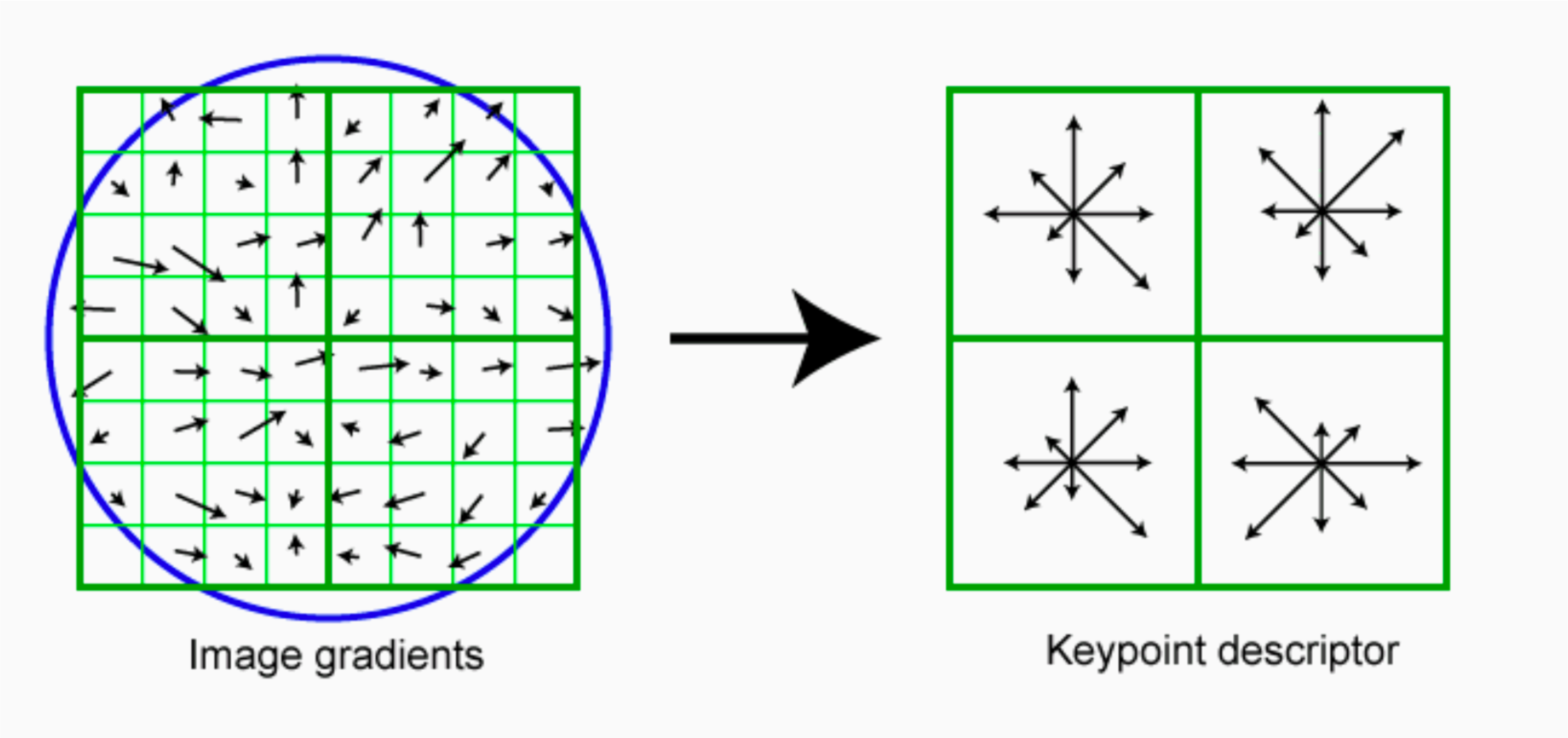


Demo

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

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

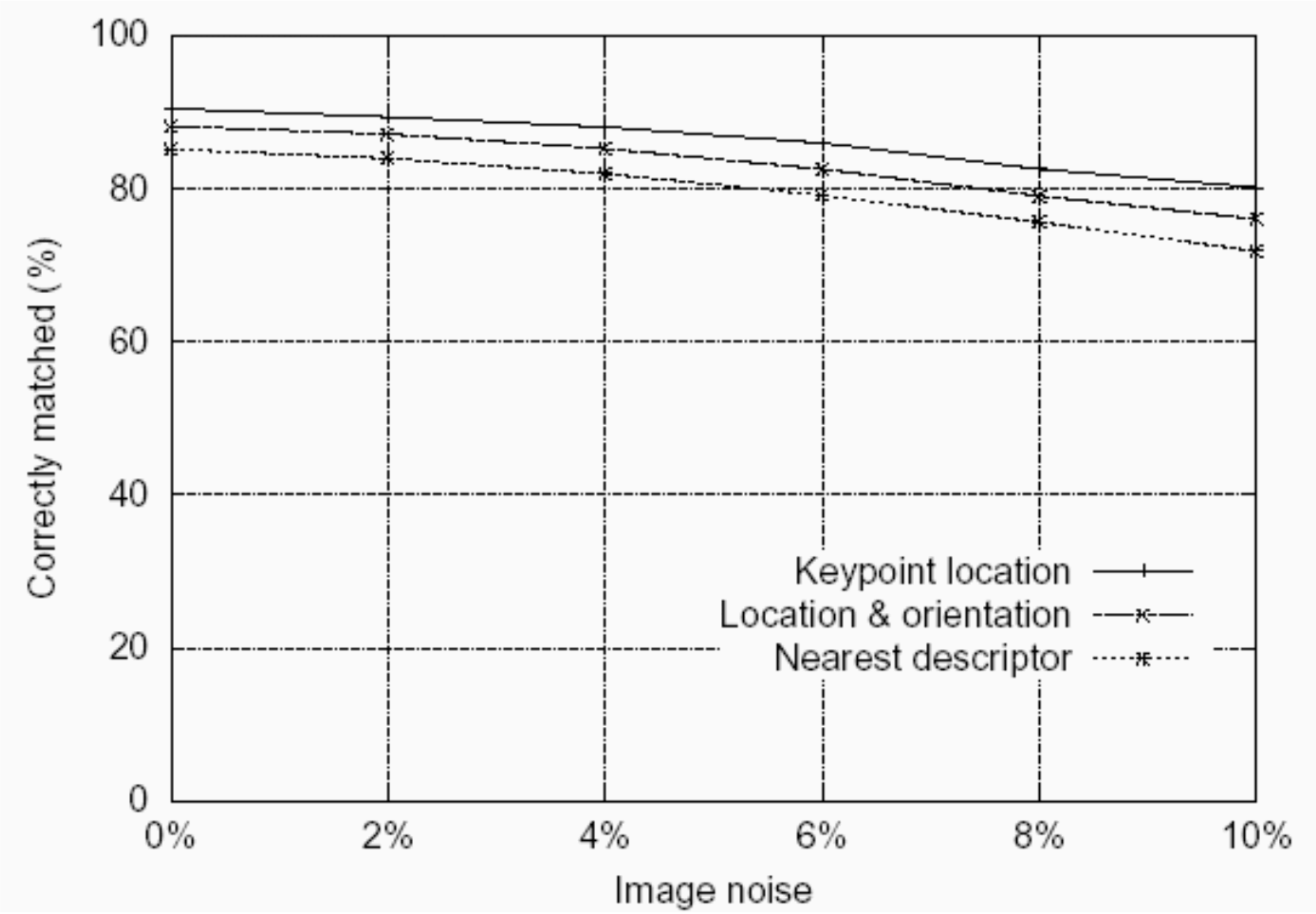
— if brightness values are scaled (multiplied) by a constant, the gradients are scaled by the same constant, and the normalization cancels the change

— 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

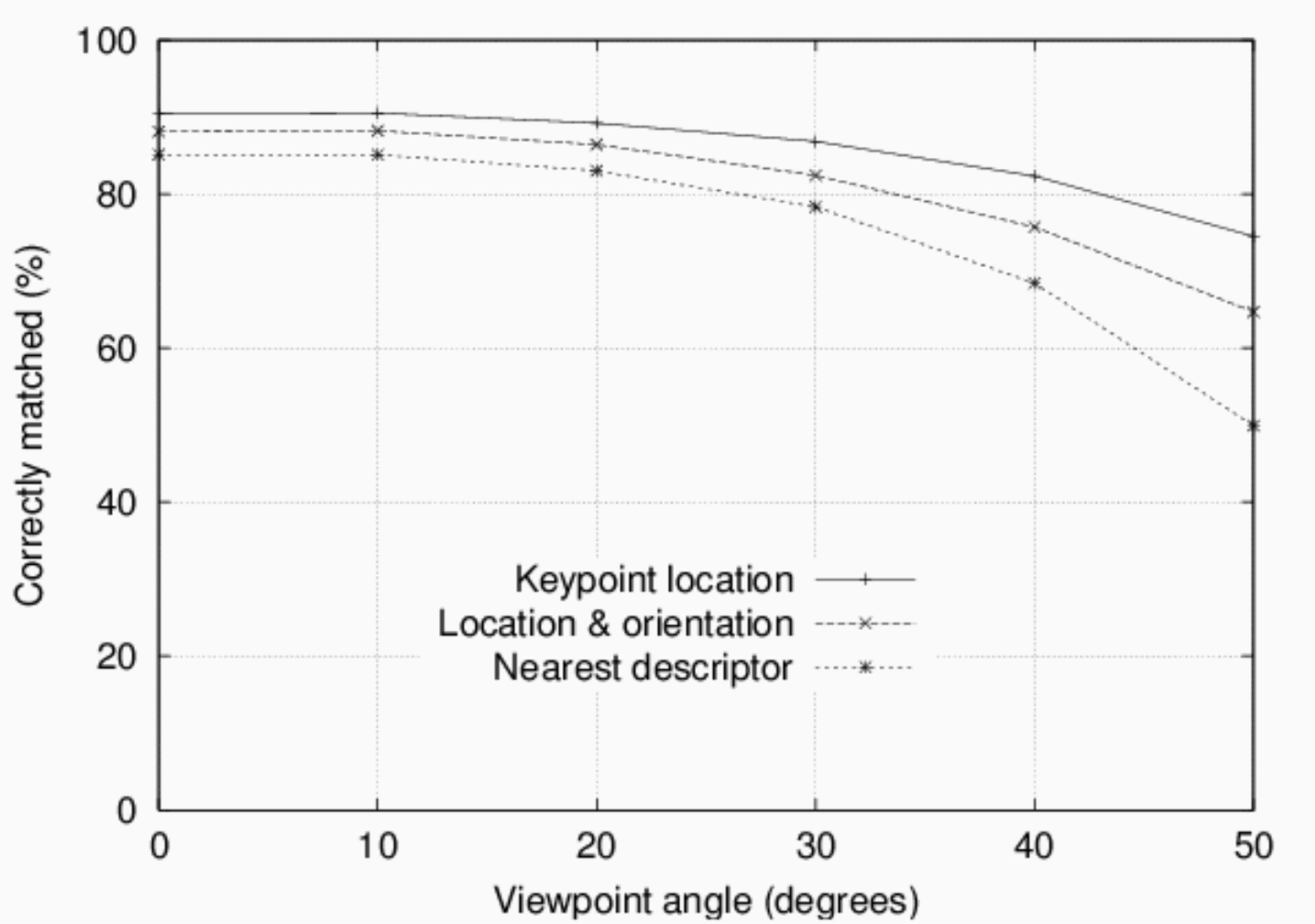
Find nearest neighbour in database of 30,000 features



Feature Stability to **Affine Change**

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

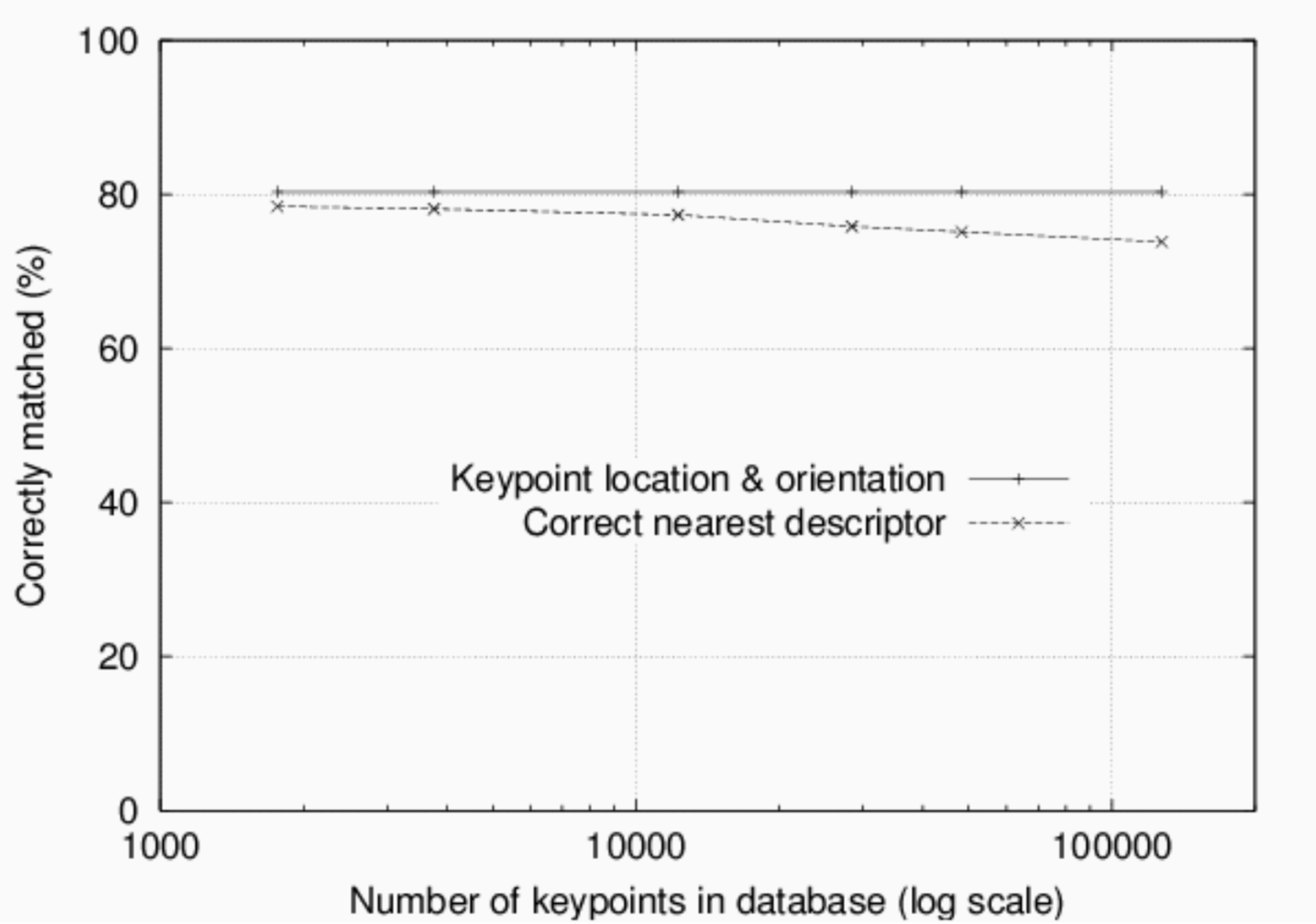
Find nearest neighbour in database of 30,000 features



Distinctiveness of Features

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

Measure % correct for single nearest neighbour match



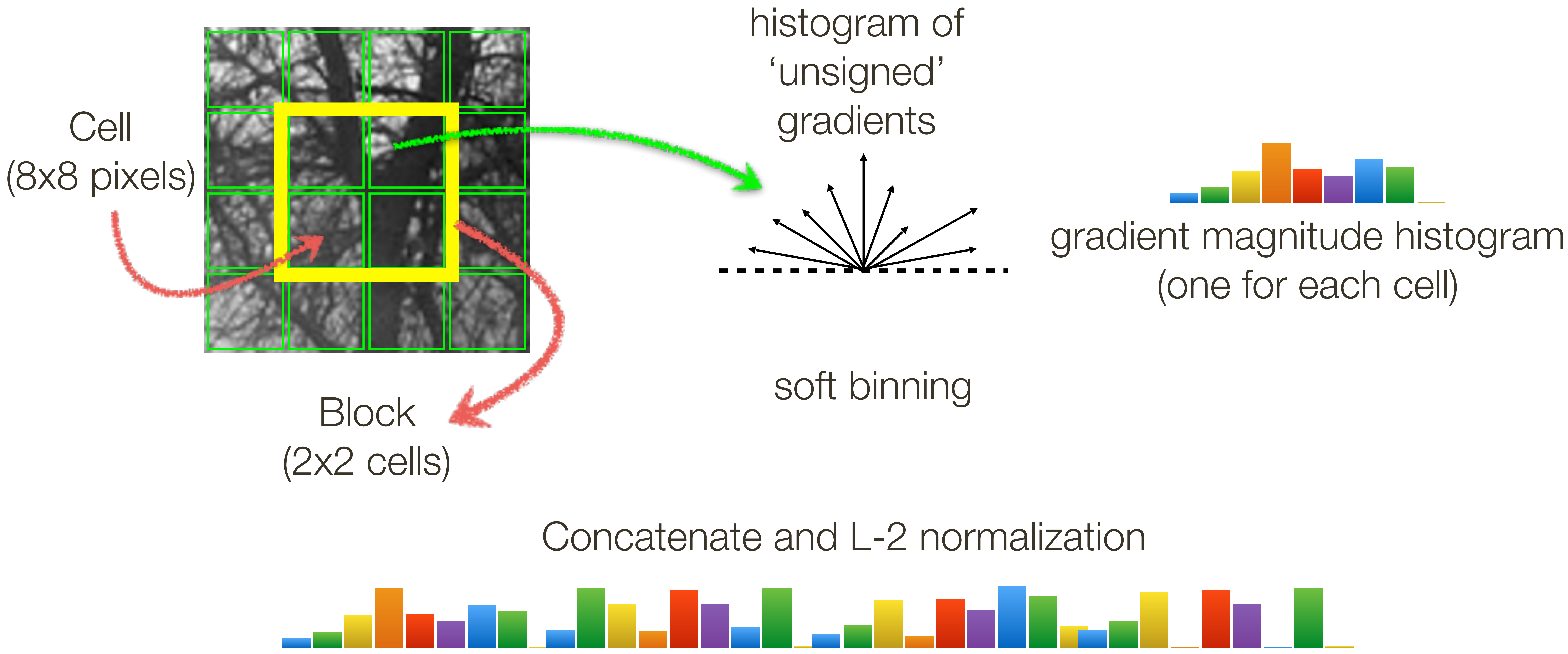
Summary

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

1 cell step size

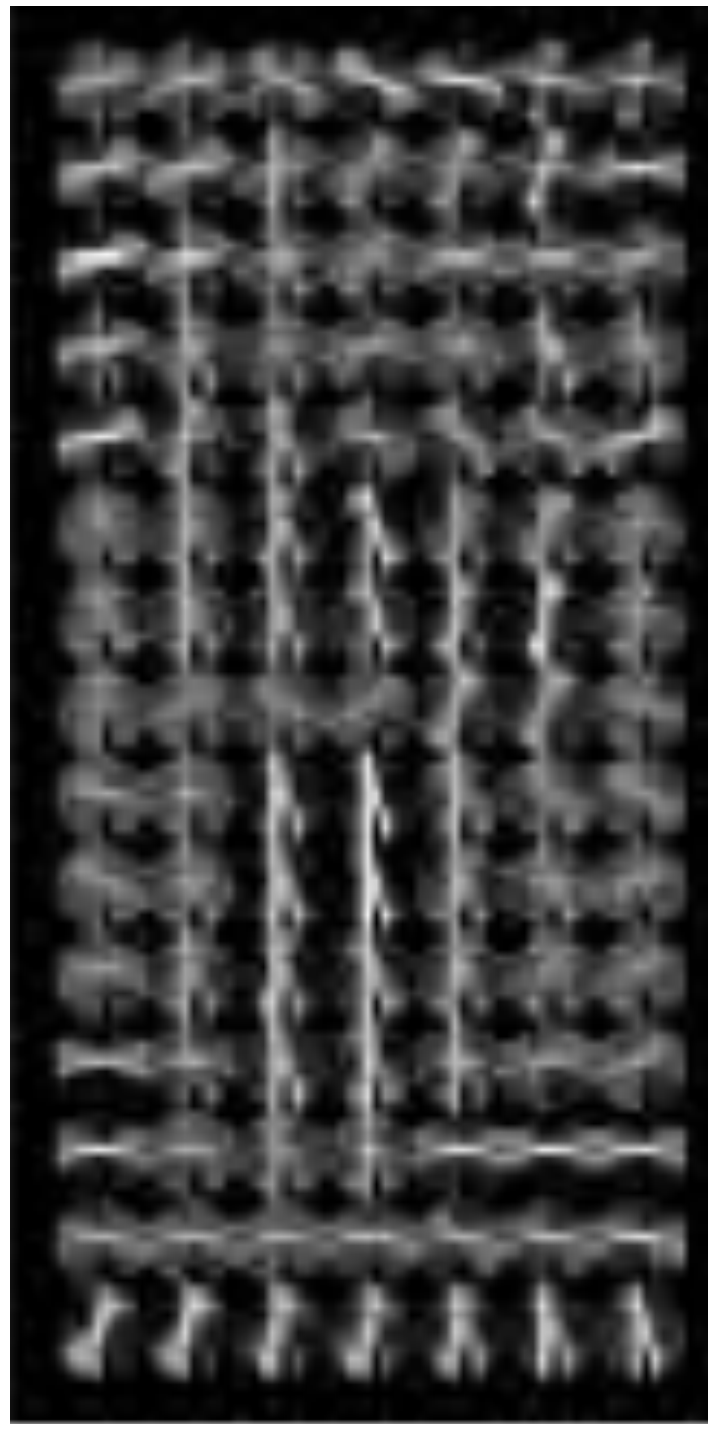
Pedestrian detection

128 pixels
16 cells
15 blocks

64 pixels
8 cells
7 blocks

15 x 7 x 4 x 9 =
3780

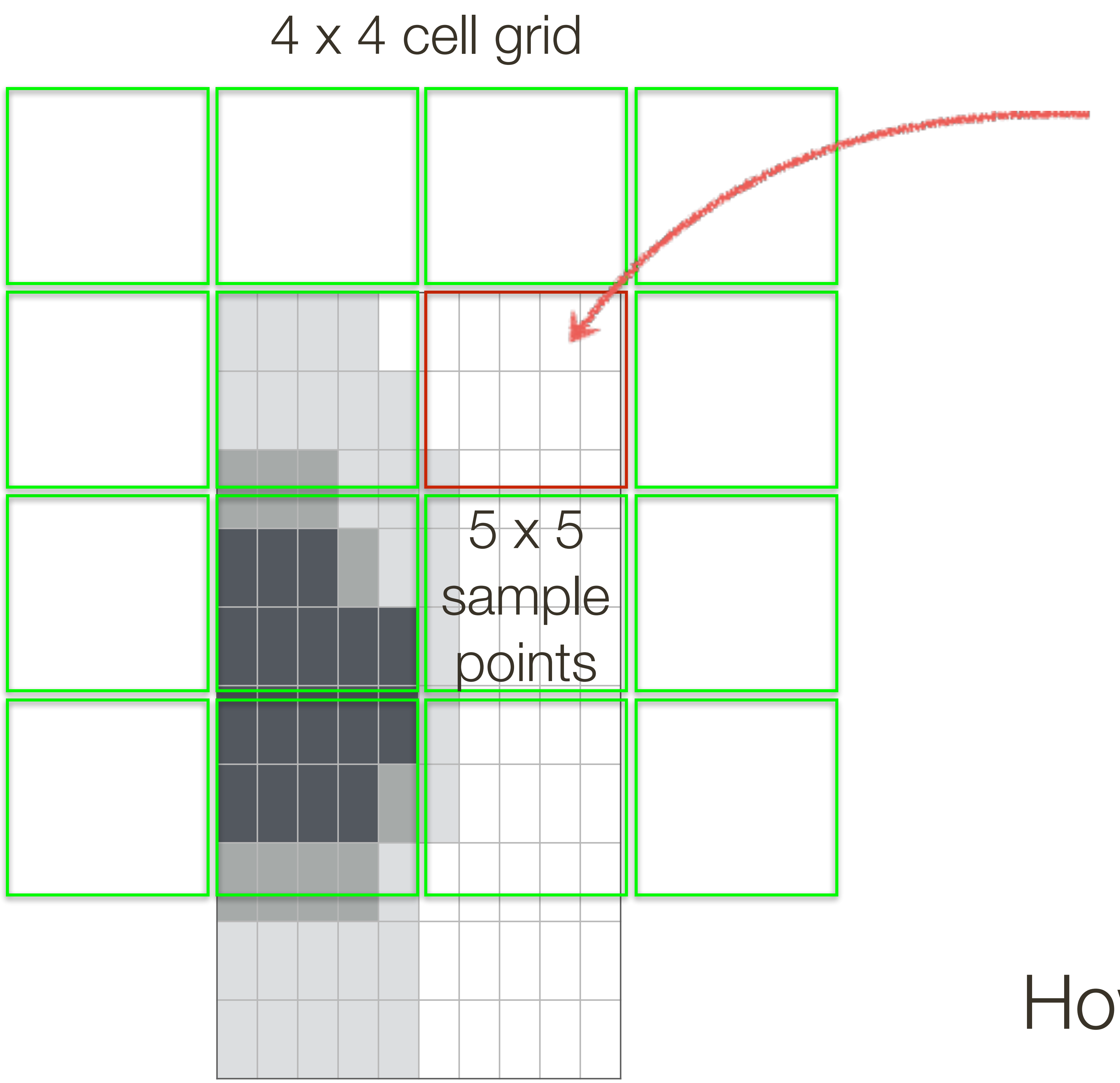
visualization



Redundant representation due to overlapping blocks

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

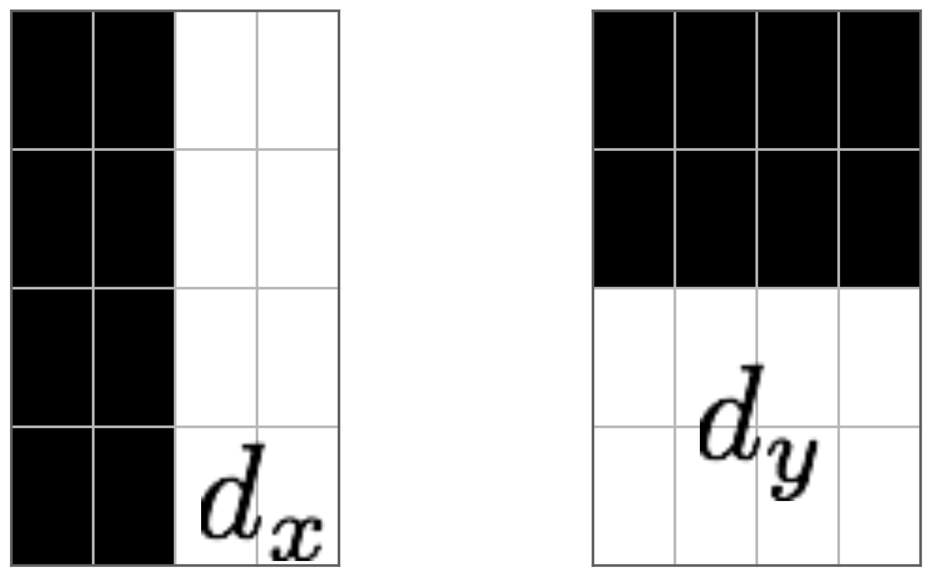
‘Speeded’ Up Robust Features (**SURF**)



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)

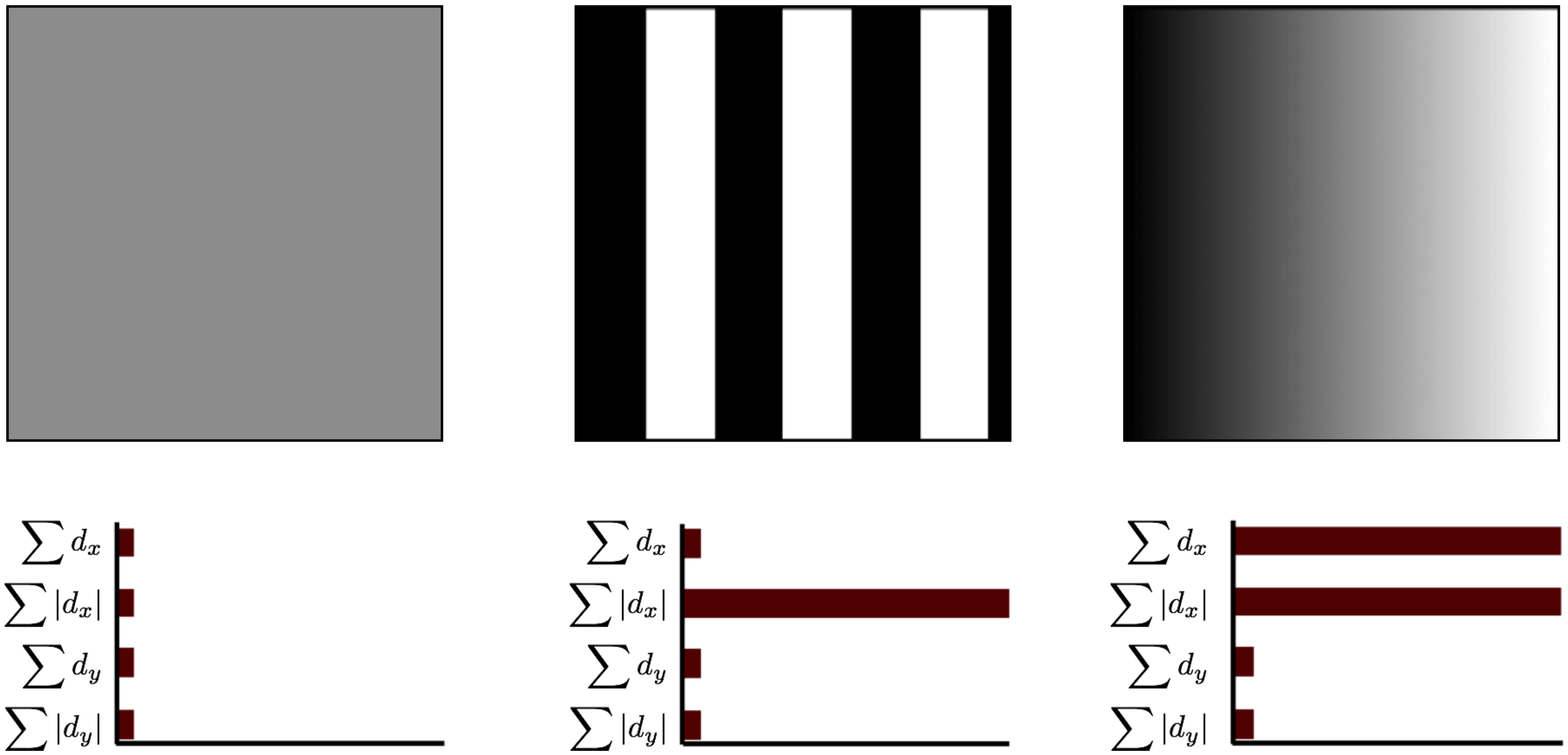


How big is the SURF descriptor?

64 dimensions

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

SIFT and **Object Recognition**

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

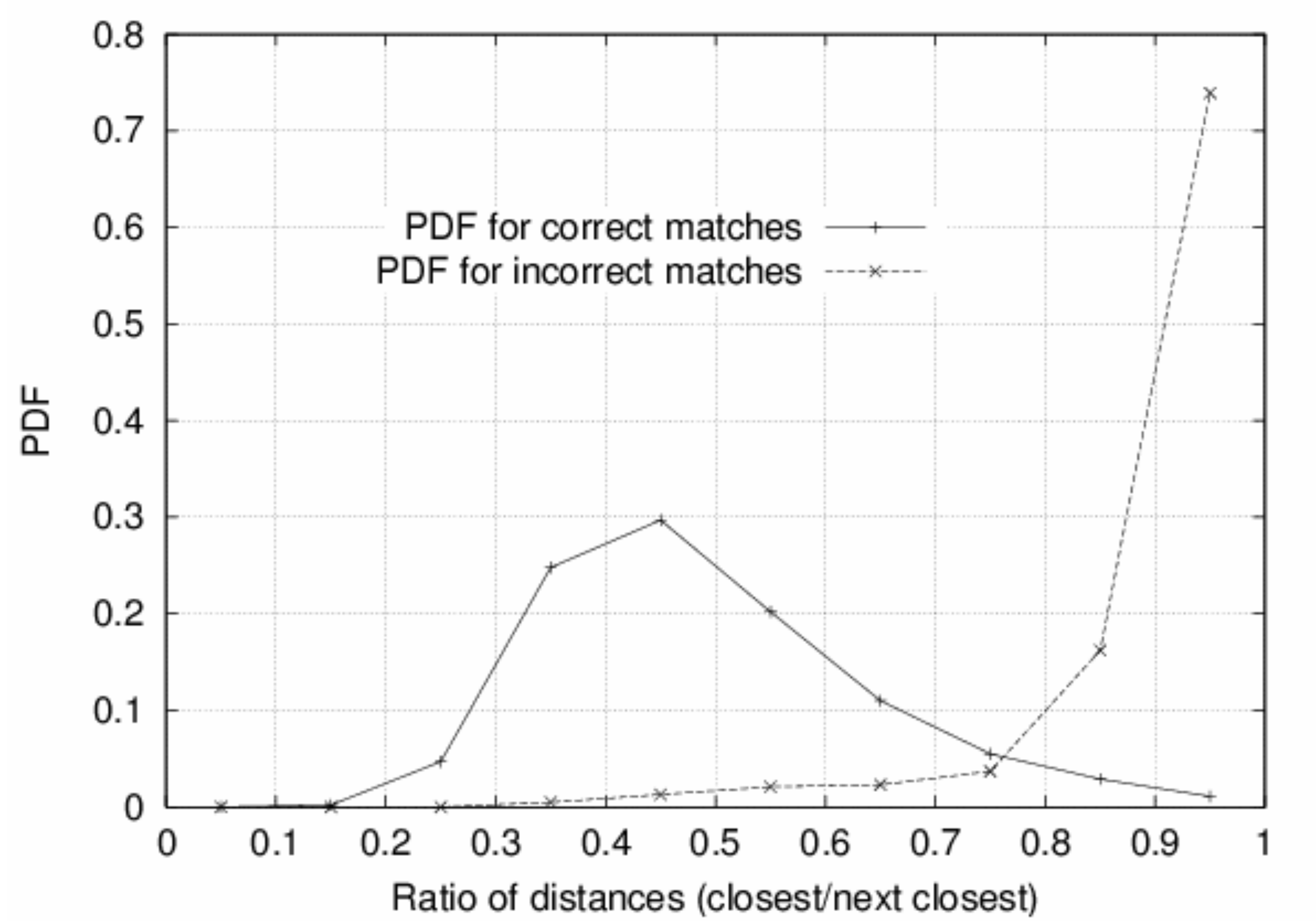
Many features will not have any correct match in the database because they arise from background clutter

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



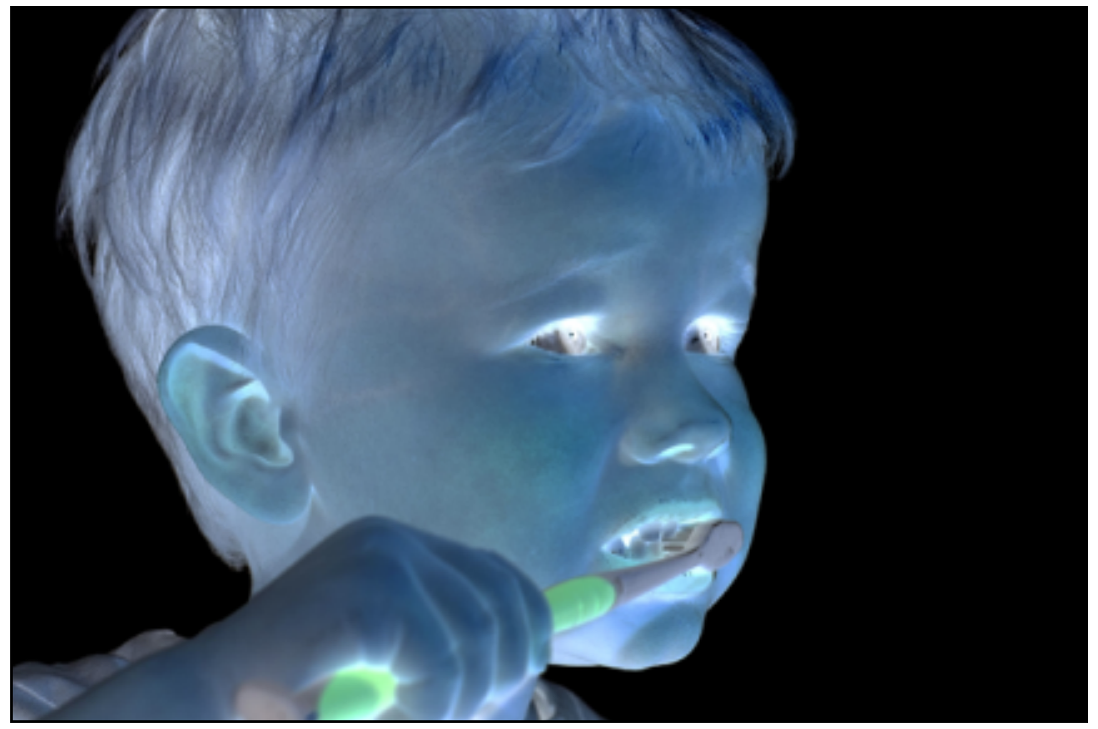
$$\frac{\text{closest}}{\text{next closest}}$$

What types of **transformations** can we do?

$I(X,Y)$

Filtering

$I'(X,Y)$

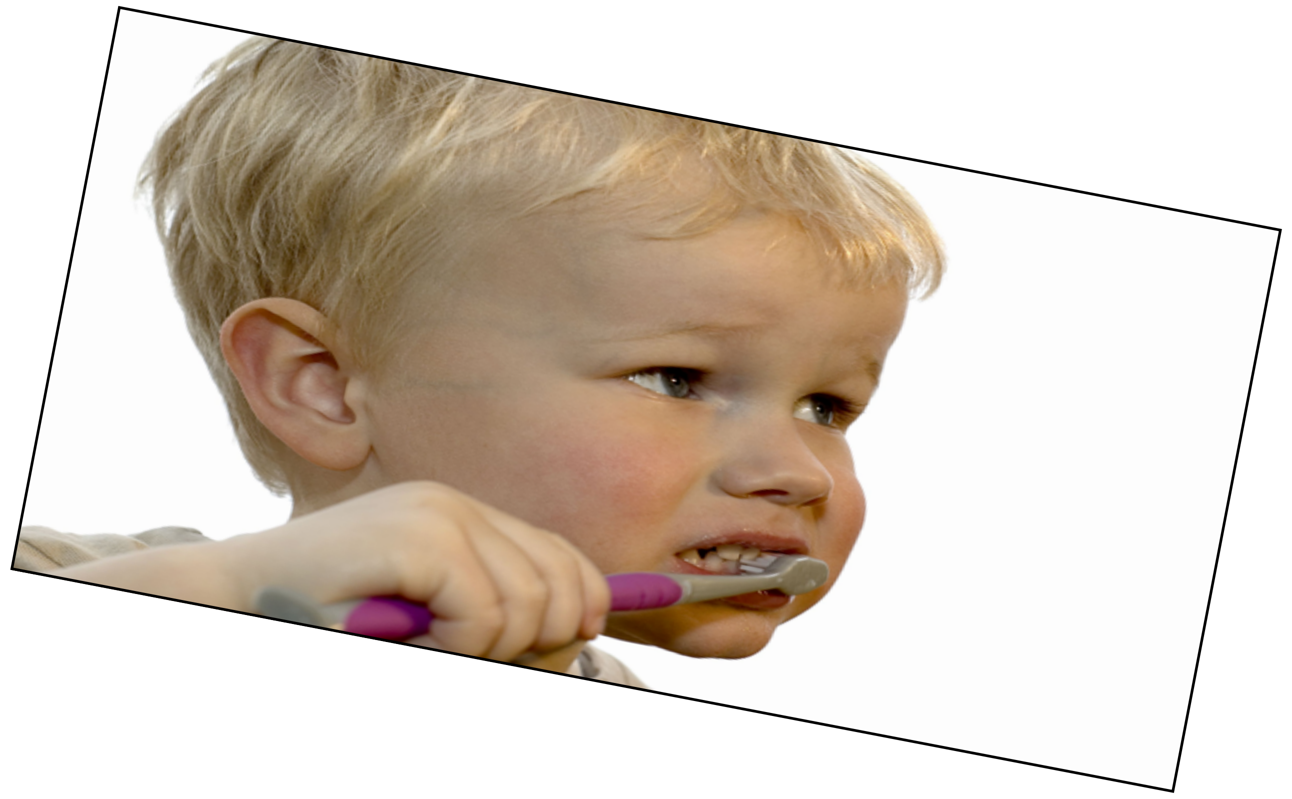


changes range of image function

$I(X,Y)$

Warping

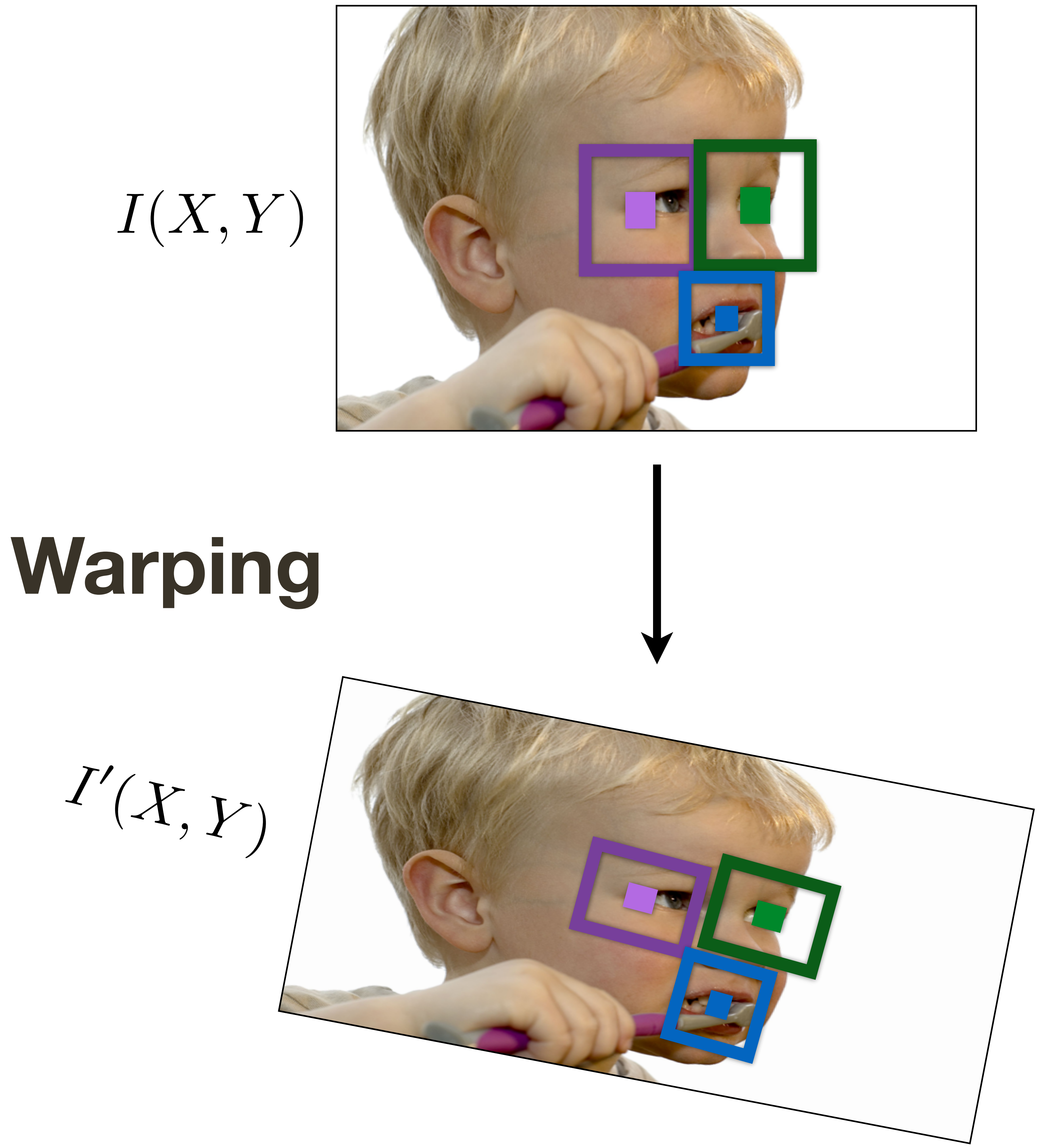
$I'(X,Y)$



changes domain of image function

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

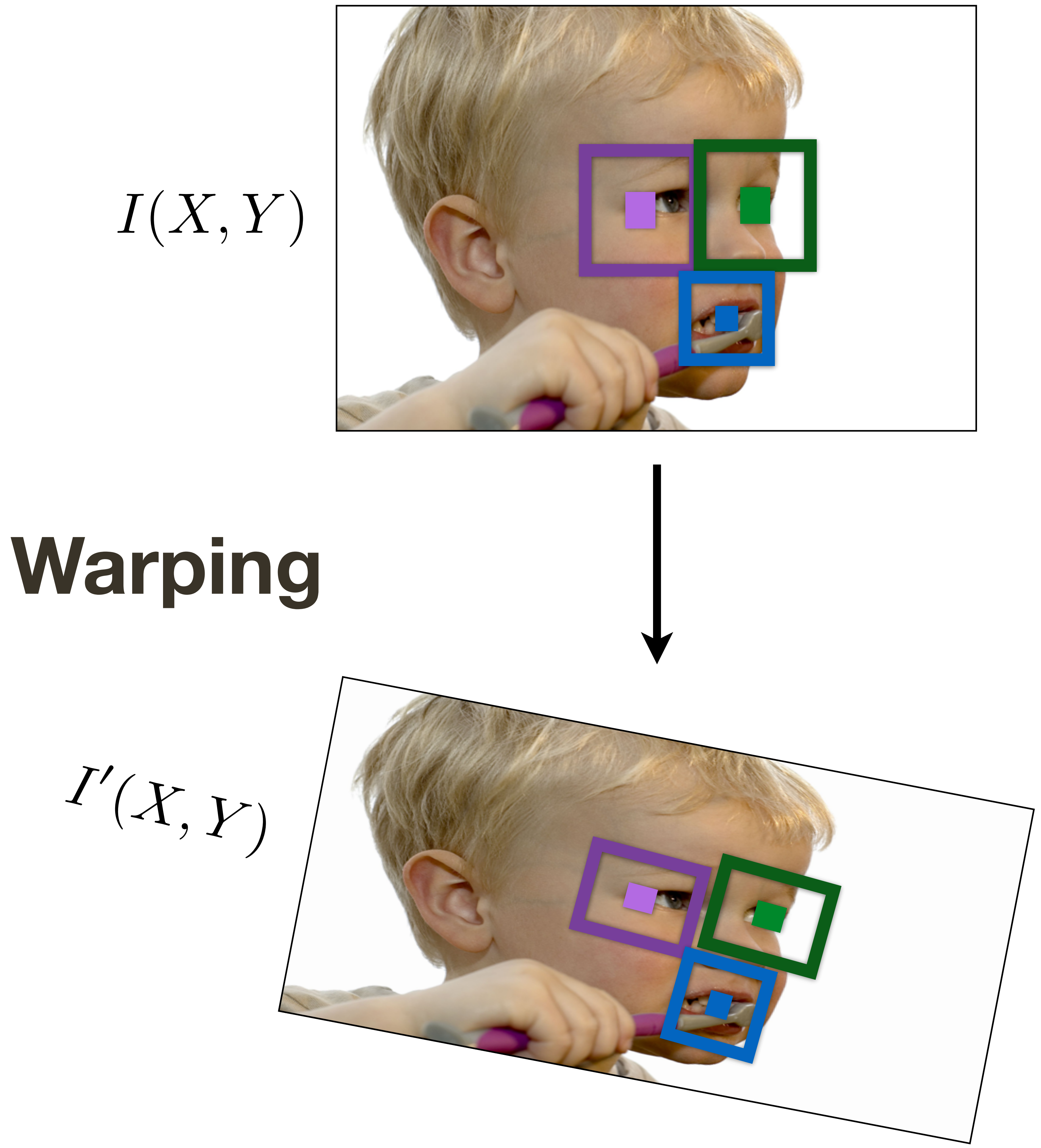
What types of **transformations** can we do?



changes domain of image function

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

What types of **transformations** can we do?



We will call this
"**Warping**" a "**Model**"

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = M \begin{bmatrix} X \\ Y \end{bmatrix}$$

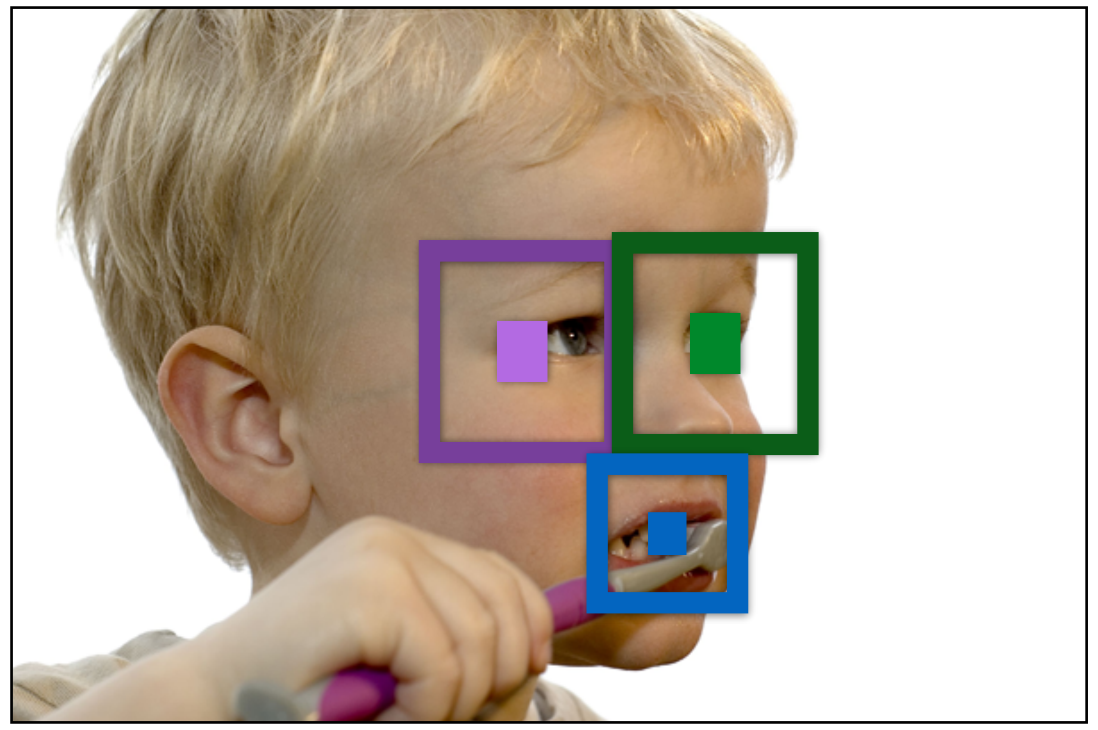
$$I'(X',Y') = I(X,Y)$$

changes domain of image function

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

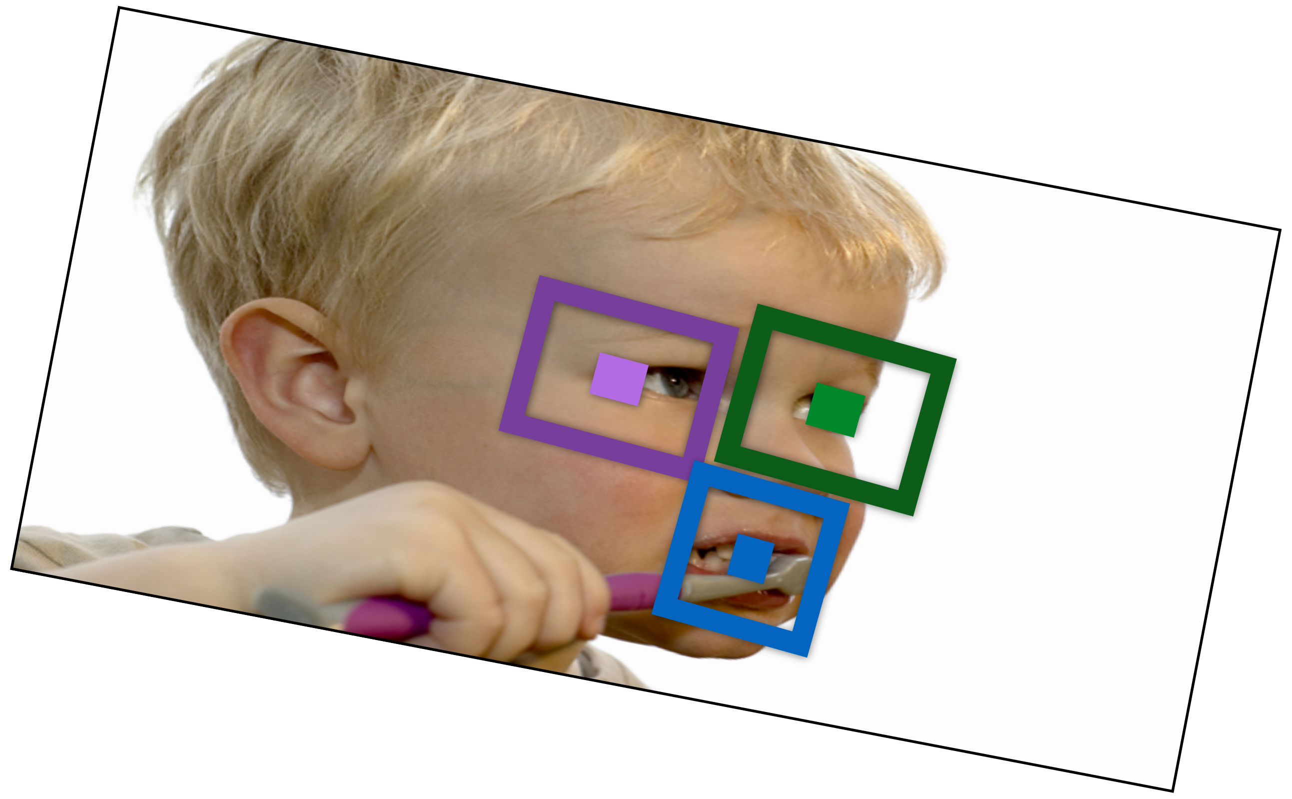
What types of **transformations** can we do?

$I(X,Y)$



Warping

$I'(X,Y)$



changes domain of image function

We will call this
"**Warping**" a "**Model**"

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = M \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$$I'(X',Y') = I(X,Y)$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Model **Verification**

1. Examine all clusters with at least 3 features

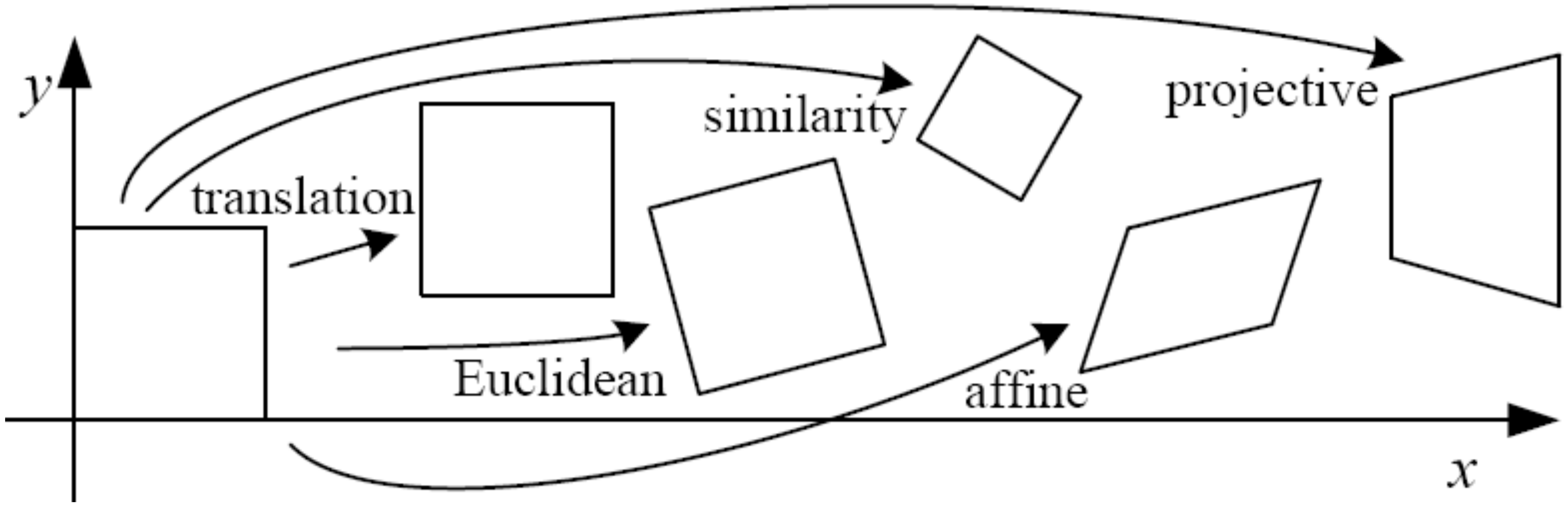
2. Perform least-squares affine **fit to model**

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)

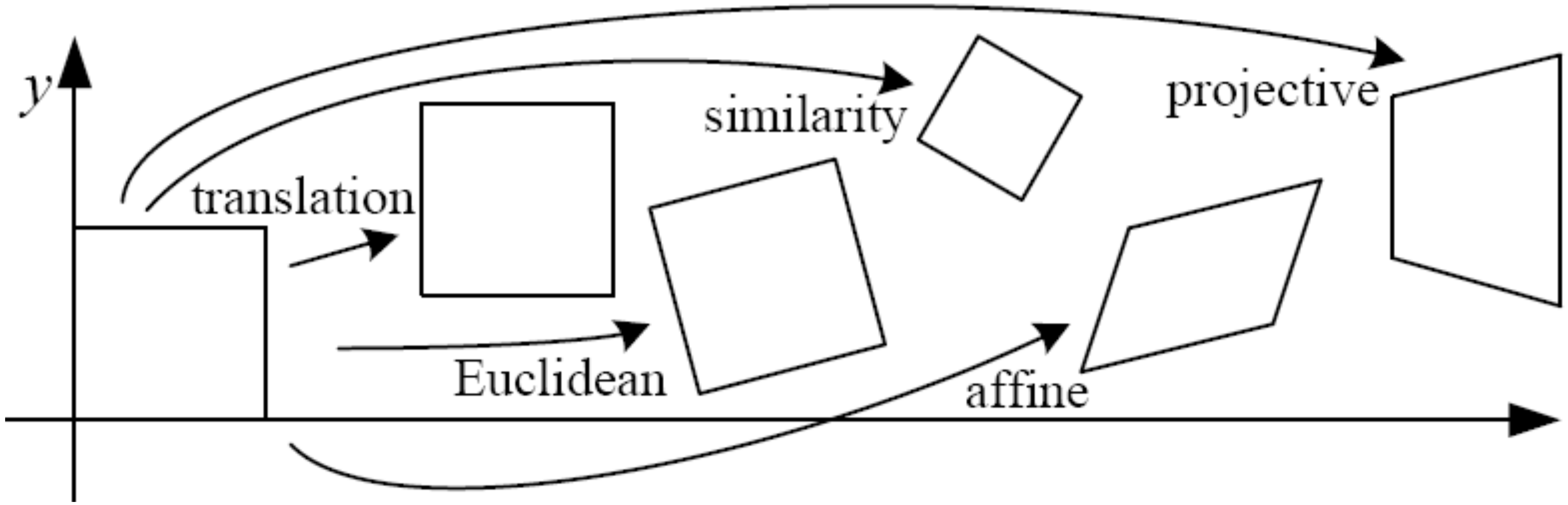
Aside: Classification of 2D Transformations



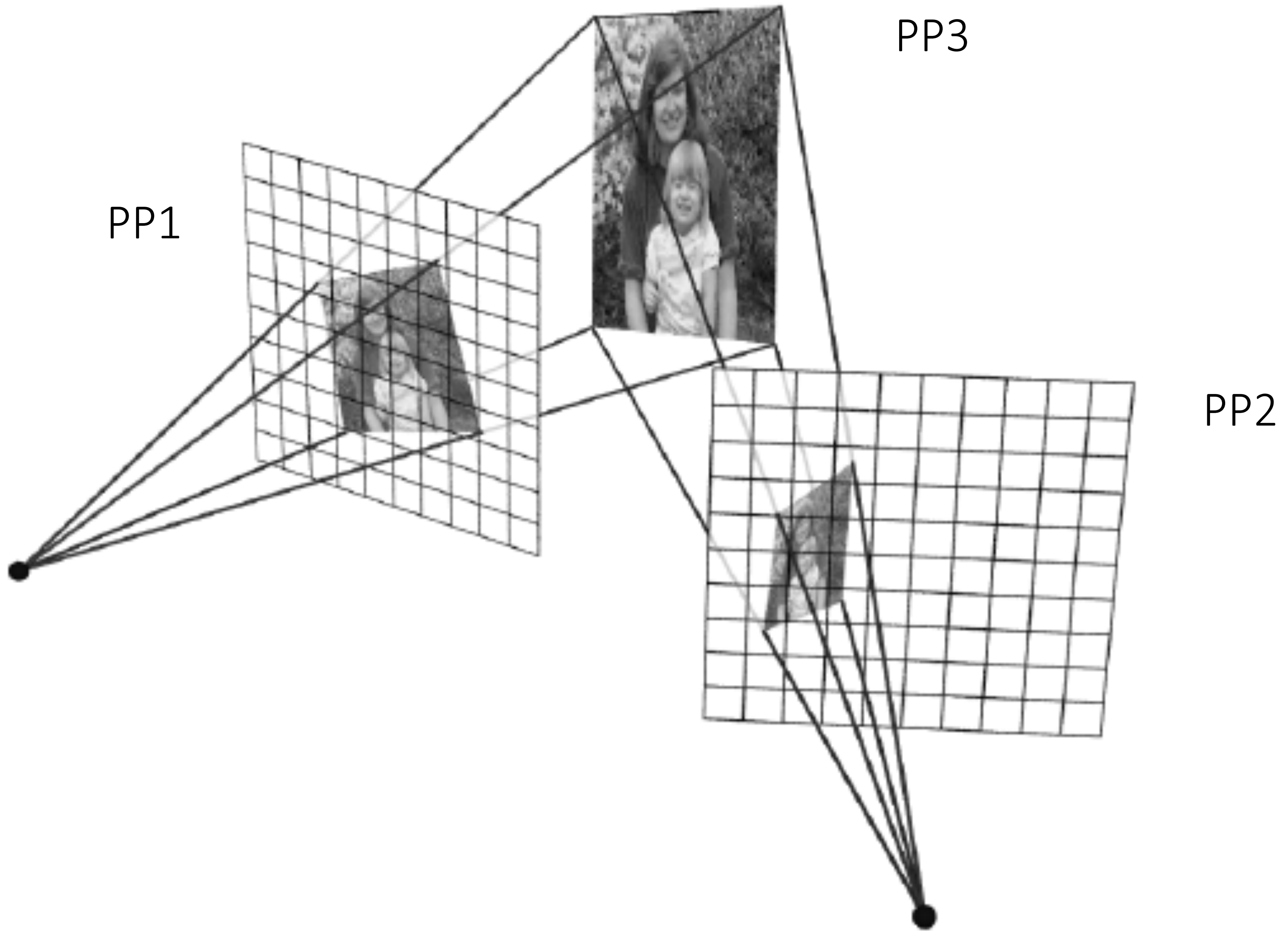
Name	Matrix	# D.O.F.
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	4
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$	6
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times 3}$	8

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Aside: Classification of 2D Transformations

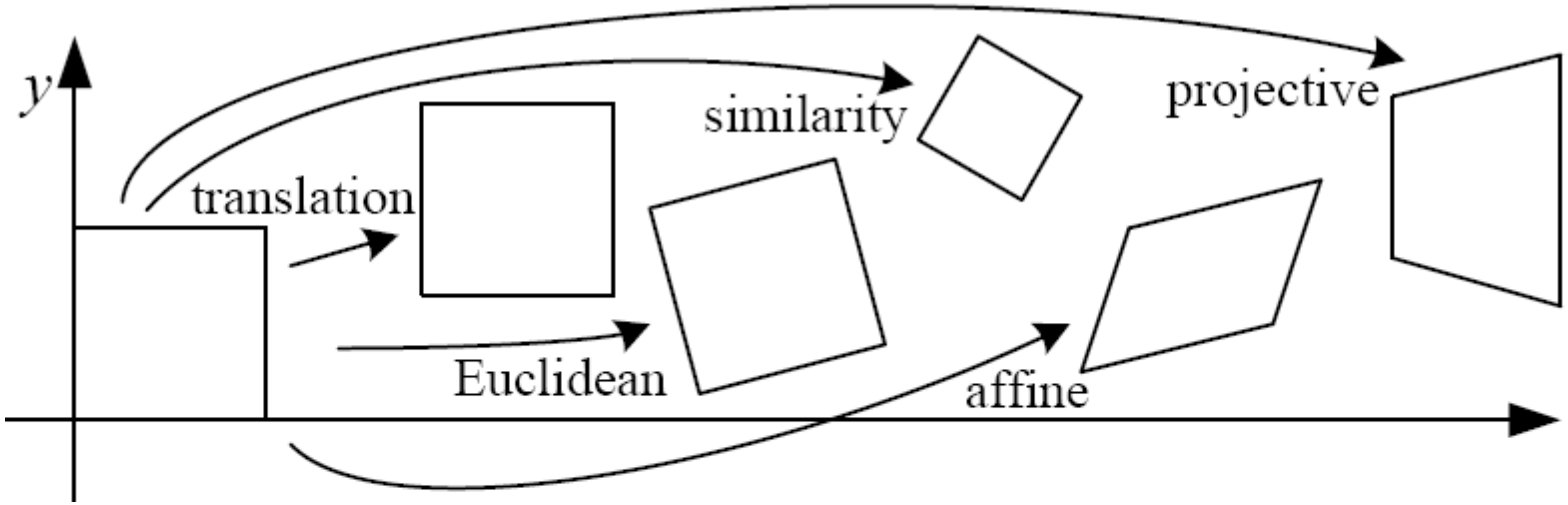


Which kind **transformation** is needed to warp projective plane 1 into projective plane 2?



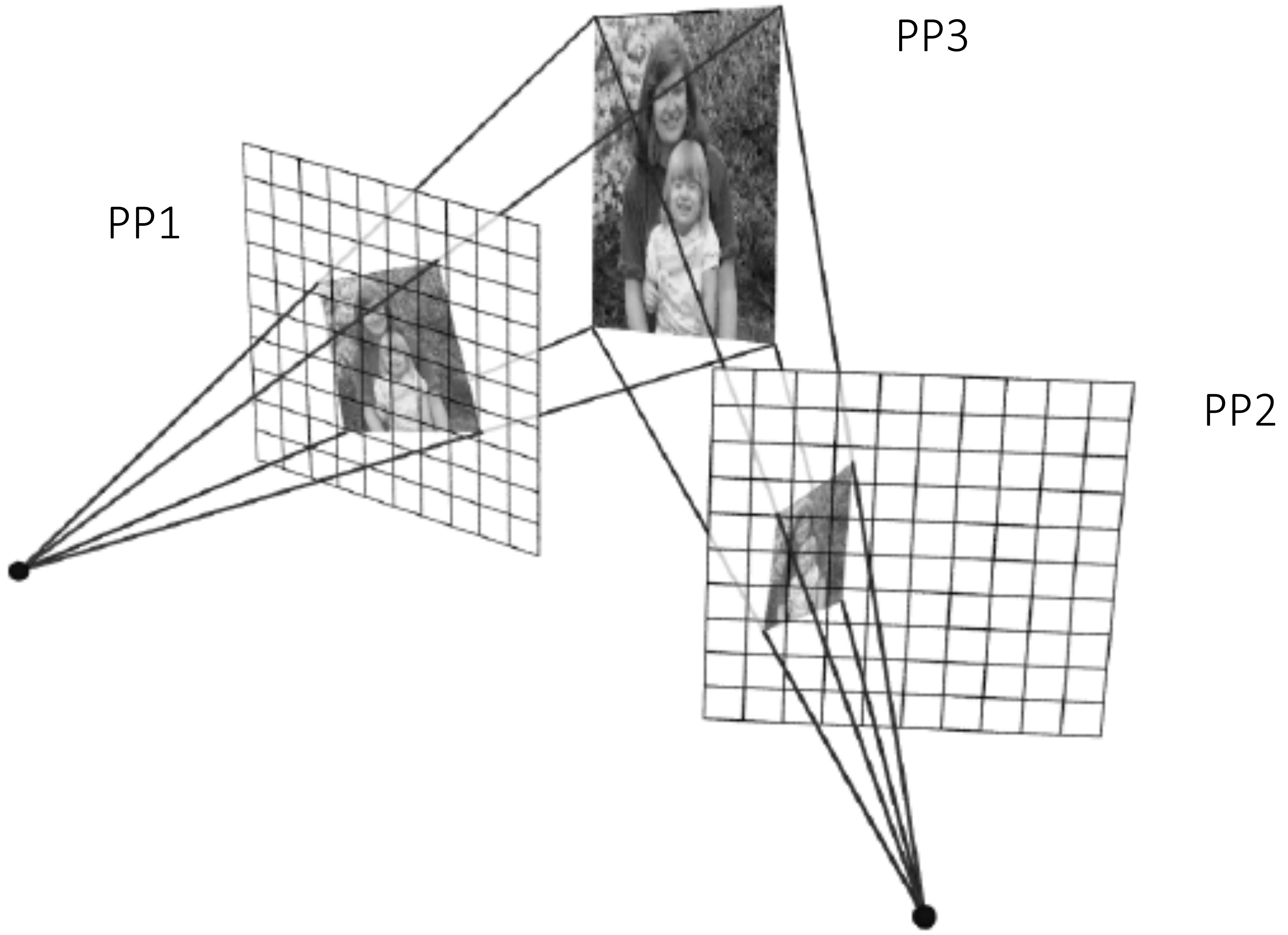
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Aside: Classification of 2D Transformations



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

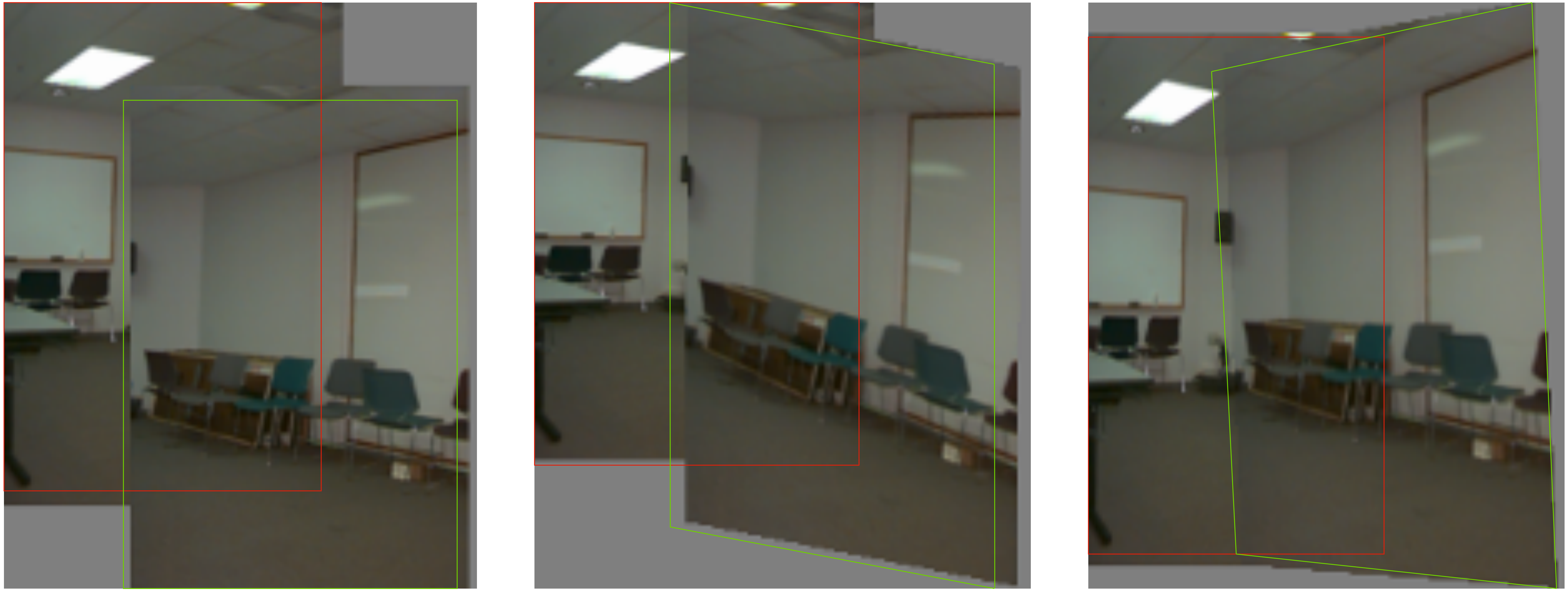
— A **projective** transformation (a.k.a. a homography).



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:

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

(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,

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

Limitation of this …

We need to have **exact** matches

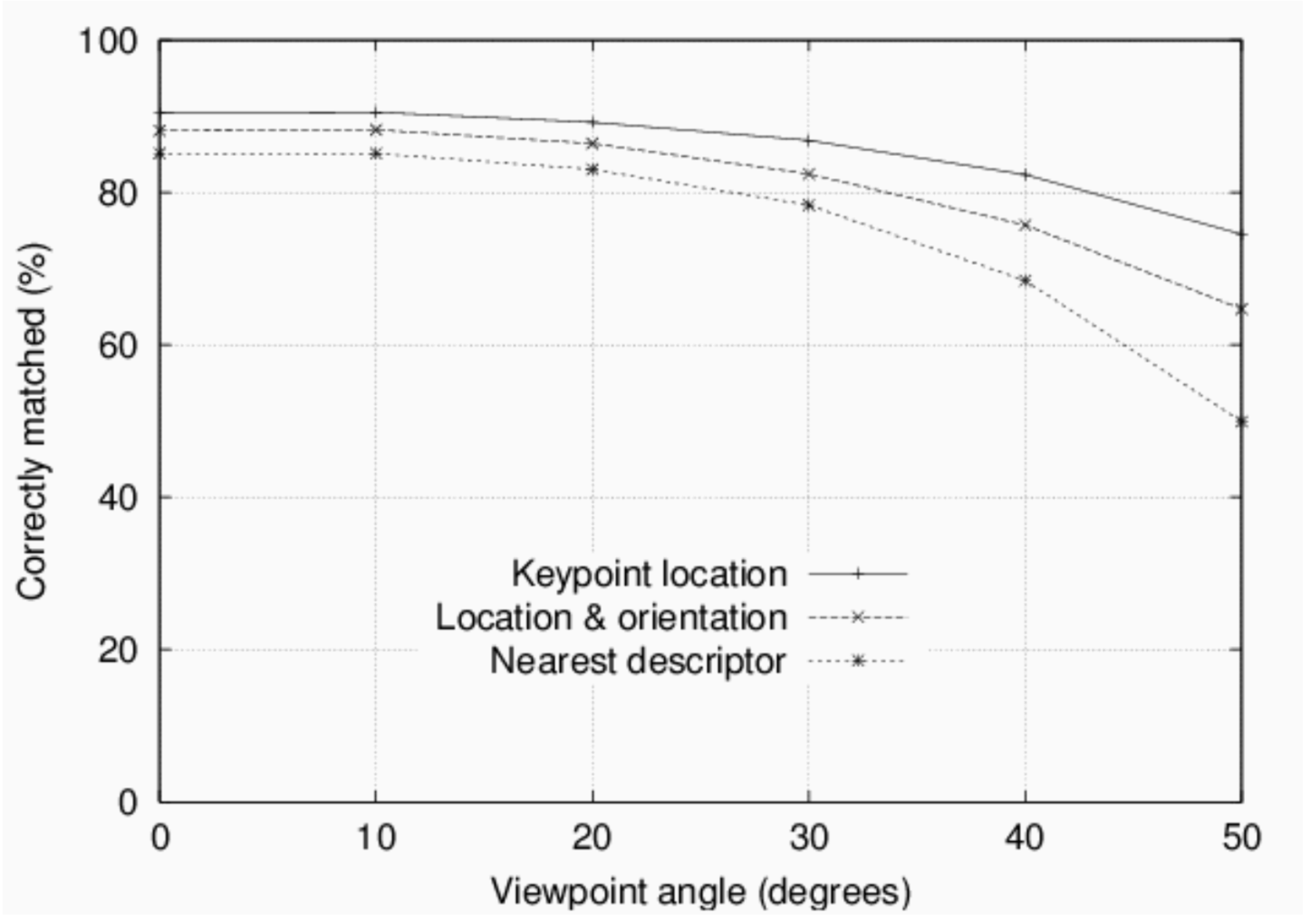
Limitation of this …

Despite all efforts this is **very** difficult …

1. If we can find exact match **80%** of the time, we can find 3 matches correctly only about **50%** of the time.

2. Image **noise**, **deformations**, will make this worse (e.g., if finding exact match drops to 50%, the probability of finding 3 exact matches will drop to 12.5%)

3. Multiple object instances will make this impossible



Fitting a Model to Noisy Data

Suppose we are **fitting a line** to a dataset that consists of 50% outliers

We can fit a line using two points

If we draw pairs of points uniformly at random, what fraction of pairs will consist entirely of ‘good’ data points (inliers)?

Fitting a Model to Noisy Data

Suppose we are **fitting a line** to a dataset that consists of 50% outliers

We can fit a line using two points

— If we draw pairs of points uniformly at random, then about 1/4 of these pairs will consist entirely of ‘good’ data points (inliers)

— We can identify these good pairs by noticing that a large collection of other points lie close to the line fitted to the pair

— A better estimate of the line can be obtained by refitting the line to the points that lie close to the line

RANSAC (**RAN**dom **SA**mple **C**onsensus)

1. Randomly choose minimal subset of data points necessary to fit model (a **sample**)

2. Points within some distance threshold, t, of model are a **consensus set**. Size of consensus set is model's **support**

3. Repeat for N samples; model with biggest support is most robust fit
   — Points within distance t of best model are inliers
   — Fit final model to all inliers

Slide Credit: Christopher Rasmussen

RANSAC (**RAN**dom **SA**mple **C**onsensus)

1. Randomly choose minimal subset of data points necessary to fit model (a **sample**)

2. Points within some distance threshold, t, of model are a **consensus set**. Size of consensus set is model's **support**

3. Repeat for N samples; model with biggest support is most robust fit
    — Points within distance t of best model are inliers
    — Fit final model to all inliers

RANSAC is very useful for variety of applications

Slide Credit: Christopher Rasmussen

RANSAC (**RAN**dom **SA**mple **C**onsensus)

1. Randomly choose minimal subset of data points necessary to fit model (a **sample**)

   Fitting a Line: 2 points

2. Points within some distance threshold, t, of model are a **consensus set**. Size of consensus set is model's **support**

3. Repeat for N samples; model with biggest support is most robust fit
   — Points within distance t of best model are inliers
   — Fit final model to all inliers

Slide Credit: Christopher Rasmussen

Example 1: Fitting a Line

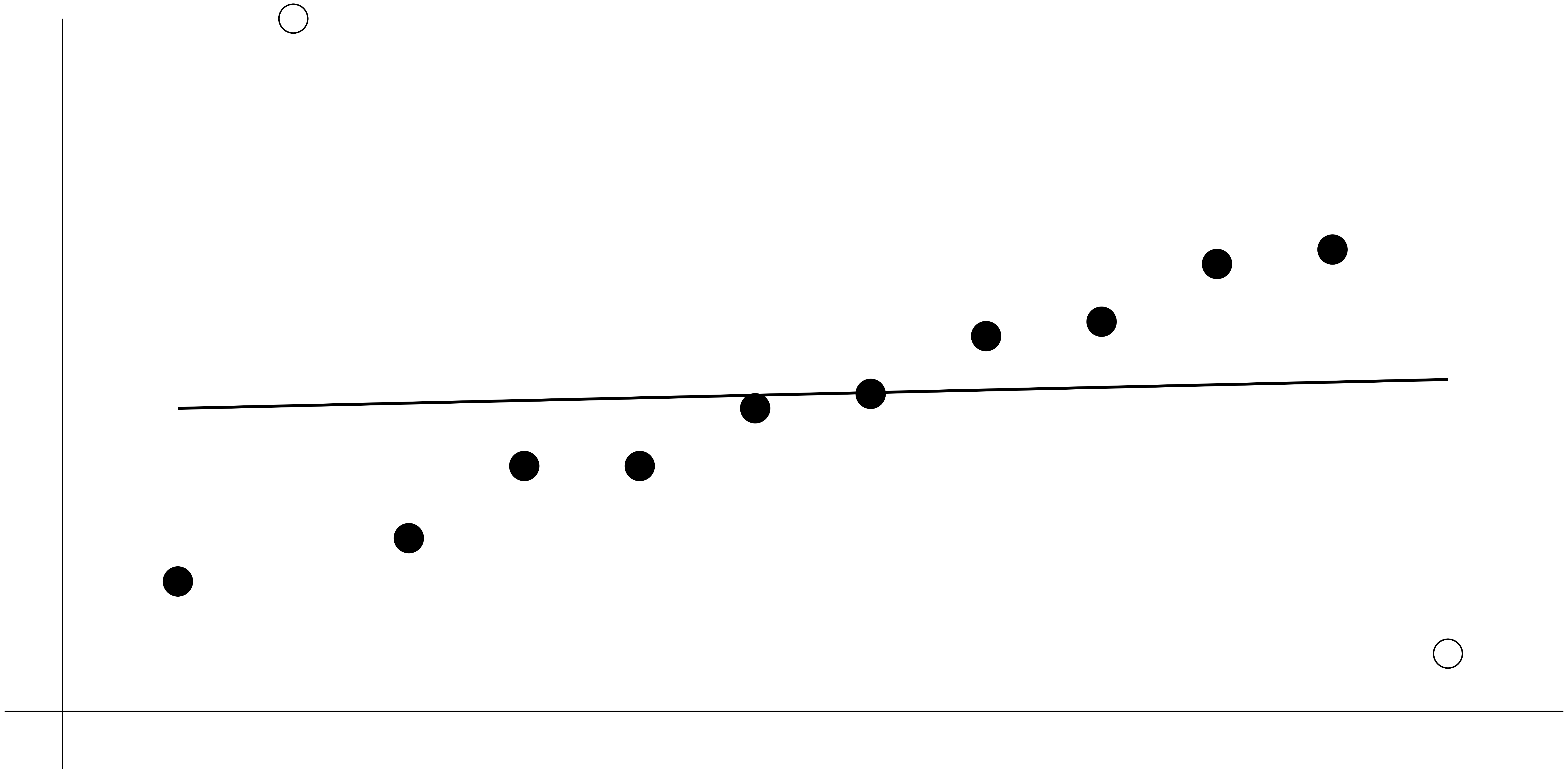


Figure Credit: Hartley & Zisserman

Example 1: Fitting a Line

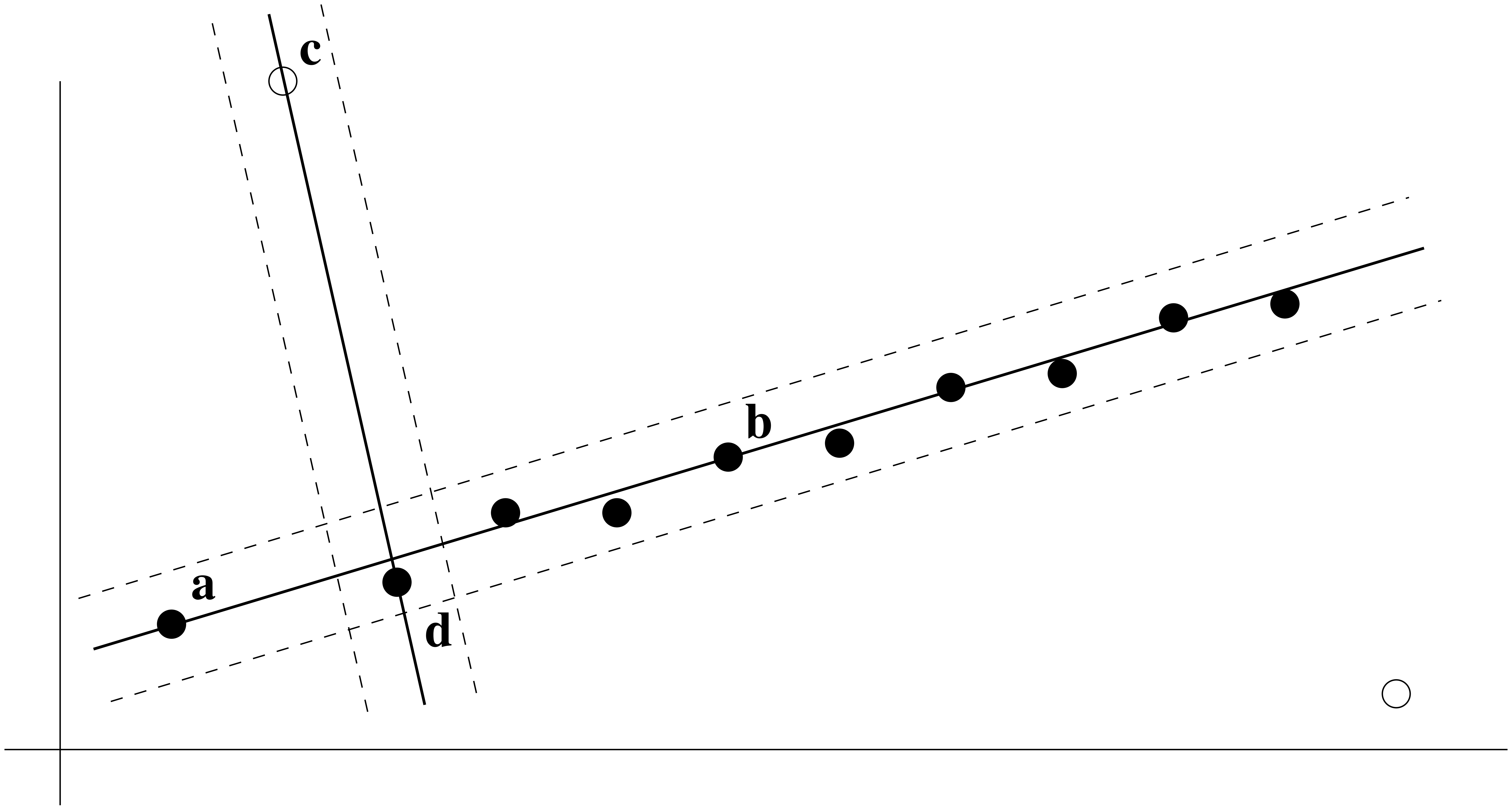


Figure Credit: Hartley & Zisserman

Example 1: Fitting a Line

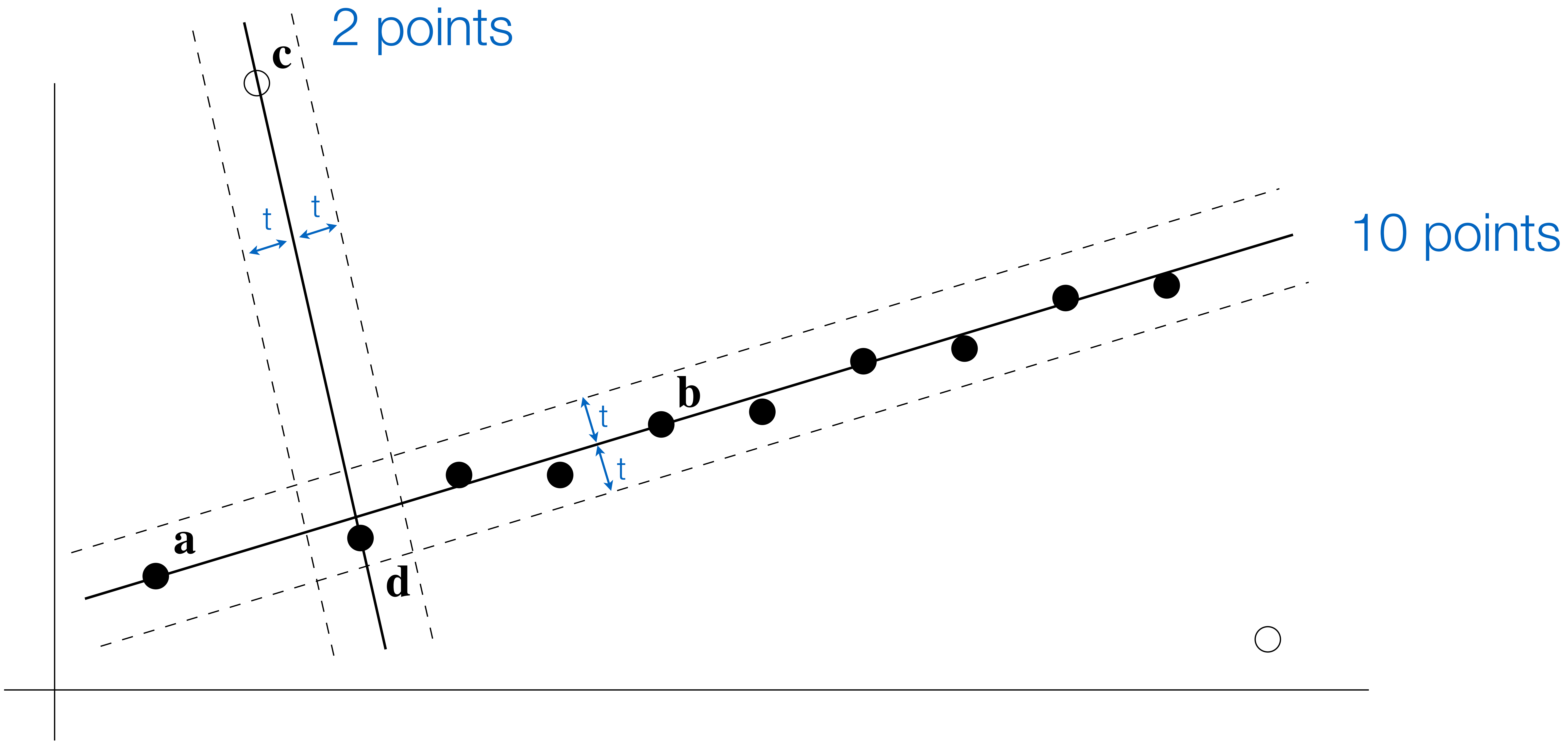


Figure Credit: Hartley & Zisserman

Algorithm 10.4

This was Algorithm 15.4 in Forsyth & Ponce (1st ed.)

Algorithm 15.4: RANSAC: fitting lines using random sample consensus

Determine:
    n — the smallest number of points required
    k — the number of iterations required
    t — the threshold used to identify a point that fits well
    d — the number of nearby points required
      to assert a model fits well
Until k iterations have occurred
    Draw a sample of n points from the data
      uniformly and at random
    Fit to that set of n points
    For each data point outside the sample
      Test the distance from the point to the line
        against t; if the distance from the point to the line
        is less than t, the point is close
    end
    If there are d or more points close to the line
      then there is a good fit. Refit the line using all
      these points.
end
Use the best fit from this collection, using the
  fitting error as a criterion

RANSAC: Fitting Lines Using Random Sample Consensus

RANSAC: How many samples?

Let ω be the fraction of inliers (i.e., points on line)

Let n be the number of points needed to define hypothesis
  ($n = 2$ for a line in the plane)

Suppose k samples are chosen

The probability that a single sample of n points is correct (all inliers) is

RANSAC: How many samples?

Let ω be the fraction of inliers (i.e., points on line)

Let n be the number of points needed to define hypothesis

$(n = 2$ for a line in the plane)

Suppose k samples are chosen

The probability that a single sample of n points is correct (all inliers) is

$$\omega^n$$

The probability that all k samples fail is

RANSAC: How many samples?

Let ω be the fraction of inliers (i.e., points on line)

Let n be the number of points needed to define hypothesis

$(n = 2$ for a line in the plane)

Suppose k samples are chosen

The probability that a single sample of n points is correct (all inliers) is

$$\omega^n$$

The probability that all k samples fail is

$$(1 - \omega^n)^k$$

Choose k large enough (to keep this below a target failure rate)

RANSAC: *k* Samples Chosen (p = 0.99)

Sample size	Proportion of outliers						
n	**5%**	**10%**	**20%**	**25%**	**30%**	**40%**	**50%**
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Figure Credit: Hartley & Zisserman

After RANSAC

RANSAC divides data into inliers and outliers and yields estimate computed from minimal set of inliers

Improve this initial estimate with estimation over all inliers (e.g., with standard least-squares minimization)

But this may change inliers, so alternate fitting with re-classification as inlier/outlier

Example 2: Fitting a Line

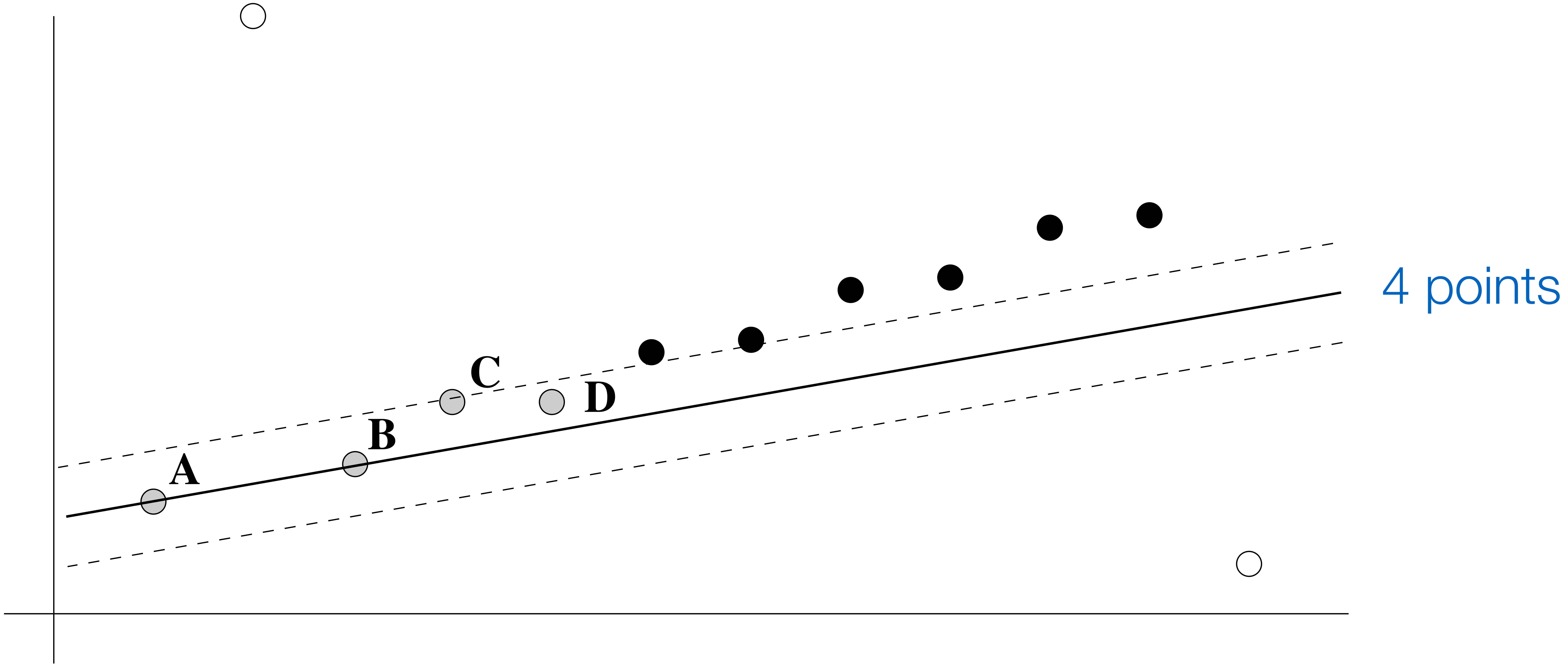


Figure Credit: Hartley & Zisserman

Example 2: Fitting a Line

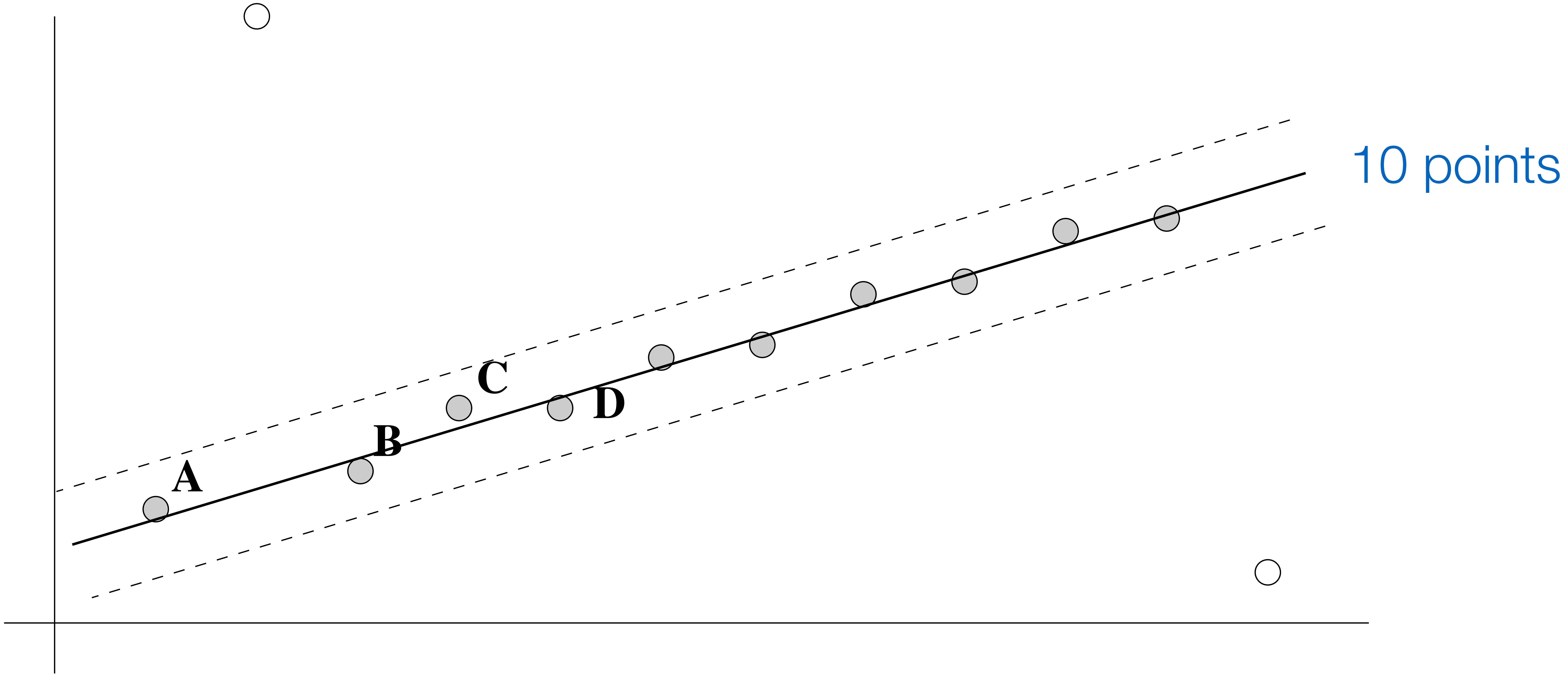


Figure Credit: Hartley & Zisserman

Example 3: Automatic Matching of Images

— How to get correct correspondences without human intervention?

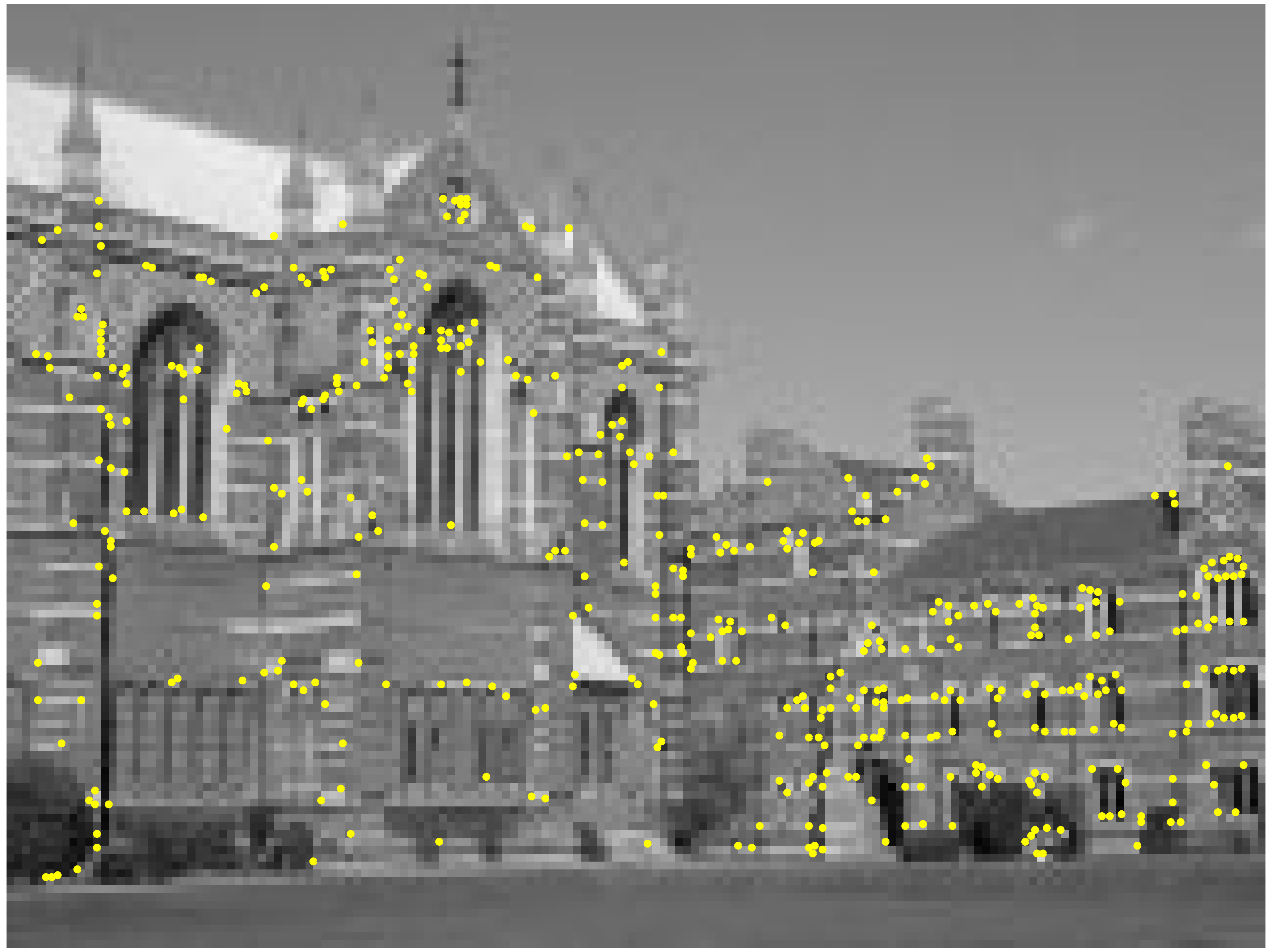
— Can be used for image stitching or automatic determination of epipolar geometry

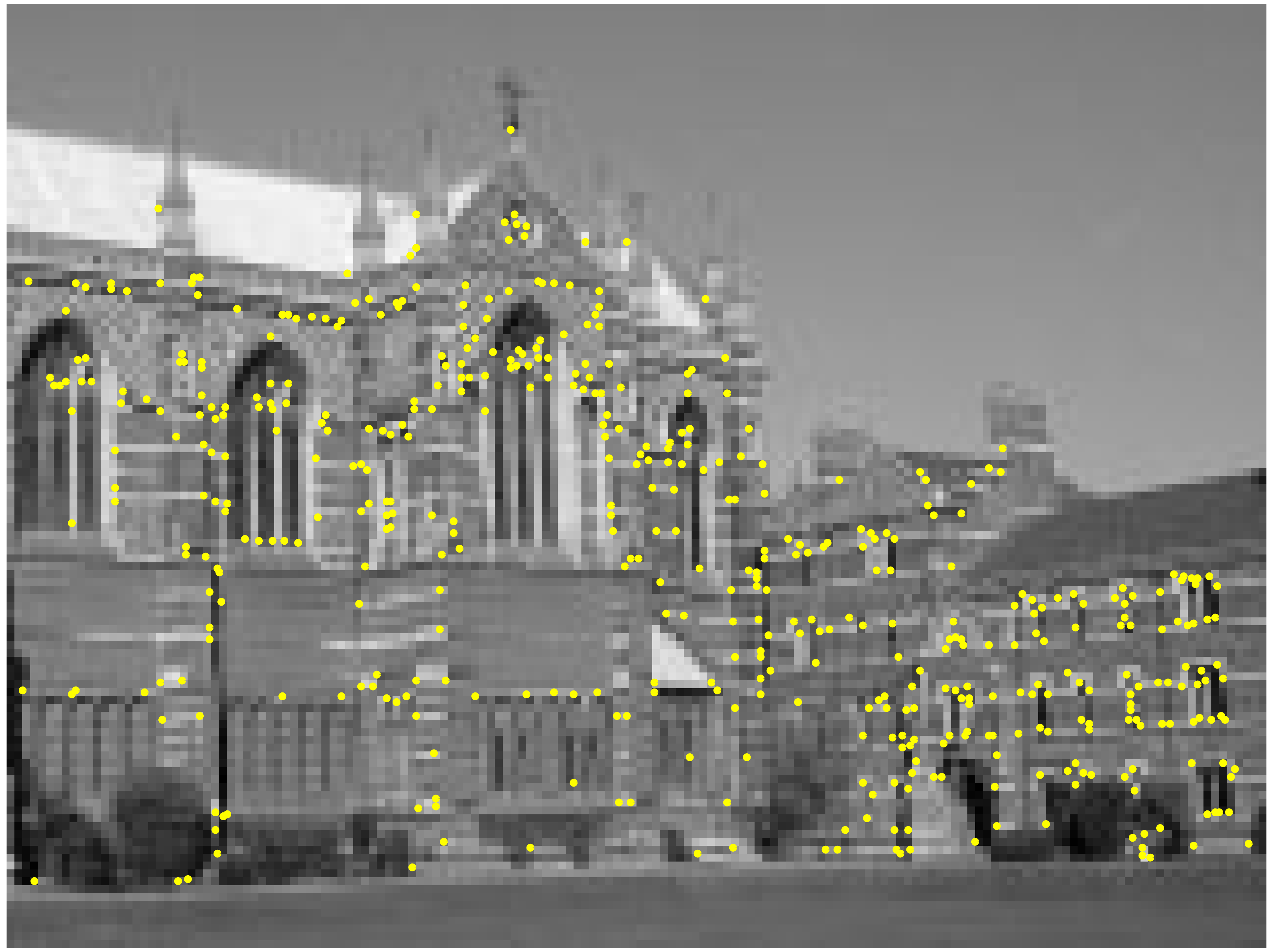
Figure Credit: Hartley & Zisserman

Example 3: Feature Extraction

— Find features in pair of images using Harris corner detector

— Assumes images are roughly the same scale



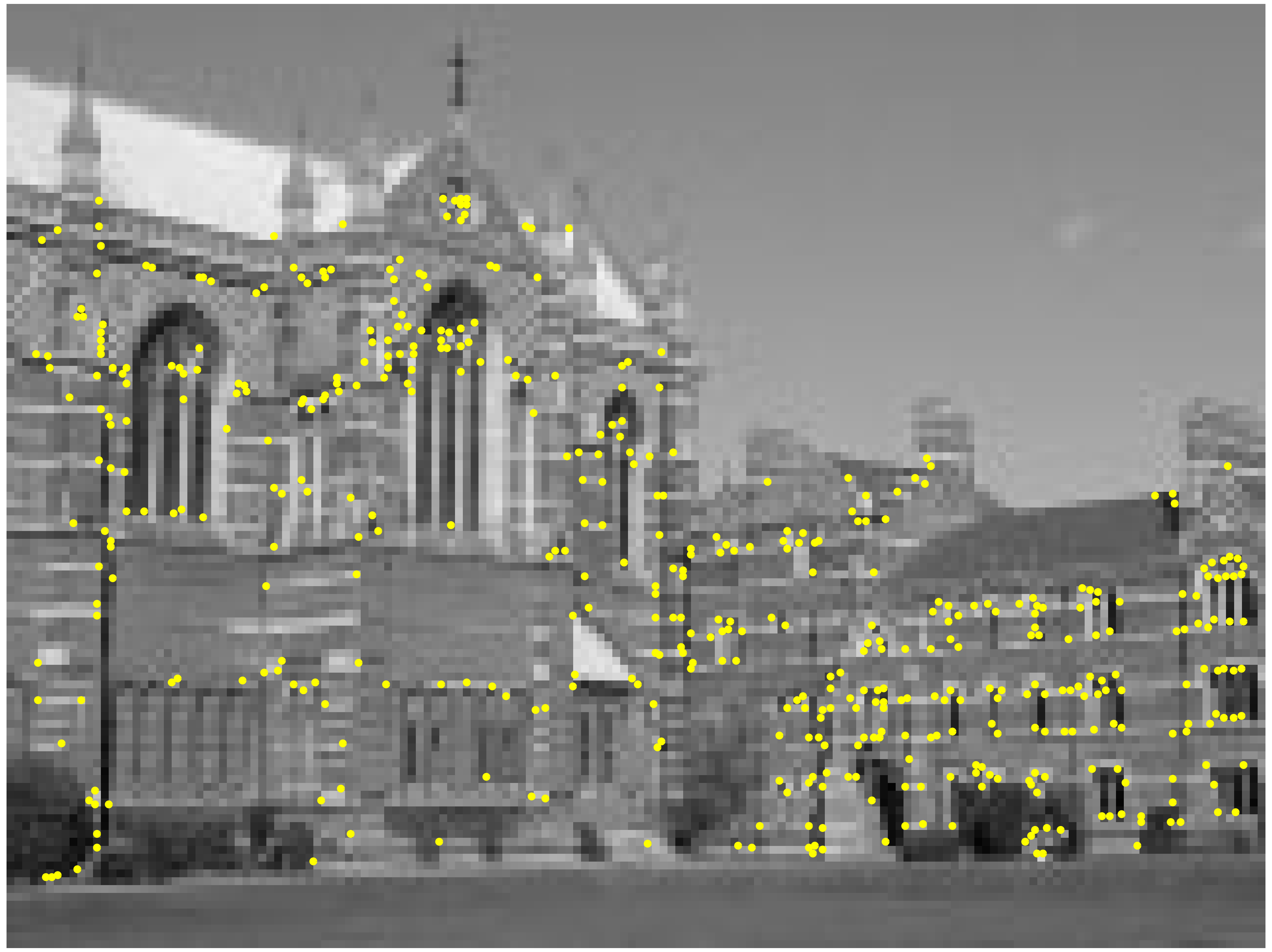


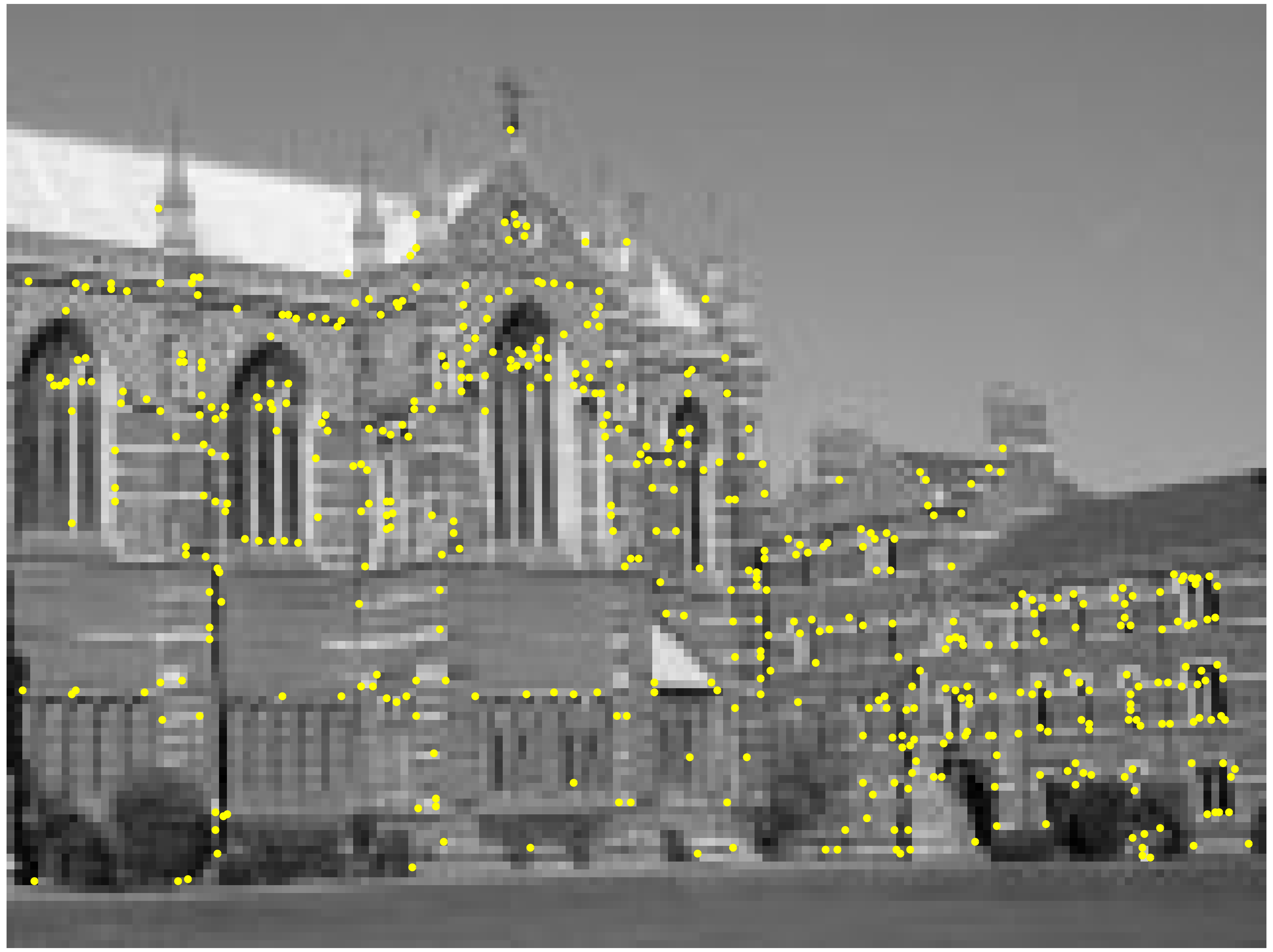
≈ 500 corner features found in each image

Figure Credit: Hartley & Zisserman

Example 3: Finding Feature Matches

Select best match over threshold within a square search window (here ±320 pixels) using SSD or (normalized) cross-correlation for small patch around the corner





≈ 500 corner features found in each image

Figure Credit: Hartley & Zisserman

Example 3: Initial Match Hypothesis

268 matched features (over SSD threshold) superimposed on left image (pointing to locations of corresponding feature in right image)

Figure Credit: Hartley & Zisserman

Example 3: Outliers & Inliers after RANSAC

— *n* is 4 for this problem (a homography relating 2 images)

— Assume up to 50% outliers

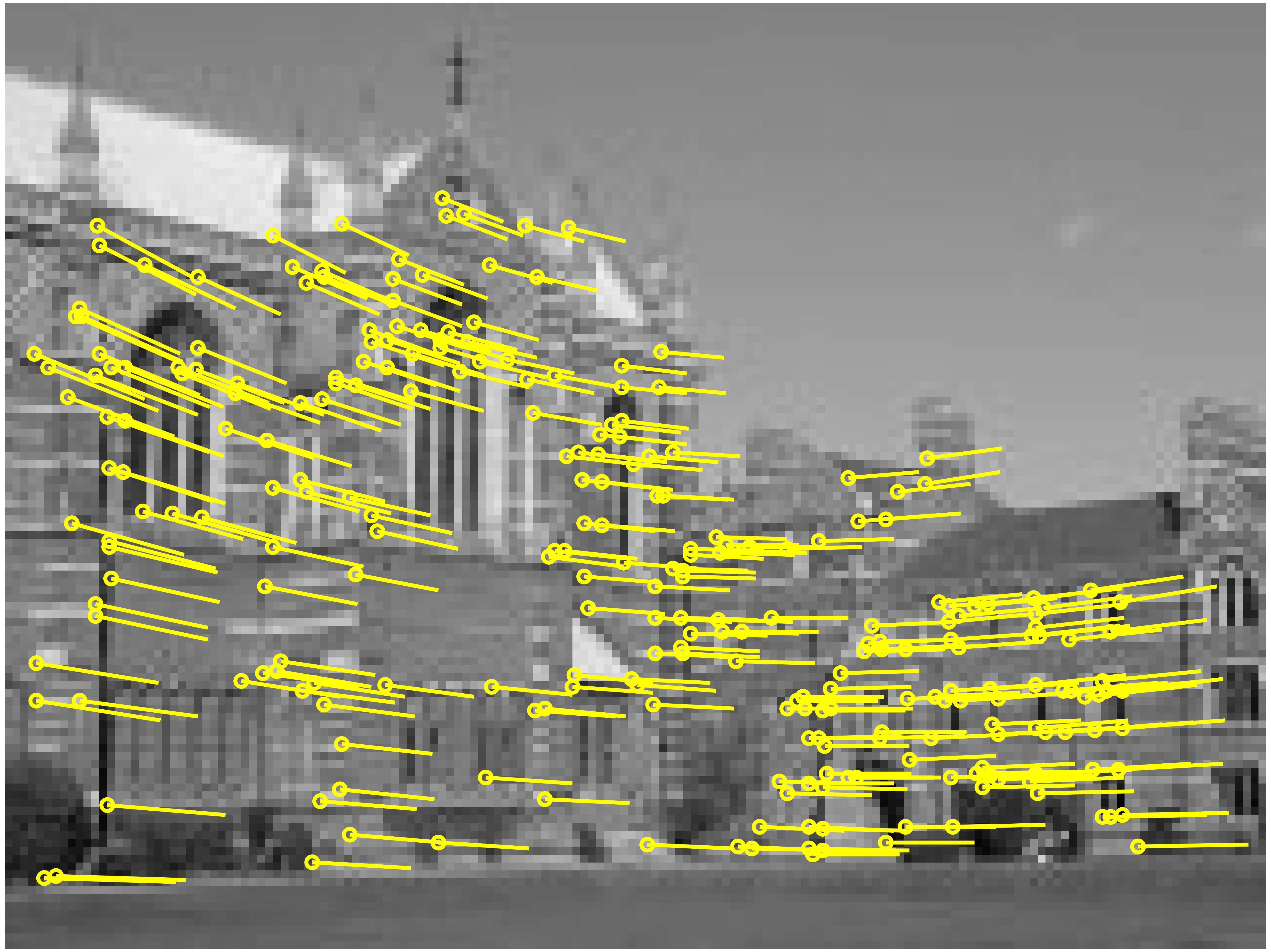
— 43 samples used with t = 1.25 pixels

117 **outliers**

151 **inliers**

Figure Credit: Hartley & Zisserman

Example 3: Final Matches



final set of 262 matches

Figure Credit: Hartley & Zisserman

Discussion of RANSAC

Advantages:

— General method suited for a wide range of model fitting problems

— Easy to implement and easy to calculate its failure rate

Disadvantages:

— Only handles a moderate percentage of outliers without cost blowing up

— Many real problems have high rate of outliers (but sometimes selective choice of random subsets can help)

The Hough transform can handle high percentage of outliers