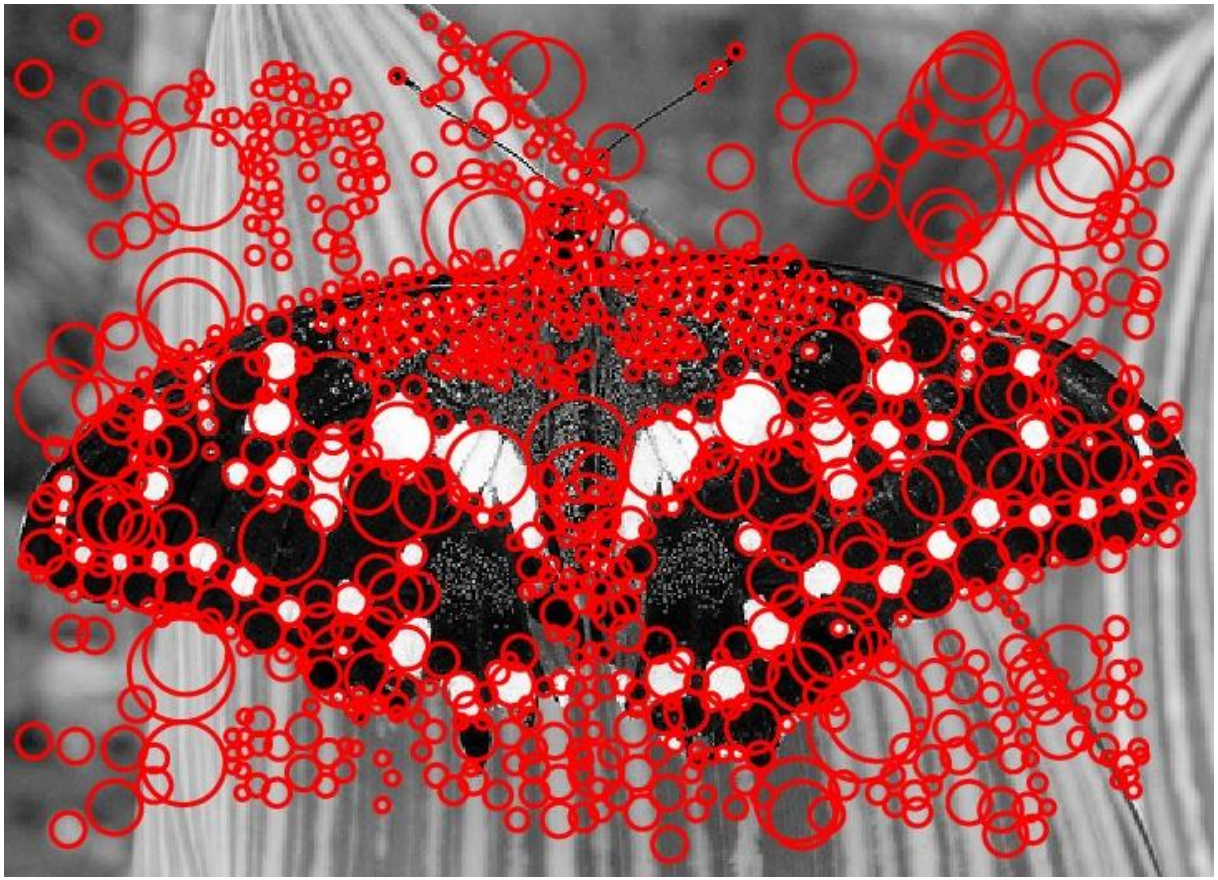
SIFT Algorithm and Applications

Dr. Mohamed Waleed Fakhr

2023

Scale Invariant Feature Transform (SIFT) Local Descriptors

- SIFT has become one of the most important local image features.
- It is used in image Stitching and panorama creation, image registration, image retrieval, image classification, object detection, etc.

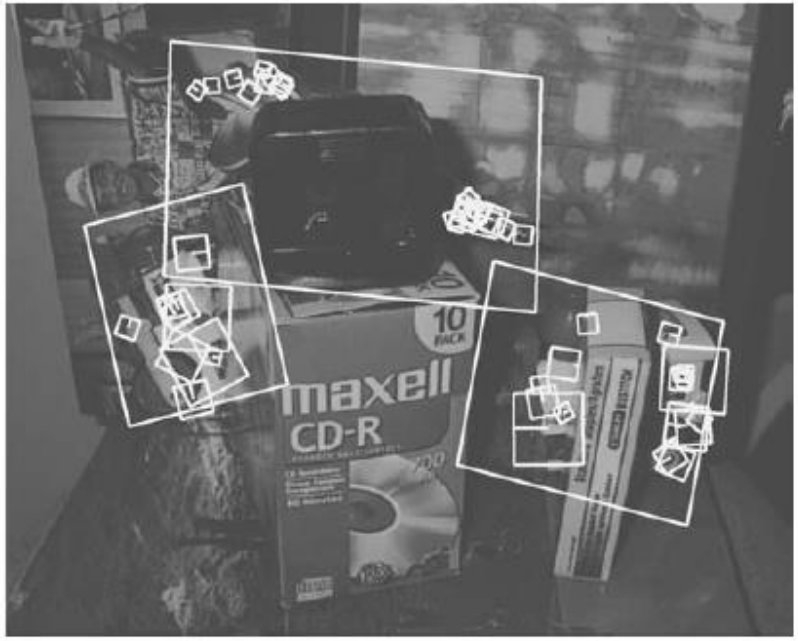


Step1: find interest points using the DoG approach which detects interest points

Step2: Extract a 128-dimension vector at each interest point


Good Matches & Object detection

Example: Object Recognition



SIFT is extremely powerful for object instance recognition, especially for well-textured objects

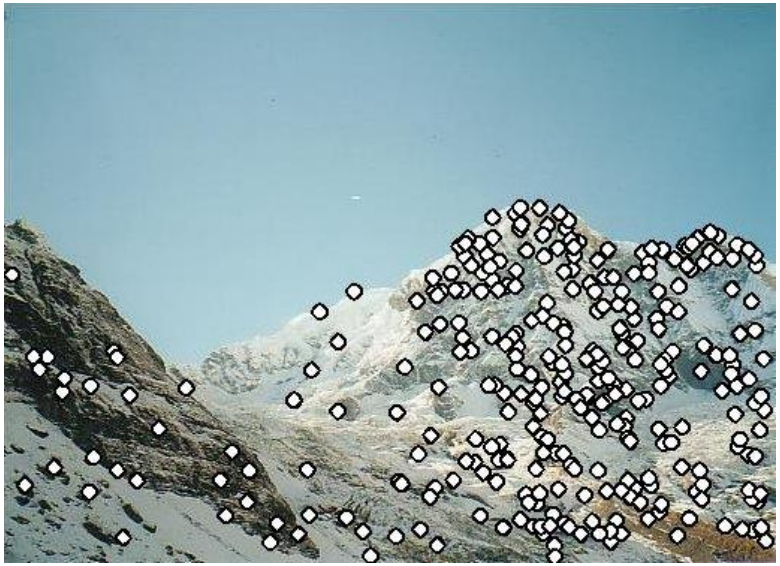
Lowe, IJCV04

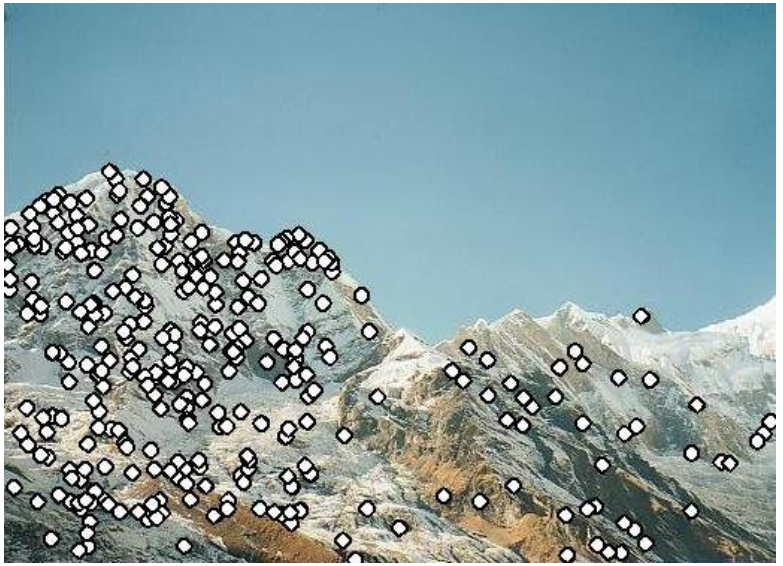
panorama?

- We need to match (align) images

Matching with Features

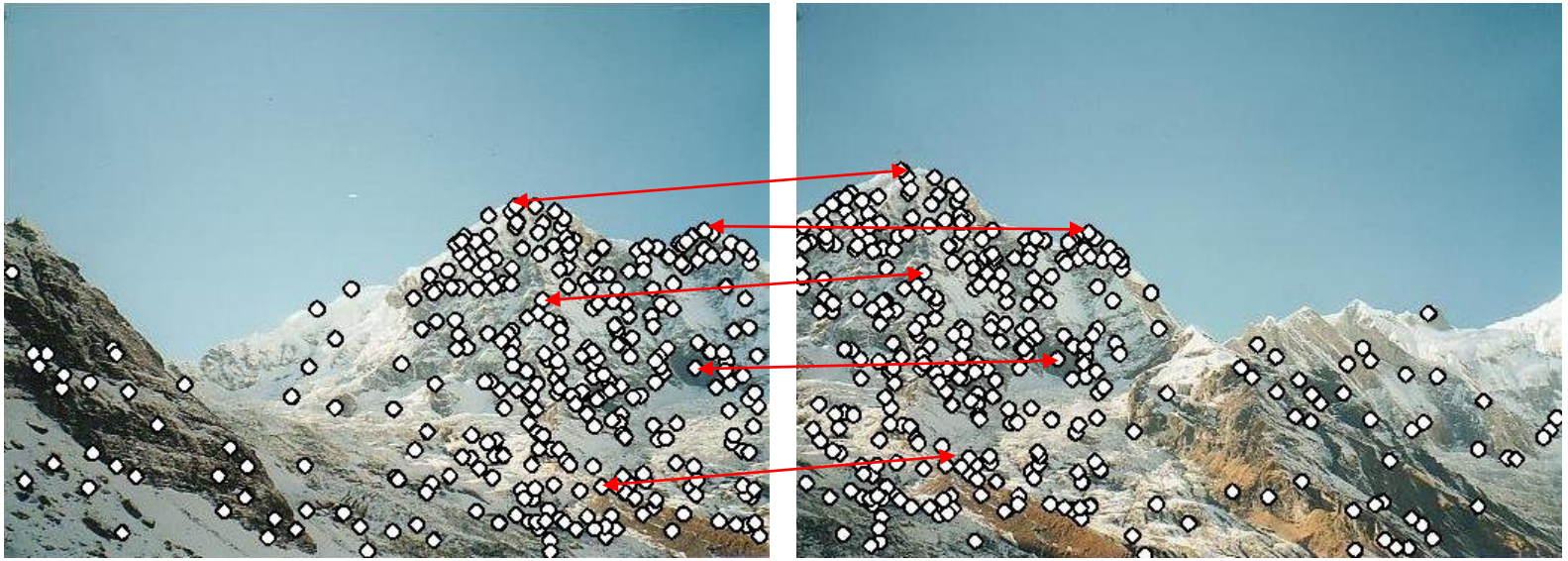
- Detect feature points in both images





Matching with Features

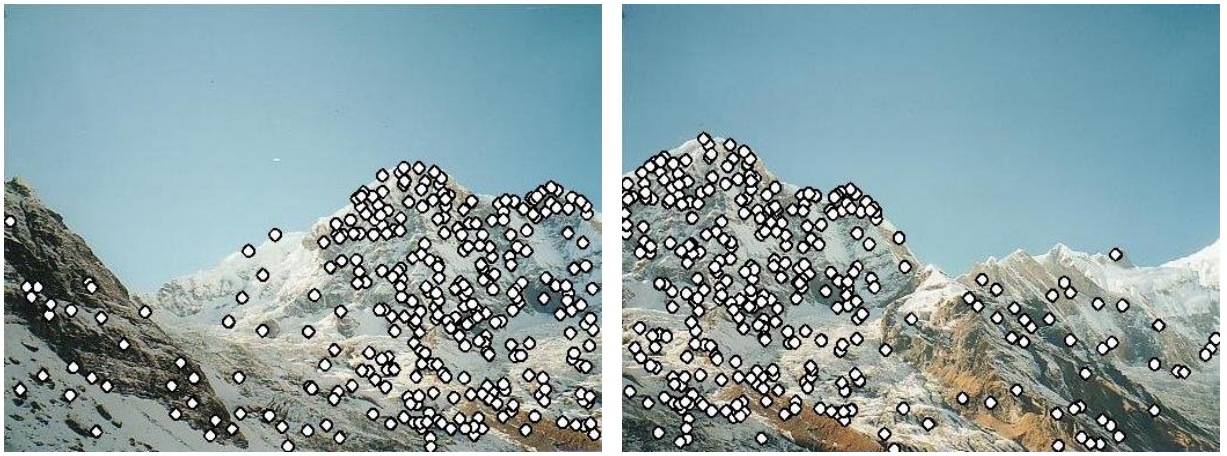
- Detect feature points in both images
- Find corresponding pairs



Matching with Features

- Detect feature points in both images
- Find corresponding pairs
- Use these matching pairs to align images - the required mapping is called a homography.

Characteristics of good features



- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature has a distinctive description
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion

Applications

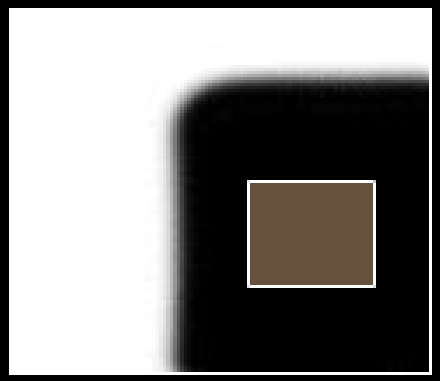
Feature points are used for:

- Motion tracking
- Image alignment
- 3D reconstruction
- Object recognition
- Indexing and database retrieval
- Robot navigation

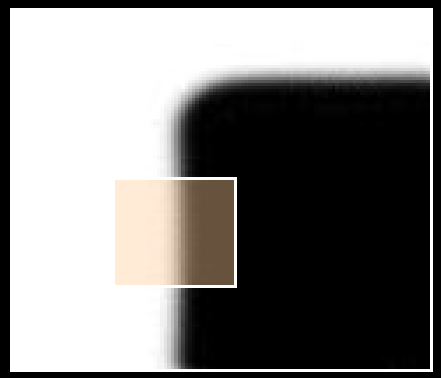
How we get the interest points?

- It is all about finding Local Extrema in the image → Corners → Blobs

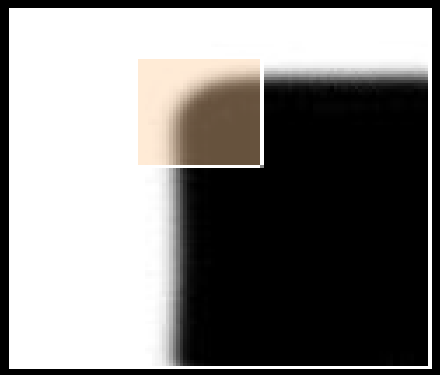
Corner Detection: Basic Idea



"flat" region:
no change in
all directions



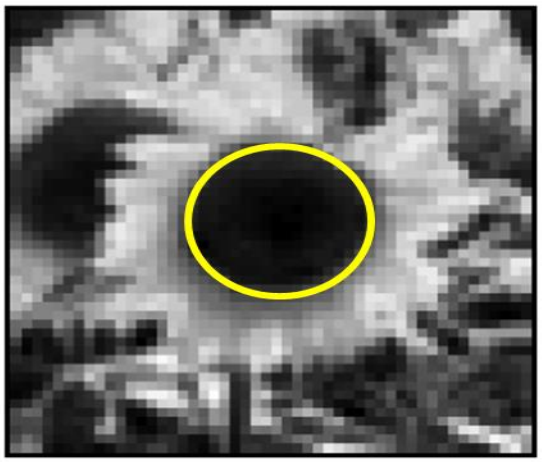
"edge":
no change
along the edge
direction

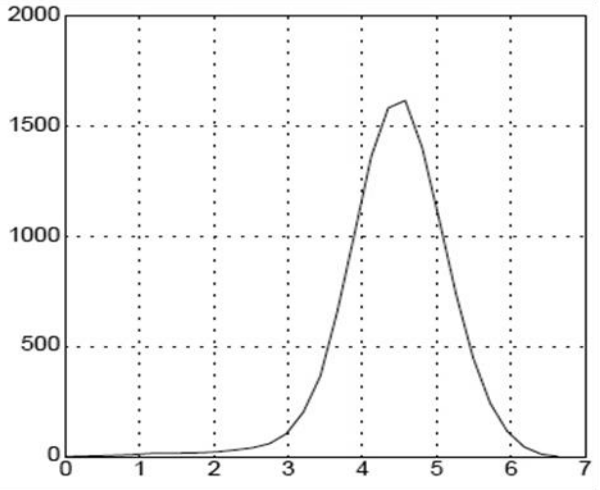


"corner":
significant change
in all directions
with small shift

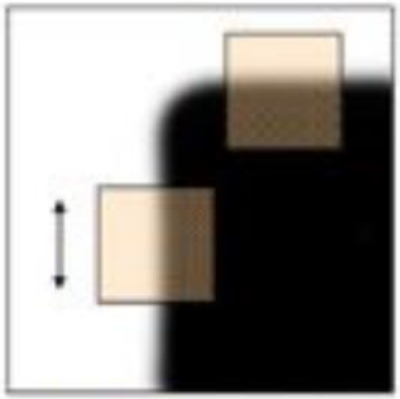
Source: A. Efros

A blob:





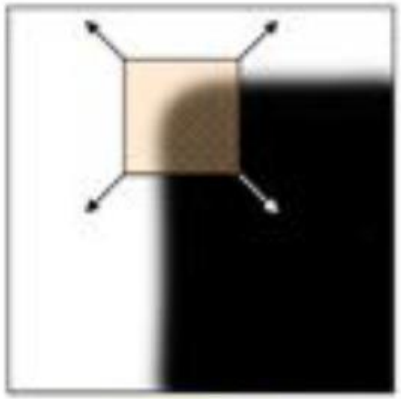
Interest Points: Corners



"edge":

$\lambda_1 >> \lambda_2$

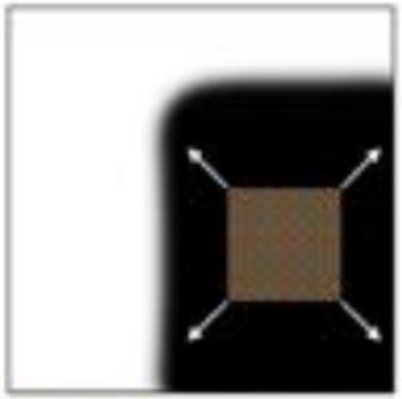
$\lambda_2 >> \lambda_1$



"corner":

λ_1 and λ_2 are large,

$\lambda_1 \sim \lambda_2$;



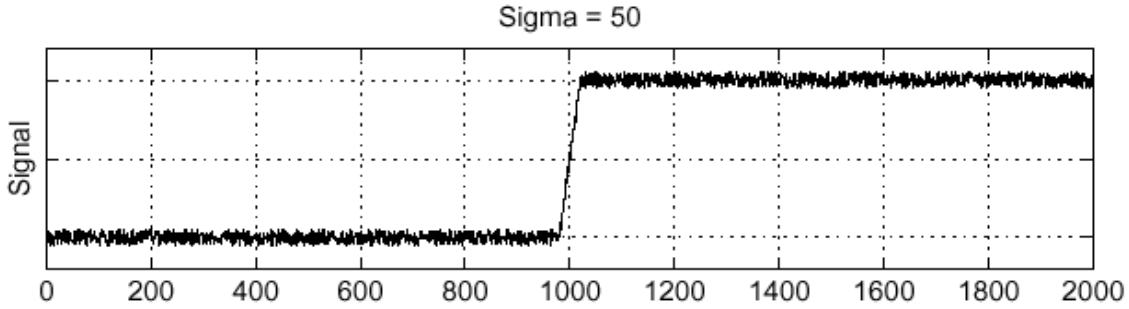
"flat" region

λ_1 and λ_2 are small;

[Source: K. Grauman, slide credit: R. Urtasun]

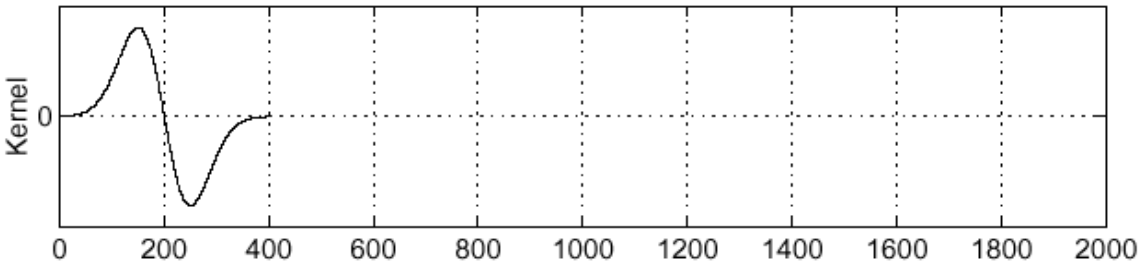
Recall: Edge detection

f



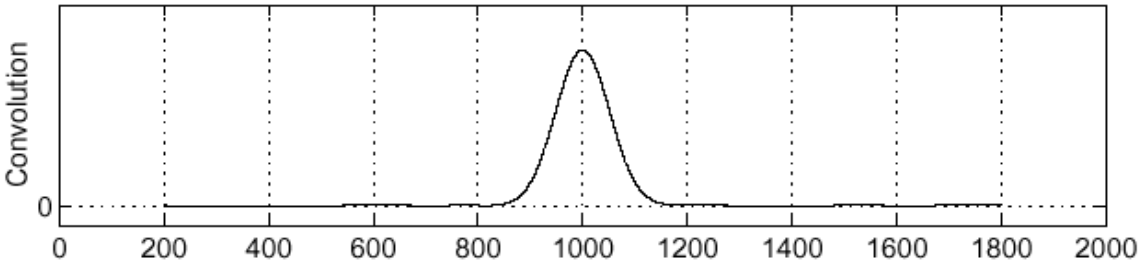
Edge

$\frac{d}{dx} g$



Derivative of Gaussian

$f * \frac{d}{dx} g$

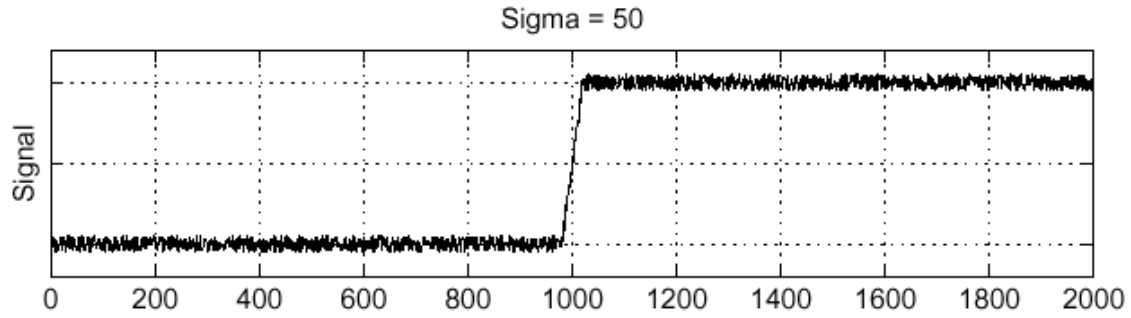


Edge = maximum of derivative

Source: S. Seitz

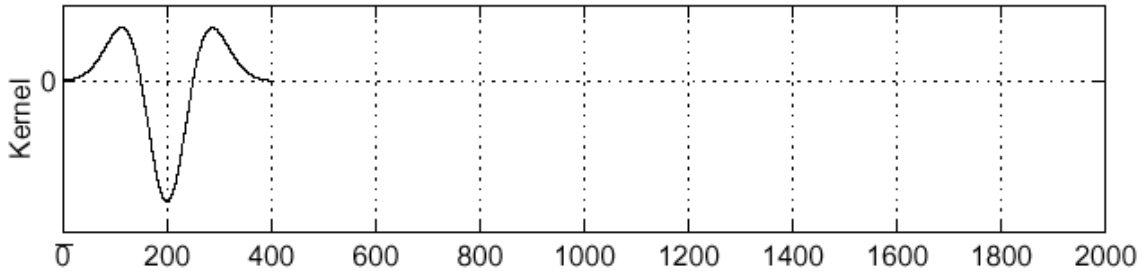
Edge detection, Take 2

f



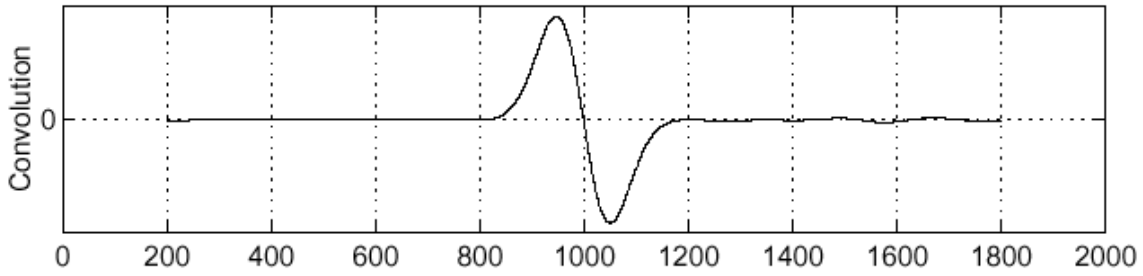
Edge

$$\frac{d^2}{dx^2} g$$



Second derivative of Gaussian (Laplacian)

$$f * \frac{d^2}{dx^2} g$$

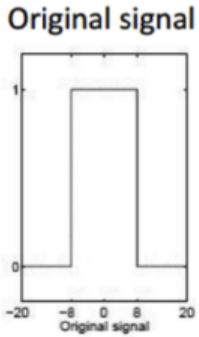


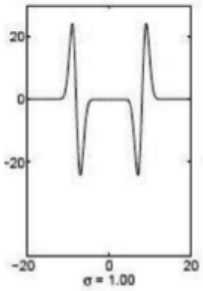
Edge = zero crossing of second derivative

Source: S. Seitz

Blob Detection – Laplacian of Gaussian

- It can be used for 2D blob detection! How?

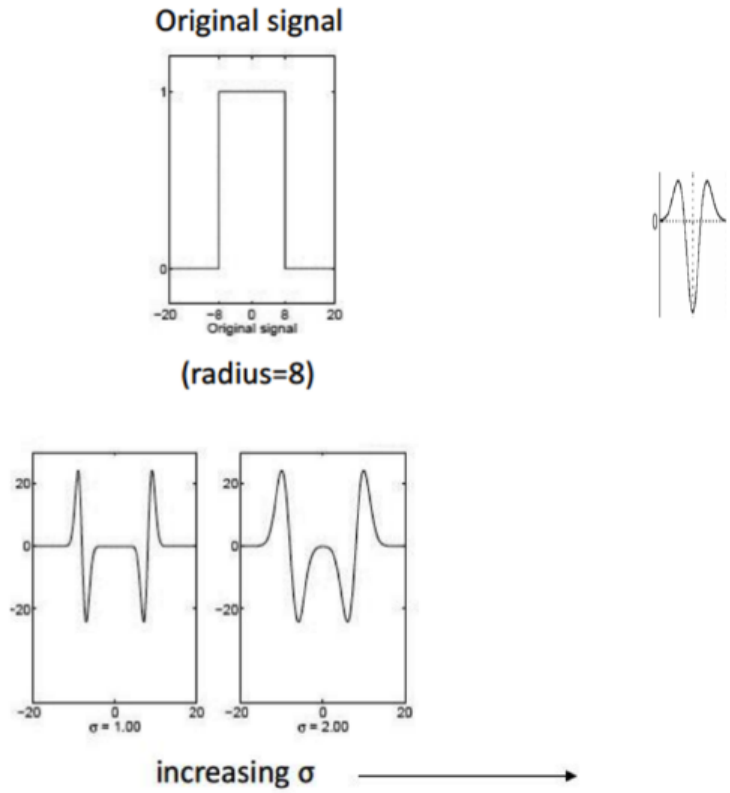




[Source: F. Flores-Mangas]

Blob Detection – Laplacian of Gaussian

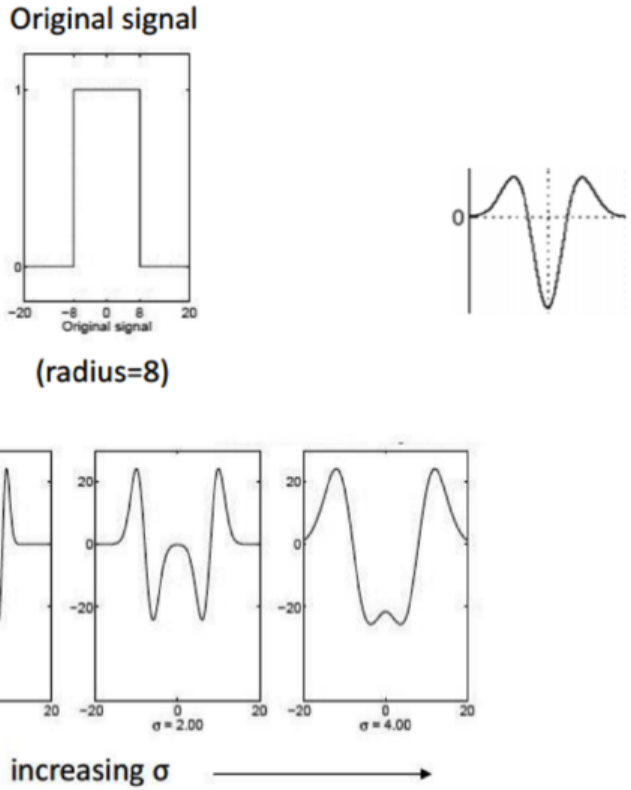
- It can be used for 2D blob detection! How?



[Source: F. Flores-Mangas]

Blob Detection – Laplacian of Gaussian

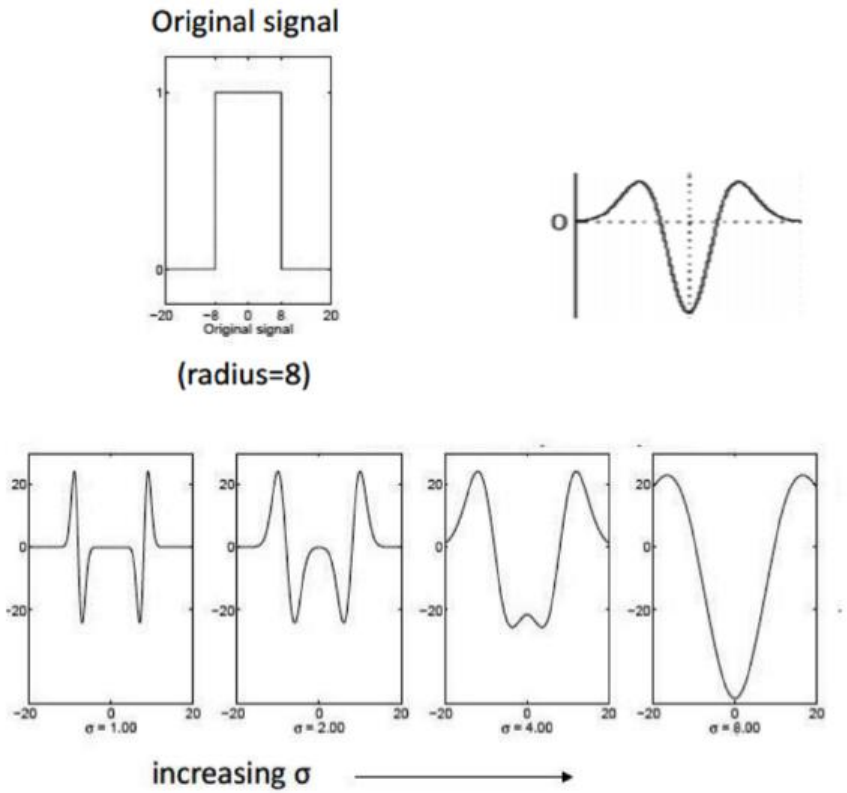
- It can be used for 2D blob detection! How?



[Source: F. Flores-Mangas]

Blob Detection – Laplacian of Gaussian

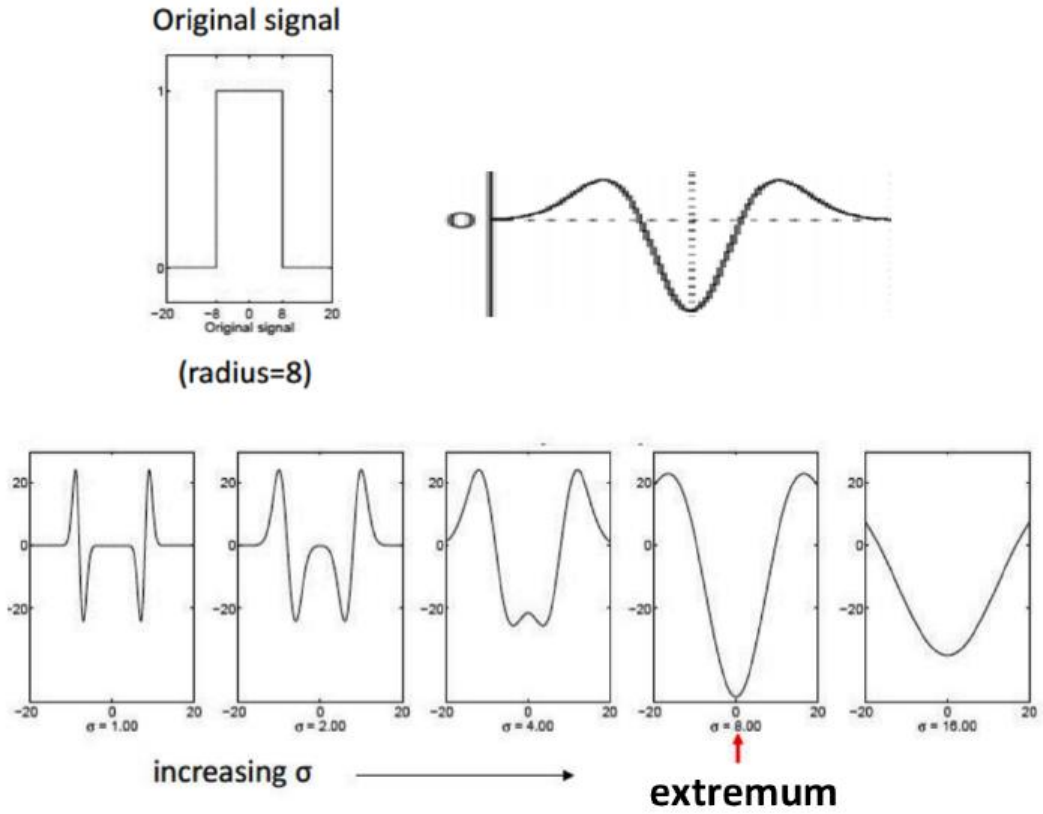
- It can be used for 2D blob detection! How?



[Source: F. Flores-Mangas]

Blob Detection – Laplacian of Gaussian

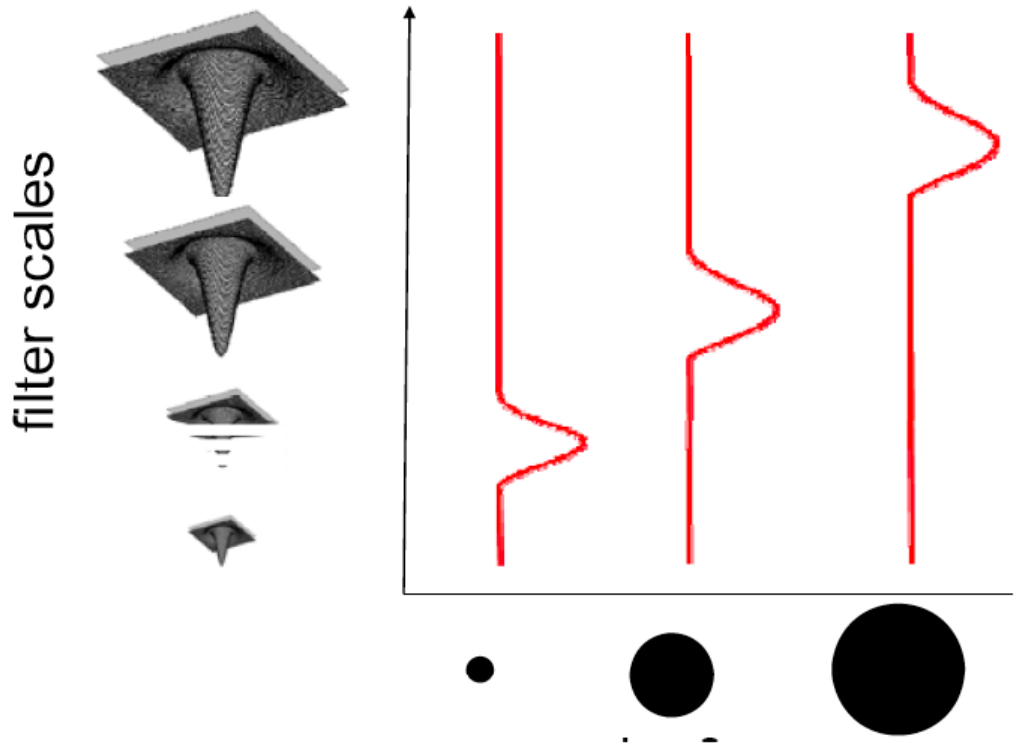
- It can be used for 2D blob detection! How?



[Source: F. Flores-Mangas]

Blob Detection in 2D: Scale Selection

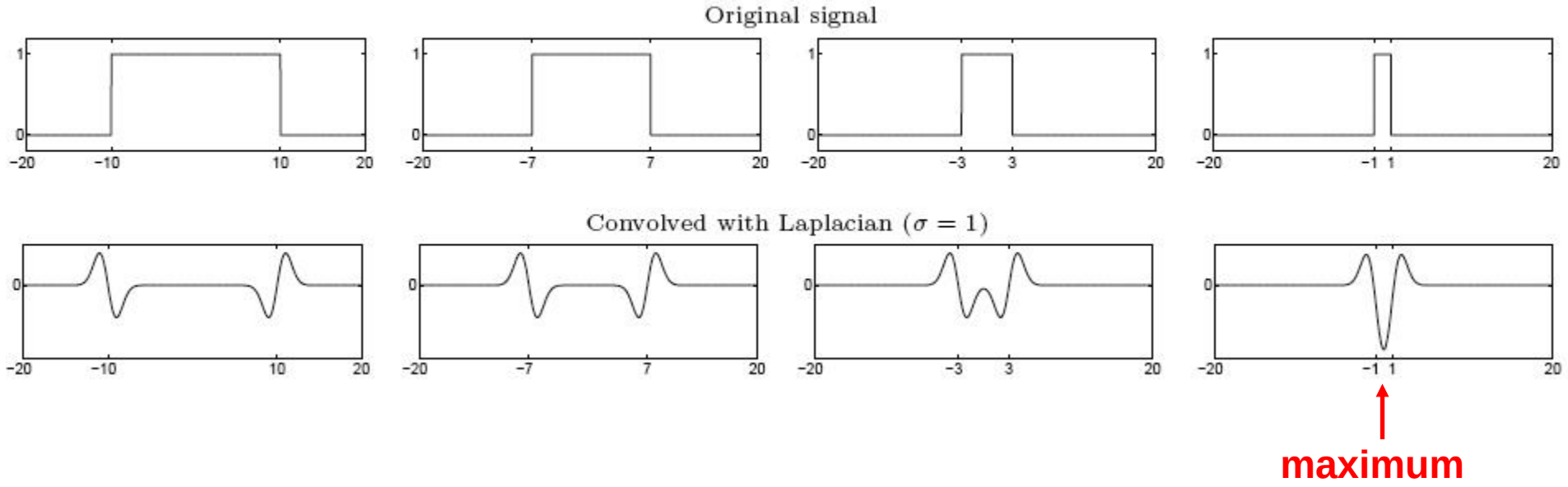
Laplacian of Gaussian = "blob" detector



[Source: B. Leibe, slide credit: R. Urtasun]

From edges to blobs

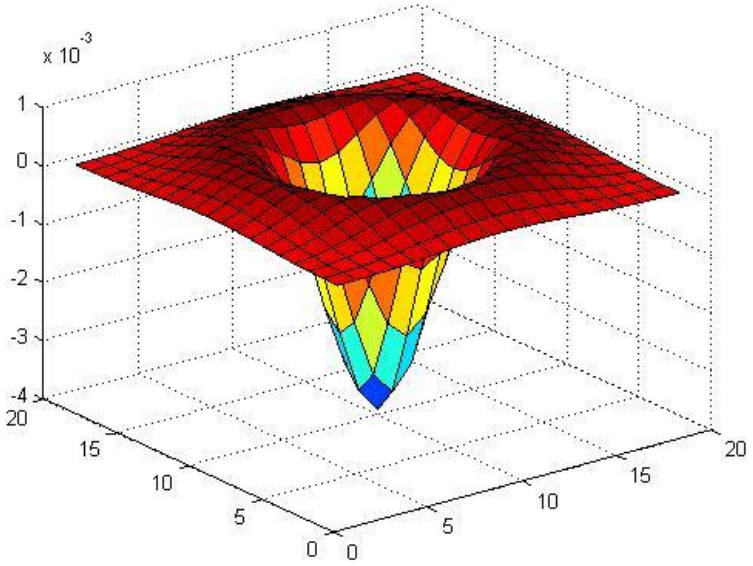
- Edge = ripple
- Blob = superposition of two ripples

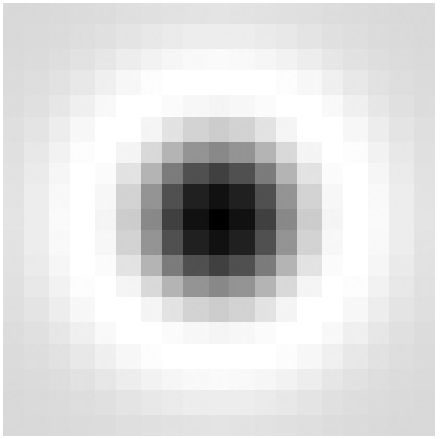


Spatial selection: the magnitude of the Laplacian response will achieve a maximum at the center of the blob, provided the scale of the Laplacian is "matched" to the scale of the blob

Blob detection in 2D

Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

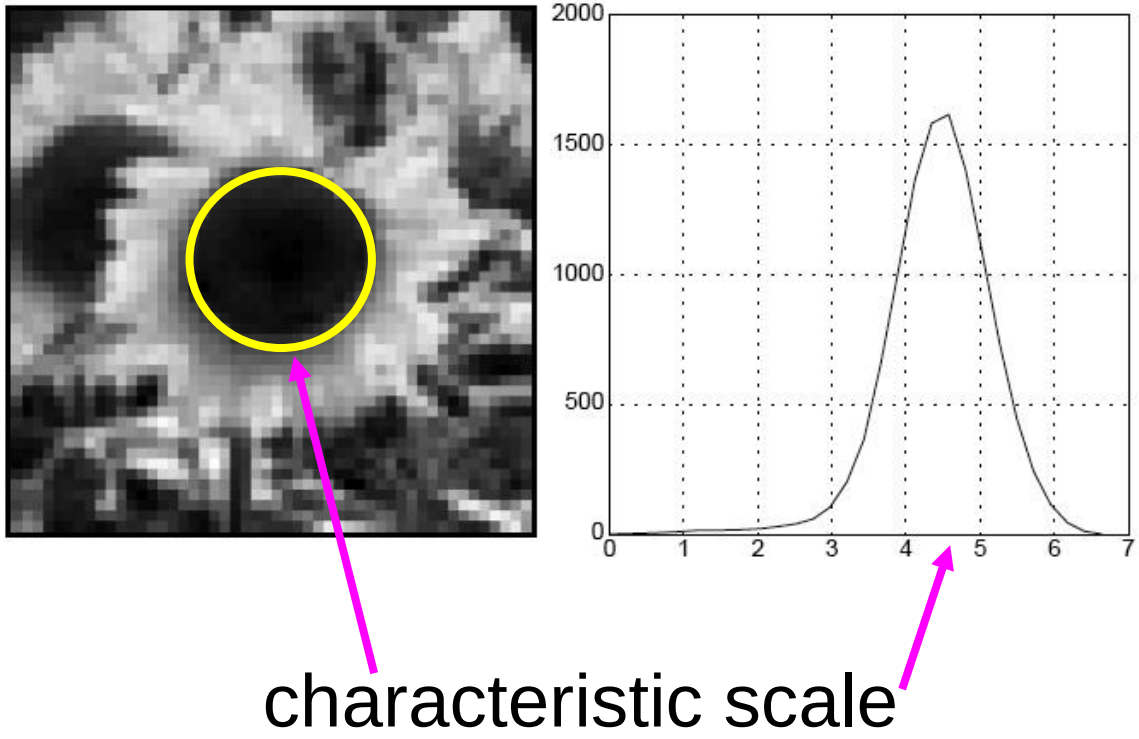




$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} = \frac{1}{\pi\sigma^4}\left(1 - \frac{x^2 + y^2}{2\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Characteristic scale

- We define the characteristic scale of a blob as the scale that produces peak of Laplacian response in the blob center



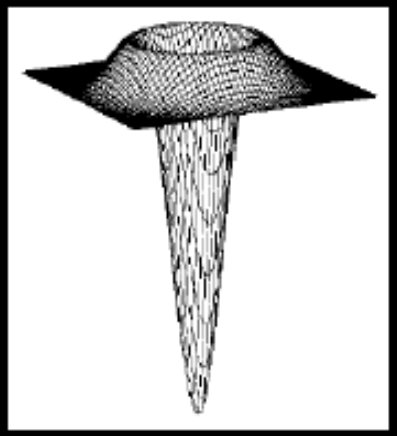
T. Lindeberg (1998). "Feature detection with automatic scale selection." *International Journal of Computer Vision* **30** (2): pp 77--116.

Scale Invariant Detection

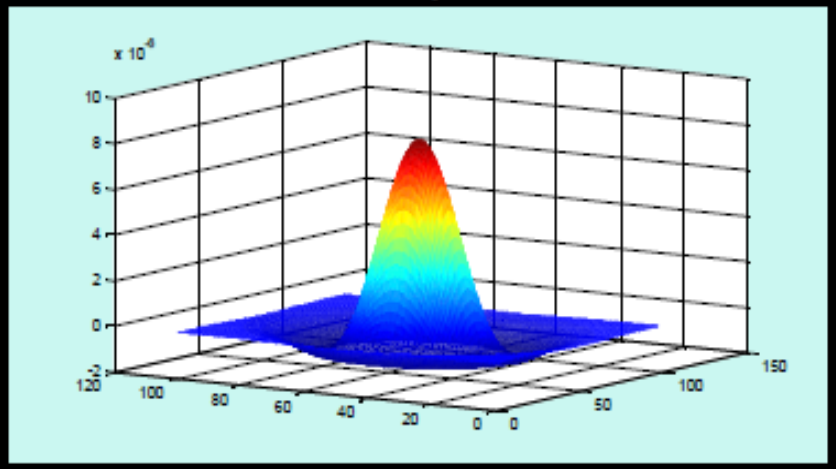
Function is just application of a kernel: $f = \mathrm{Kernel} * \mathrm{Image}$

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

(Laplacian of Gaussian - LoG)



Laplacian of Gaussian



$$\nabla^2 h_\sigma(u, v)$$

Scale Invariant Detection

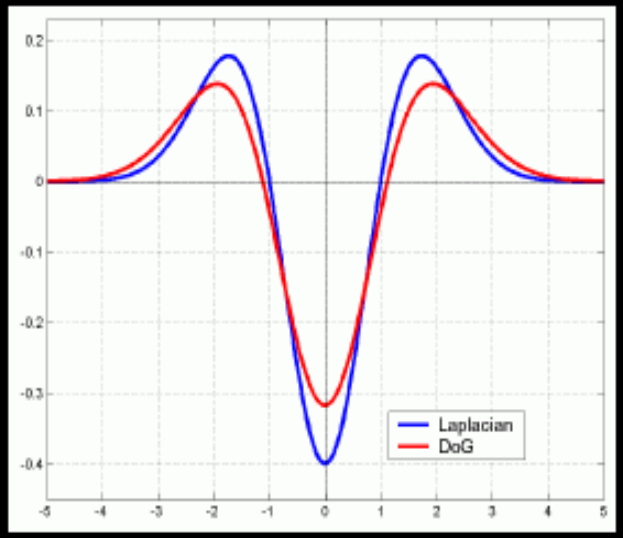
Function is just application of a kernel: $f = \text{Kernel} * \text{Image}$

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

(Laplacian of Gaussian - LoG)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)

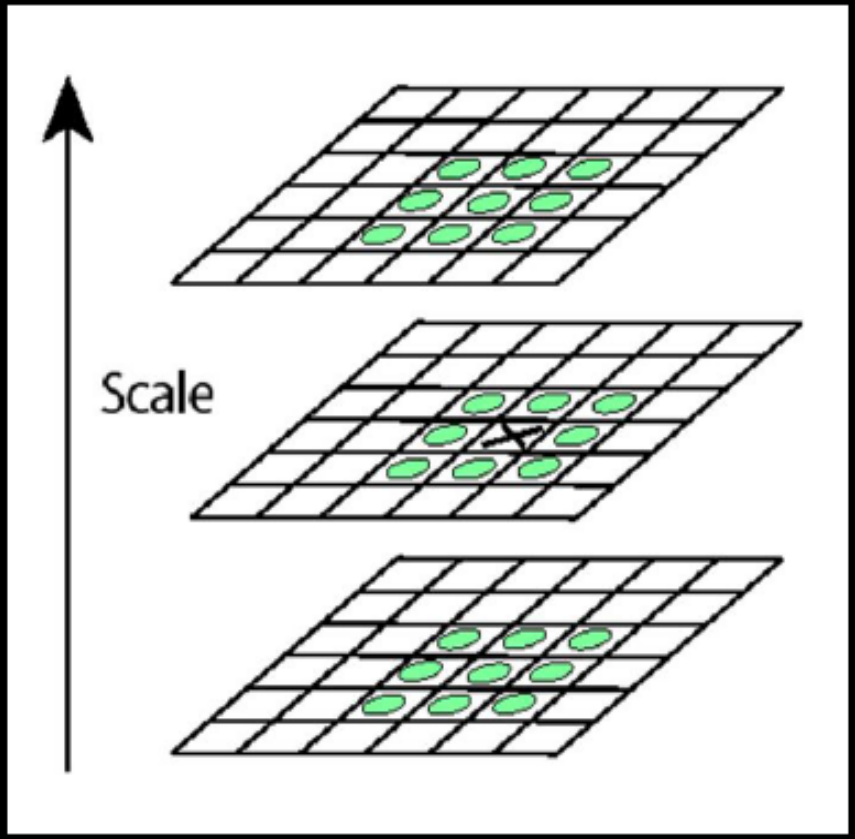


Note: both kernels are invariant to *scale* and *rotation*

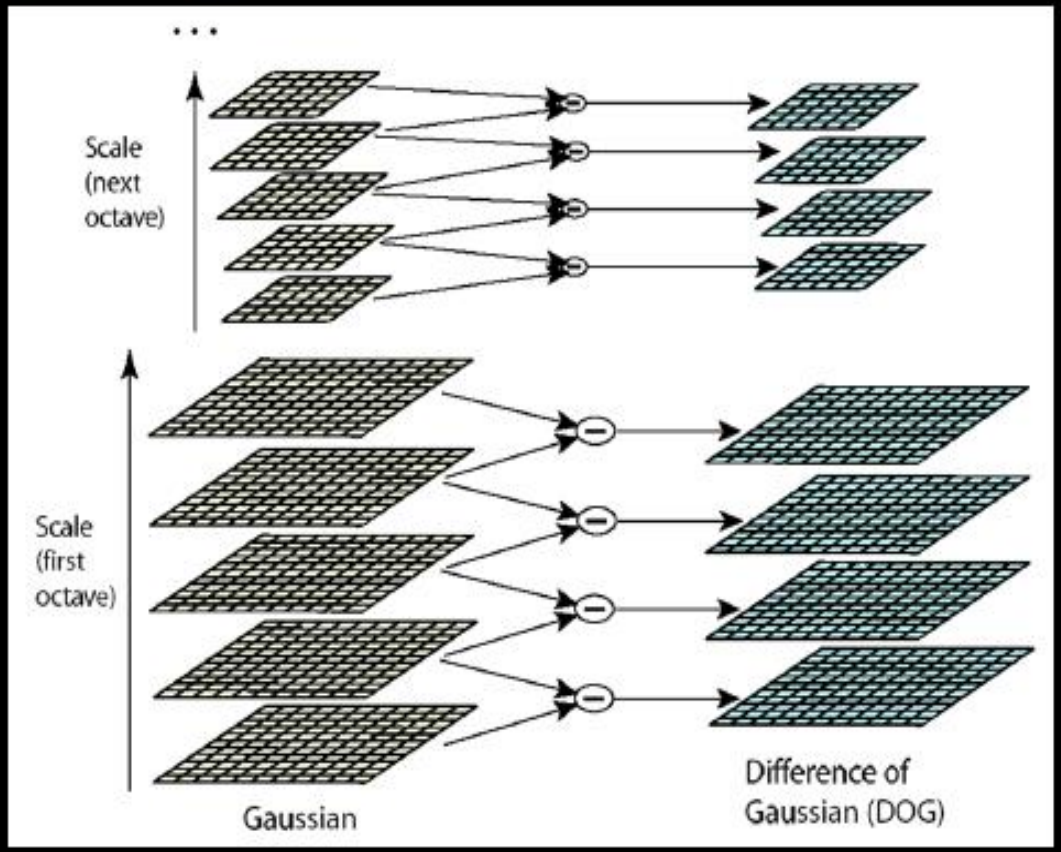
Key point localization

- SIFT: **S**cale **I**nvariant **F**eature **T**ransform

- Specific suggestion: use DoG pyramid to find maximum values (remember edge detection?) – then eliminate "edges" and pick only corners.

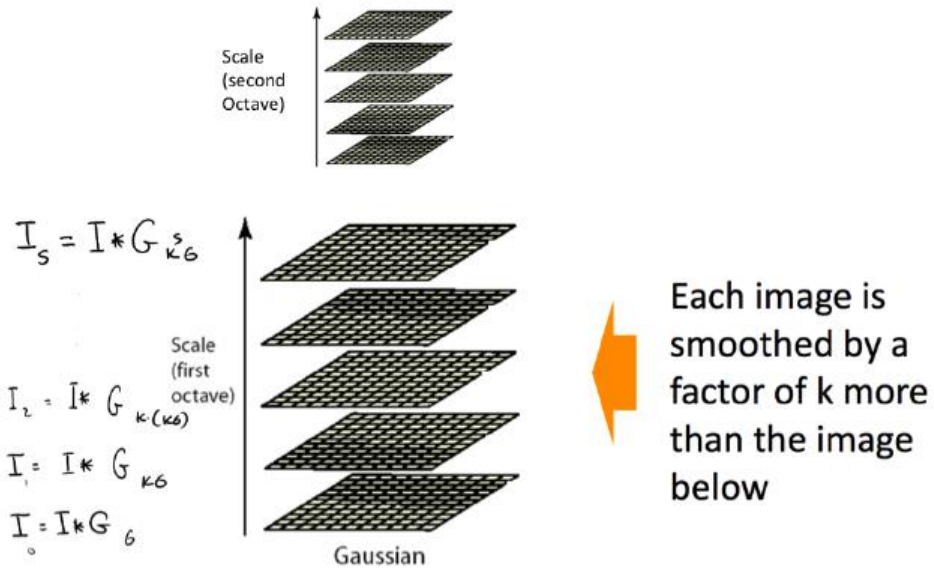


Scale space processed one octave at a time



Lowe's DoG

- First compute a Gaussian image pyramid



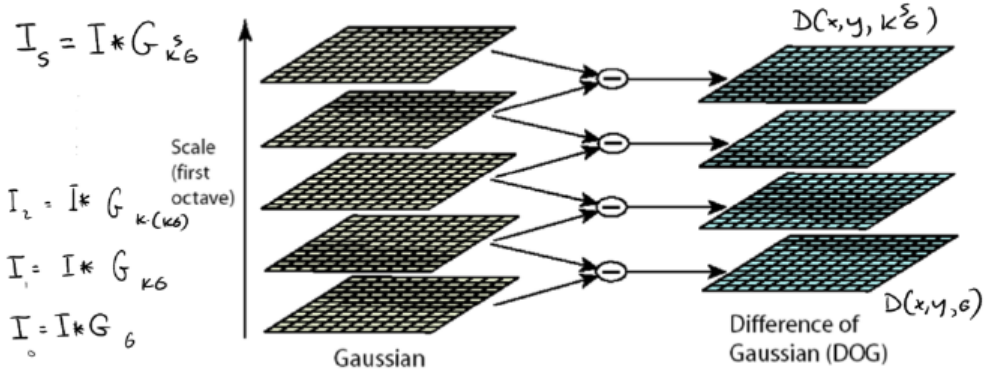
[Source: F. Flores-Mangas]

Lowe's DoG

- First compute a Gaussian image pyramid
- Compute Difference of Gaussians

$$D(x, y, \rho) = I(x, y) * (G(x, y, k\rho) - G(x, y, \rho))$$

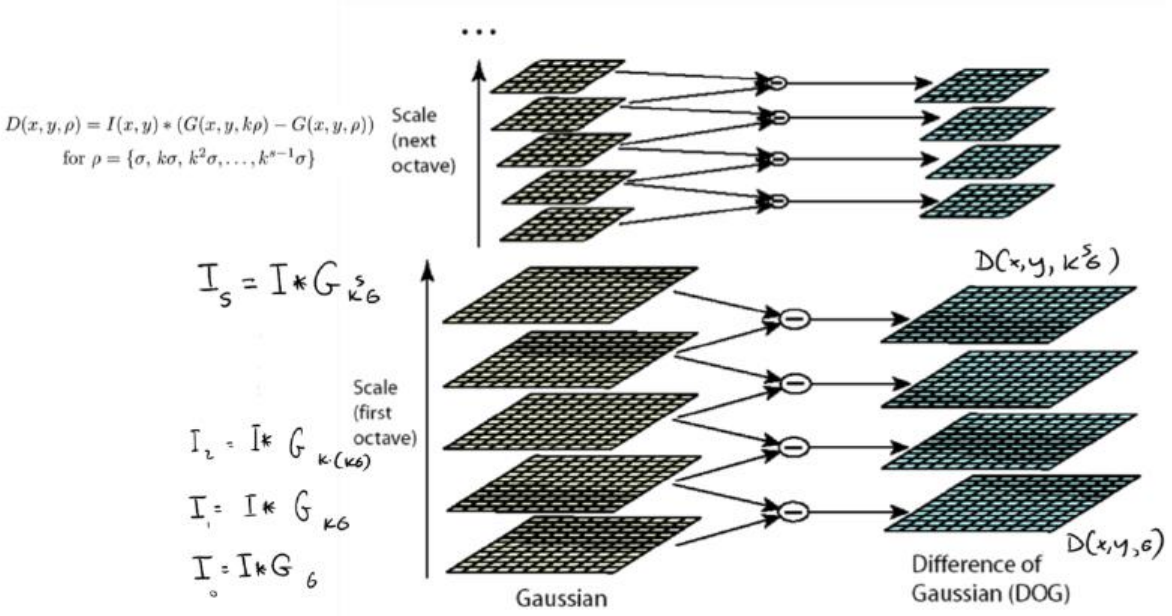
$$\text{for } \rho = \{\sigma,\ k\sigma,\ k^2\sigma, \ldots, k^{s-1}\sigma\}, \quad k = 2^{1/s}$$



[Source: F. Flores-Mangas]

Lowe's DoG

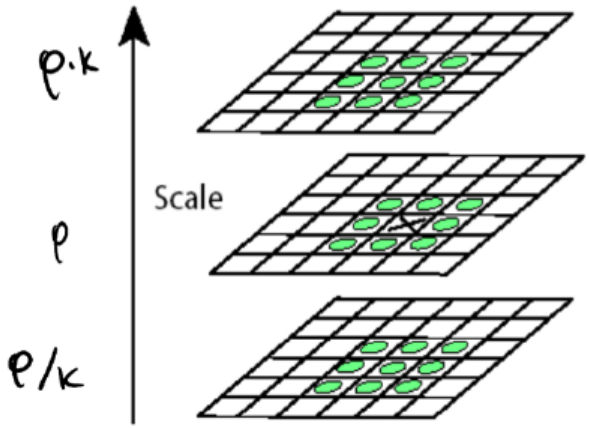
- First compute a Gaussian image pyramid
- Compute Difference of Gaussians
- At every scale



[Source: F. Flores-Mangas]

Lowe's DoG

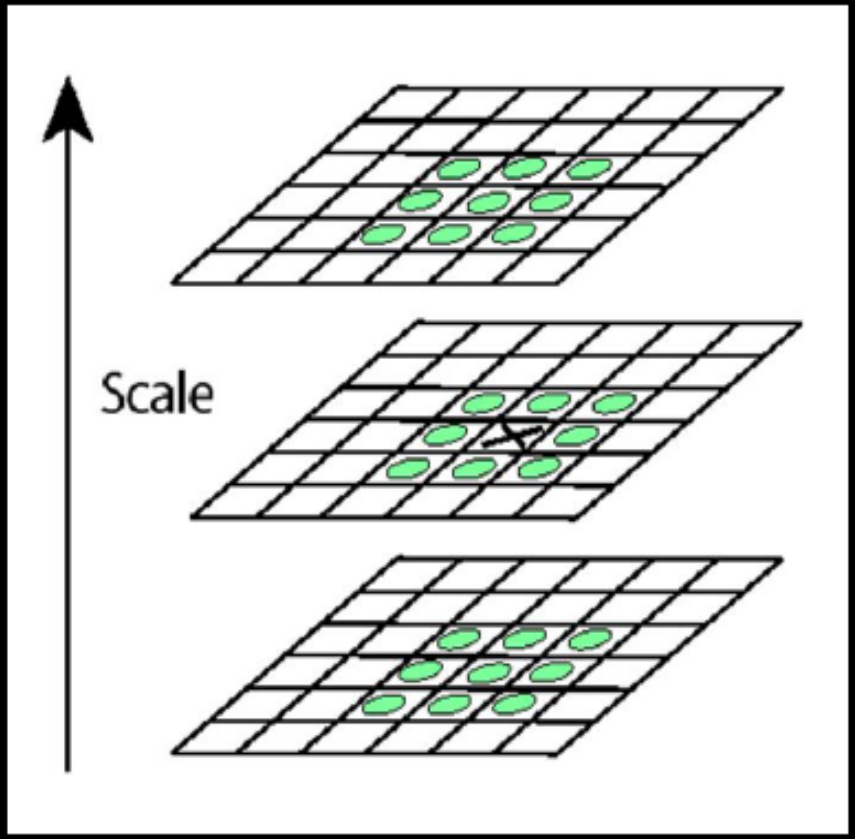
- First compute a Gaussian image pyramid
- Compute Difference of Gaussians
- At every scale
- Find local maxima in scale
- A bit of pruning of bad maxima and we're done!



[Source: F. Flores-Mangas]

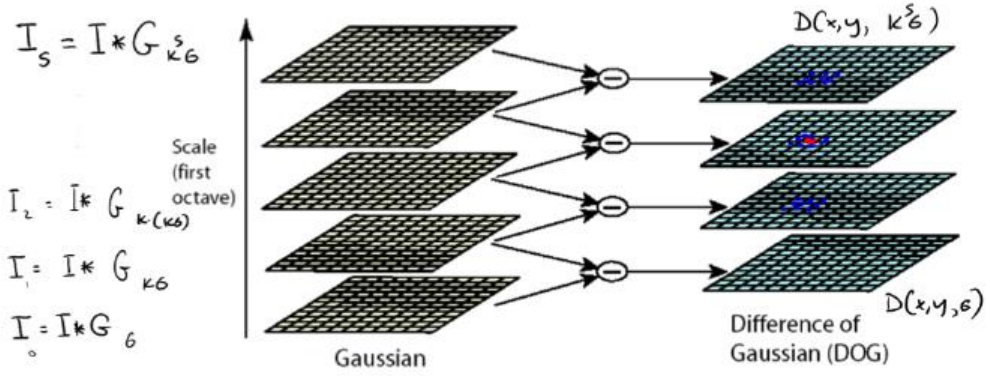
Key point localization

(Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below.)



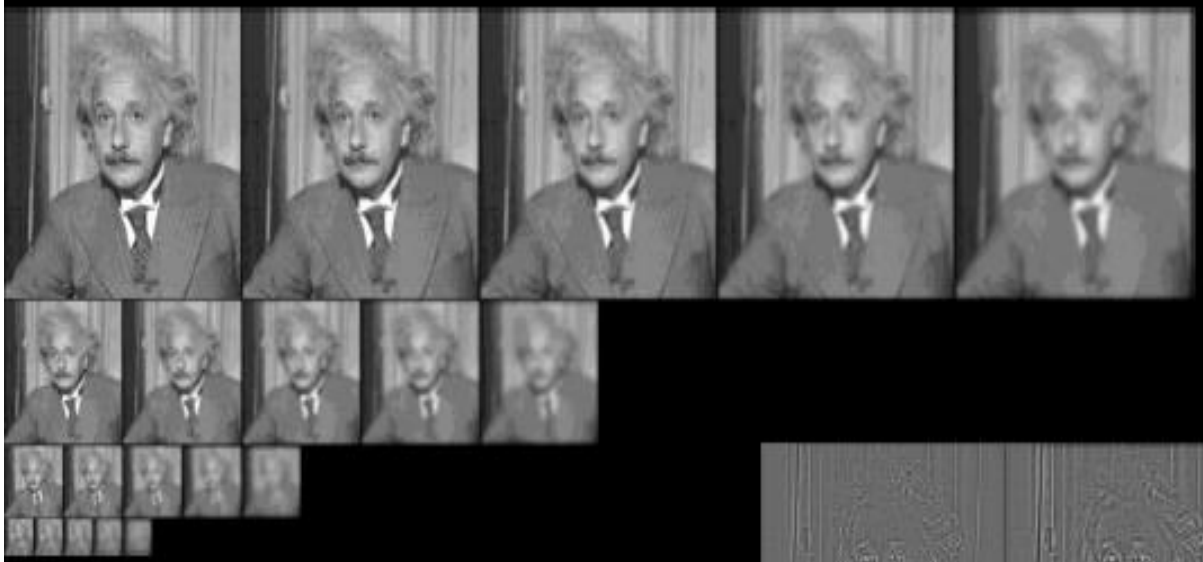
Lowe's DoG

- First compute a Gaussian image pyramid
- Compute Difference of Gaussians
- At every scale
- Find local maxima in scale
- A bit of pruning of bad maxima and we're done!



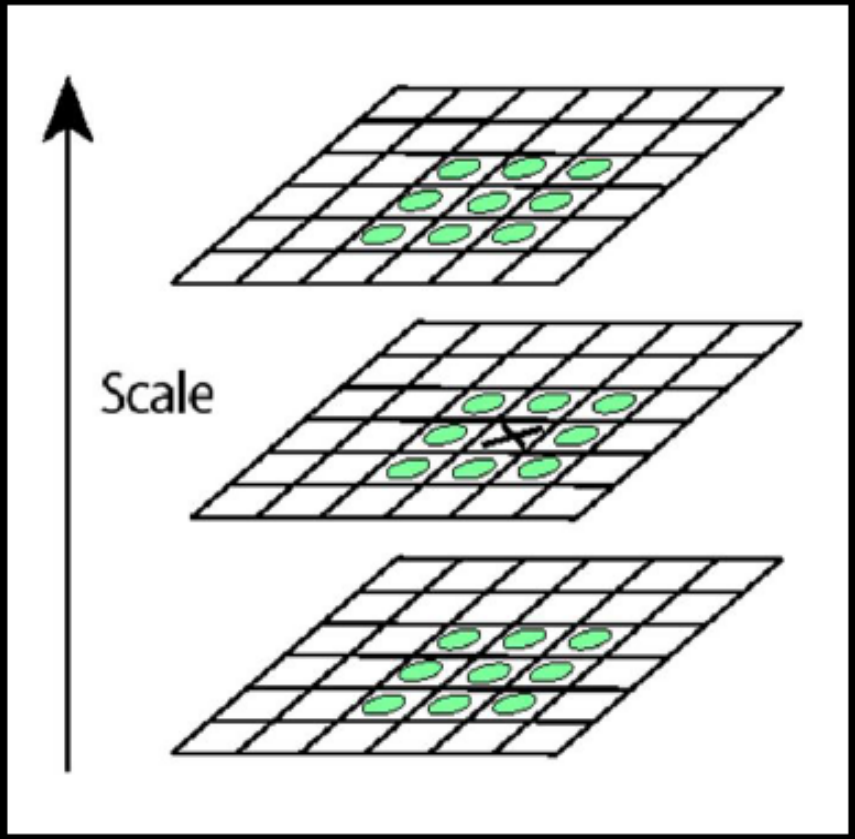
[Source: F. Flores-Mangas]

Extrema at different scales

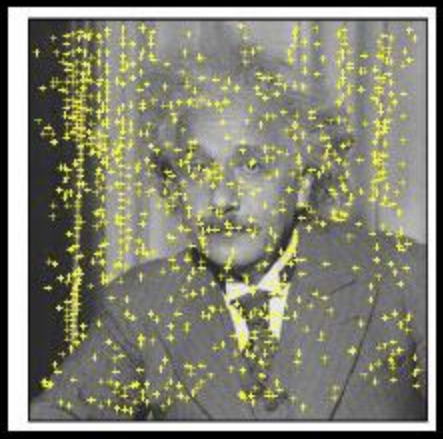


Key point localization

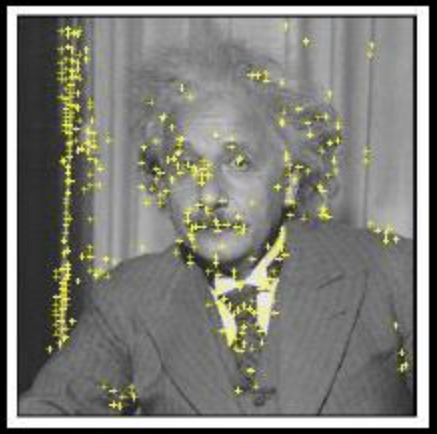
- General idea: find robust extremum (maximum or minimum) both in space and in scale.



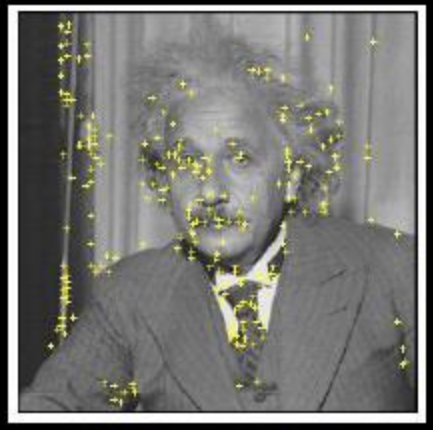
Remove low contrast, edge bound



Extrema points



Contrast > C



Not on edge

Scale Invariant Feature Transform (SIFT) Local Descriptors

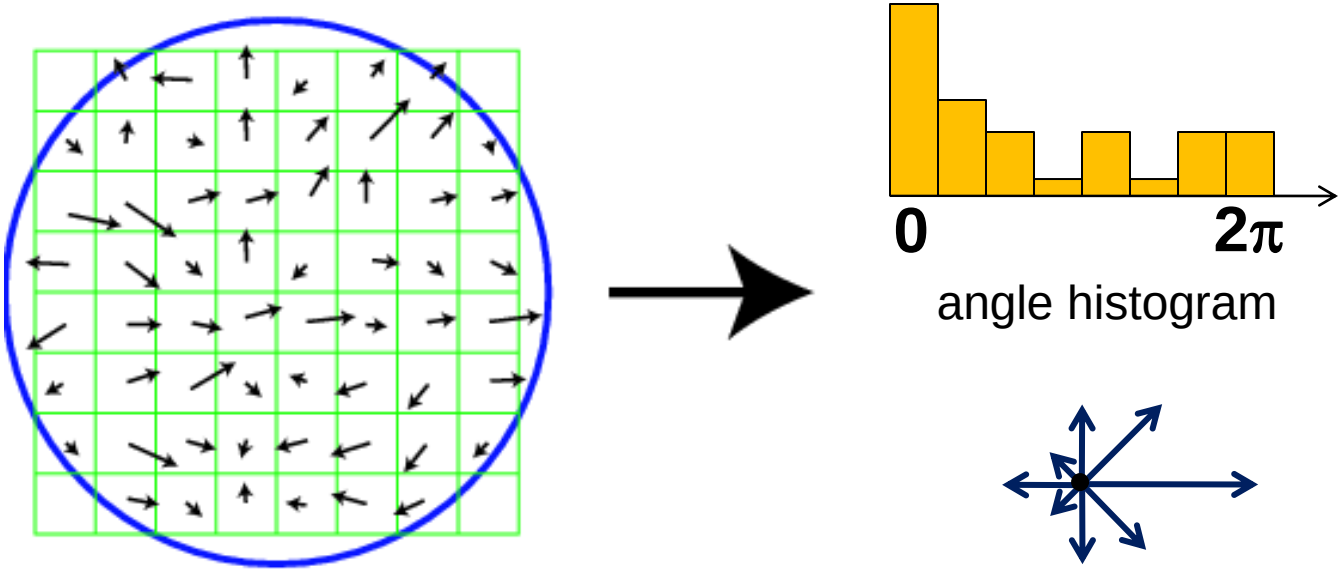
- SIFT has become one of the most important local image features.
- It is used in image retrieval, image classification, object detection, etc.
- The new trend is to use it in association with the bag-of-words (BoW) approach.

Scale Invariant Feature Transform

David Lowe IJCV 2004

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

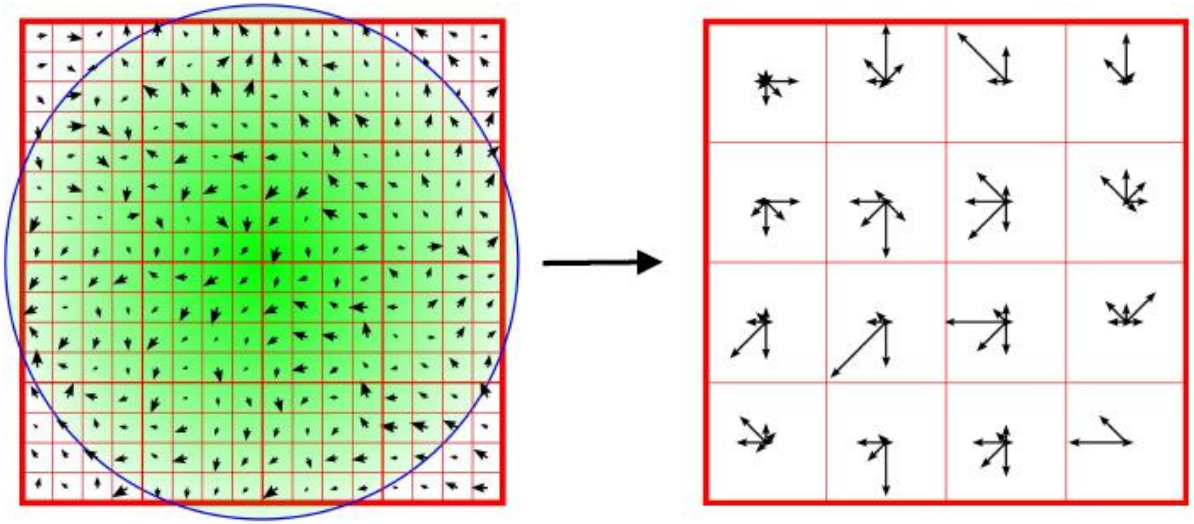


Adapted from slide by David Lowe

Former NYU faculty &
Prof. Ken Perlin's advisor

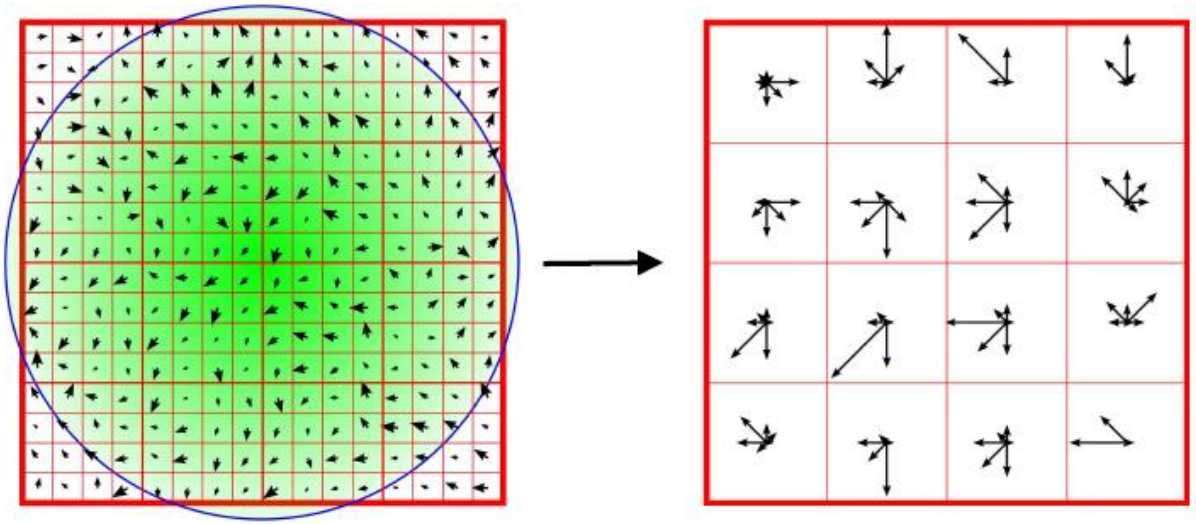
Orientation Histogram

- 4x4 spatial block (16 blocks total)
- Gaussian center-weighting
- 8-bin orientation histogram per block
- 8 x 16 = 128 dimensions total
- Normalized to unit norm



Orientation Histogram at each interest point

- At each interest point, have a window of 16x16 pixels.
- Divide into 4x4 spatial bins (16 bins total)
- Gaussian center-weighting
- 8-bin orientation histogram per bin
- 8 x 16 = 128 dimensions feature vector per point



State of the art features

Recently, it has been shown that local descriptors are the best representation of images in retrieval domain.

Scale-invariant feature transform (SIFT), and speeded-up robust features (SURF) are the best descriptors right now.

To compare 2 images, we need to fix the length of the feature vectors, so we use the bag of words (BoW) approach

BOW

A dictionary is made using thousands of image descriptors. Assume the dictionary has M descriptors (atoms).

For any image, we get its descriptors then create a histogram for each descriptor in the dictionary (how many times this descriptor [in the nearest neighbor sense] occurred in the image). So, the histogram vector is of length M.

Image Retrieval as an Unsupervised Classification

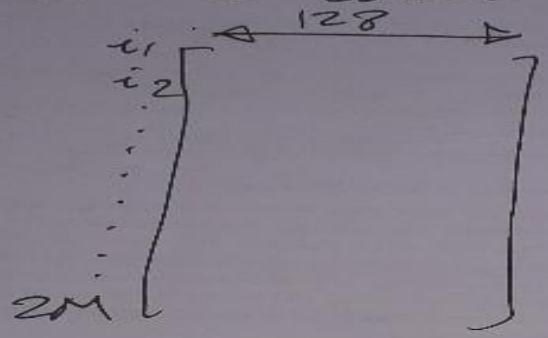
- A feature vector is extracted for each image in the database.
- For the query, the feature vector is extracted.
- A distance is calculated between the query feature vector and all the database vectors.
- City Block and Euclidean distances are the most popular.
- If we have N images in the database, and the feature vector dimension is M, then complexity is **O**(N.M)

Bag of Words Approach

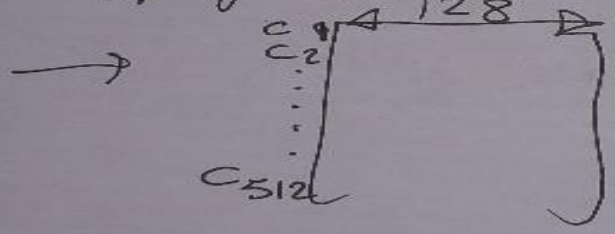
- Collect a large number of images, N, e.g. N = 10,000
- Apply the SIFT interest points algorithm on the N images
- So, we create a Huge array of SIFT *descriptor* vectors.
- Apply K-means with K=512 atoms for example
- For each image, get a histogram using nearest atom to each descriptor in the image
- Thus, each image is represented by a vector of length 512 in this case.

① collect a lot of Images
(e.g. 10,000 images)

② assume each image ≃ 200 interest point

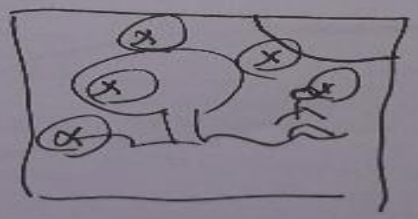


③ Apply (K-means) with (K = 512)



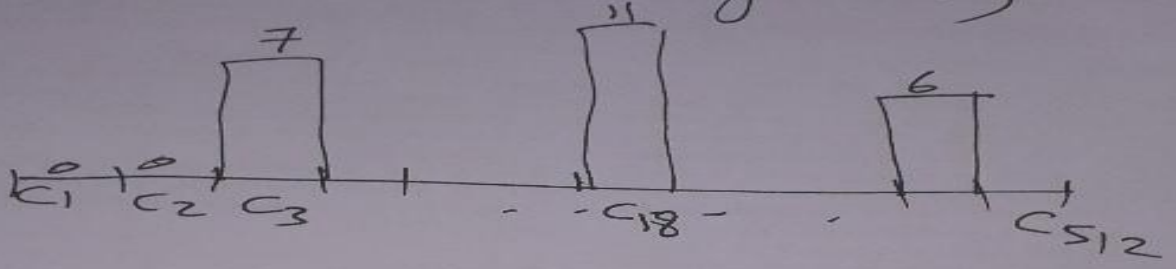
dictionary
(visual words)

④ each image has many interest points
e.g ⟹ (215)



(5) each point, find its nearest v.word from dictionary → increase counter (+1)

(histogram)

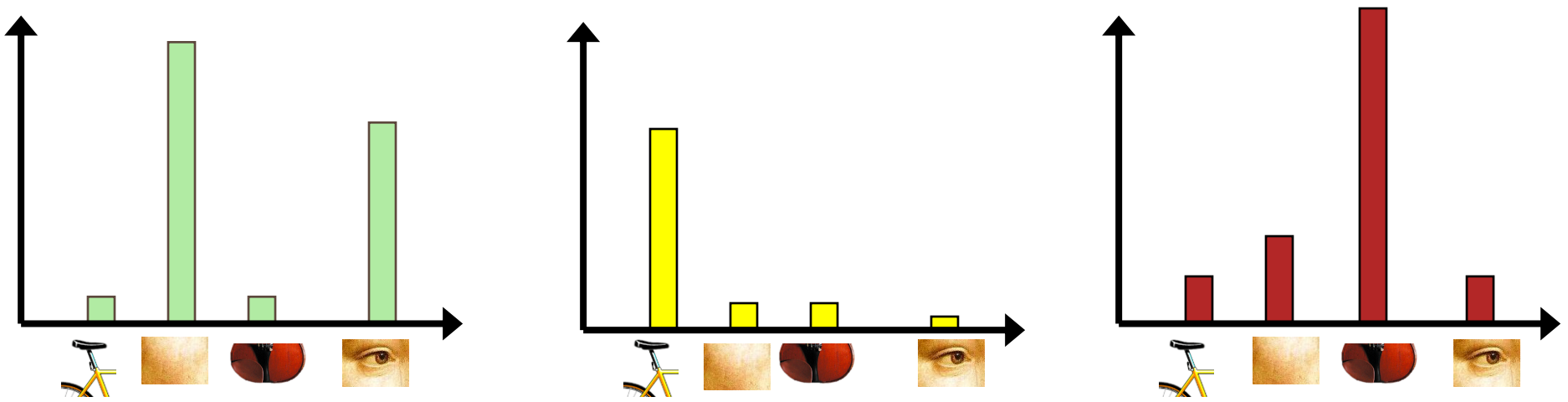


(6) then normalize by (215) ← # of interest points

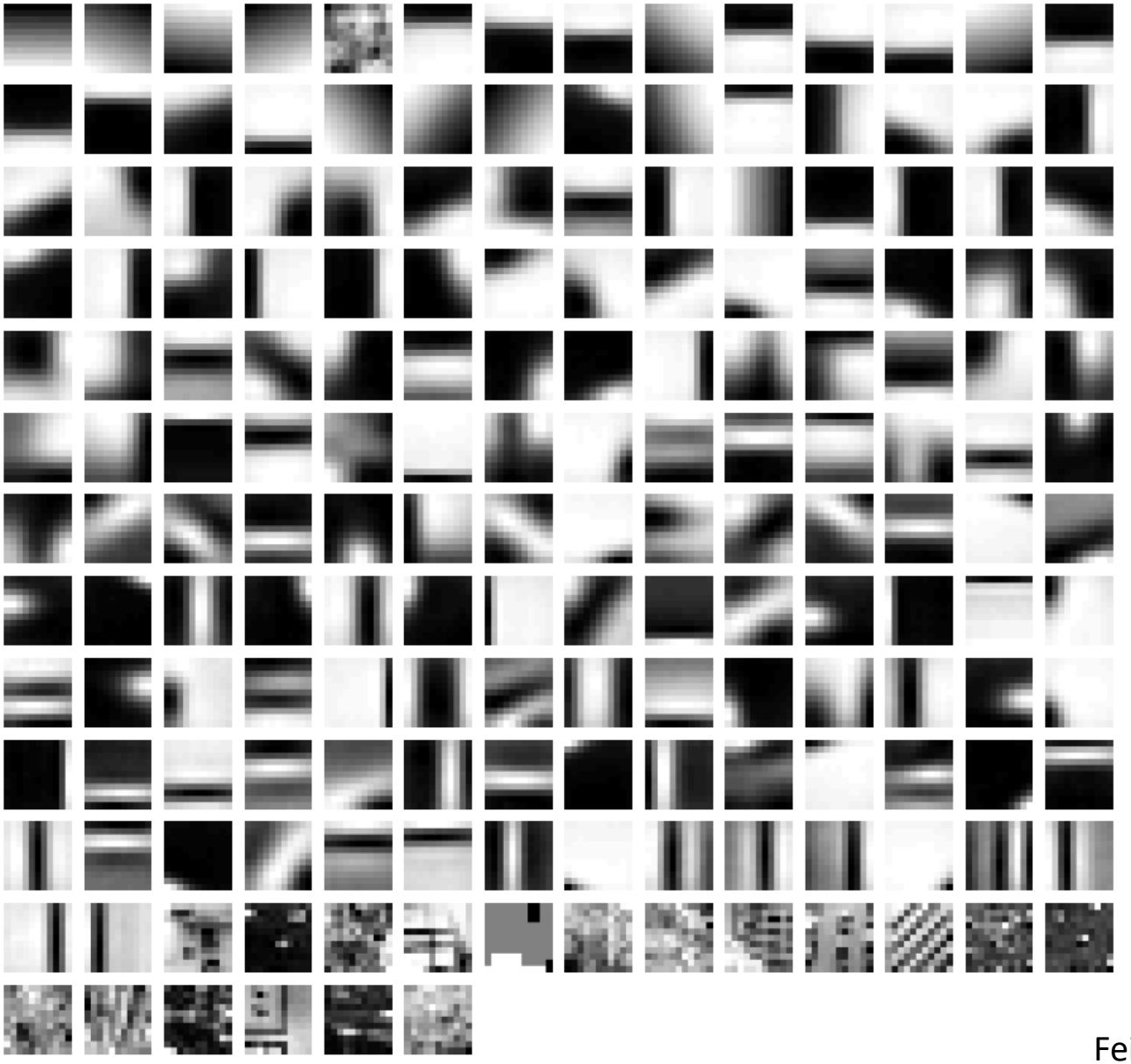
(7) so, (BoW) feature vector length is (512).

Bags of features for image classification

1. Extract features (e.g., the SIFT features)
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary (K-means clustering)
4. Represent images by normalized frequencies of “visual words” (normalized histograms)



Example visual vocabulary



Fei-Fei et al. 2005

Euclidean Distance

- Euclidean Distance

$$dist = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2}$$

Where n is the number of dimensions (attributes) and p_k and q_k are, respectively, the kth attributes (components) or data objects p and q.

Minkowski Distance

- Minkowski Distance is a generalization of Euclidean Distance

$$dist = \left(\sum_{k=1}^{n} | p_k - q_k |^r \right)^{\frac{1}{r}}$$

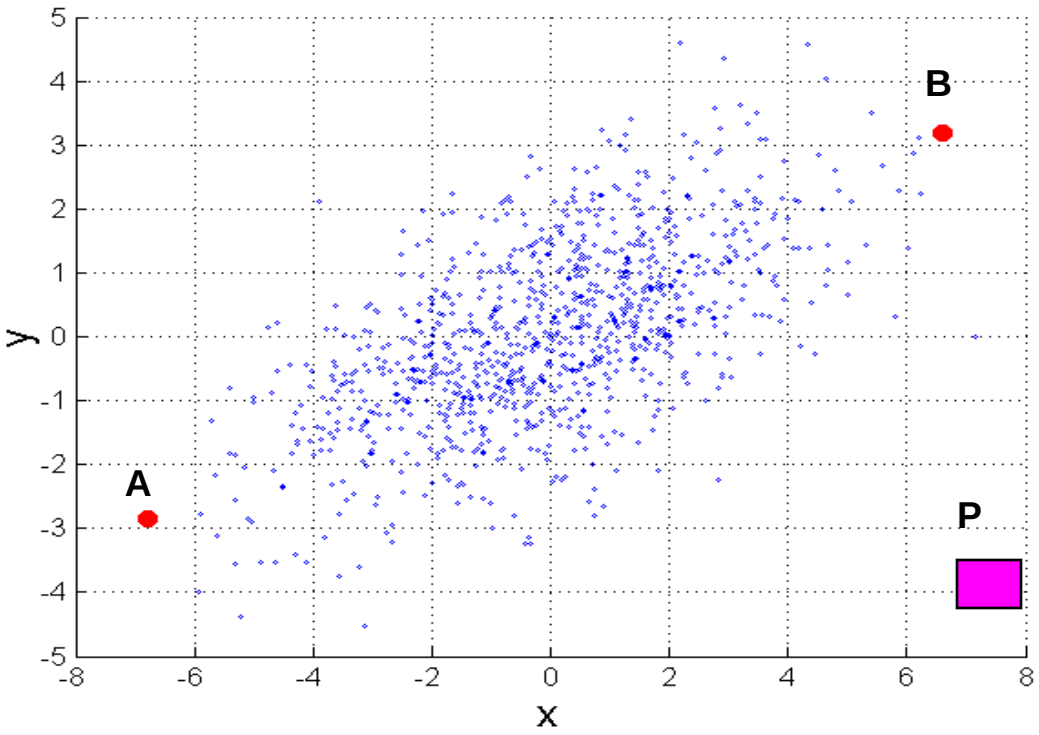
Where r is a parameter, n is the number of dimensions (attributes) and p_k and q_k are, respectively, the kth attributes (components) or data objects p and q.

Minkowski Distance: Examples

- $r = 1$. City block (Manhattan, taxicab, L_1 norm) distance.
  - A common example of this is the Hamming distance, which is just the number of bits that are different between two binary vectors
- $r = 2$. Euclidean distance
- $r \rightarrow \infty$. "supremum" (L_{max} norm, L_∞ norm) distance.
  - This is the **maximum difference** between any component of the vectors
  - Example: L_infinity of (1, 0, 2) and (6, 0, 3) = ??

Mahalanobis Distance

$$mahalanobis(p,q) = (p-q)\Sigma^{-1}(p-q)^T$$



Σ is the covariance matrix of the input data *X*

$$\Sigma_{j,k} = \frac{1}{n-1}\sum_{i=1}^{n}(X_{ij} - \overline{X}_j)(X_{ik} - \overline{X}_k)$$

When the covariance matrix is identity Matrix, the mahalanobis distance is the same as the Euclidean distance.

Useful for detecting outliers.

Q: what is the shape of data when covariance matrix is identity?
Q: A is closer to P or B?

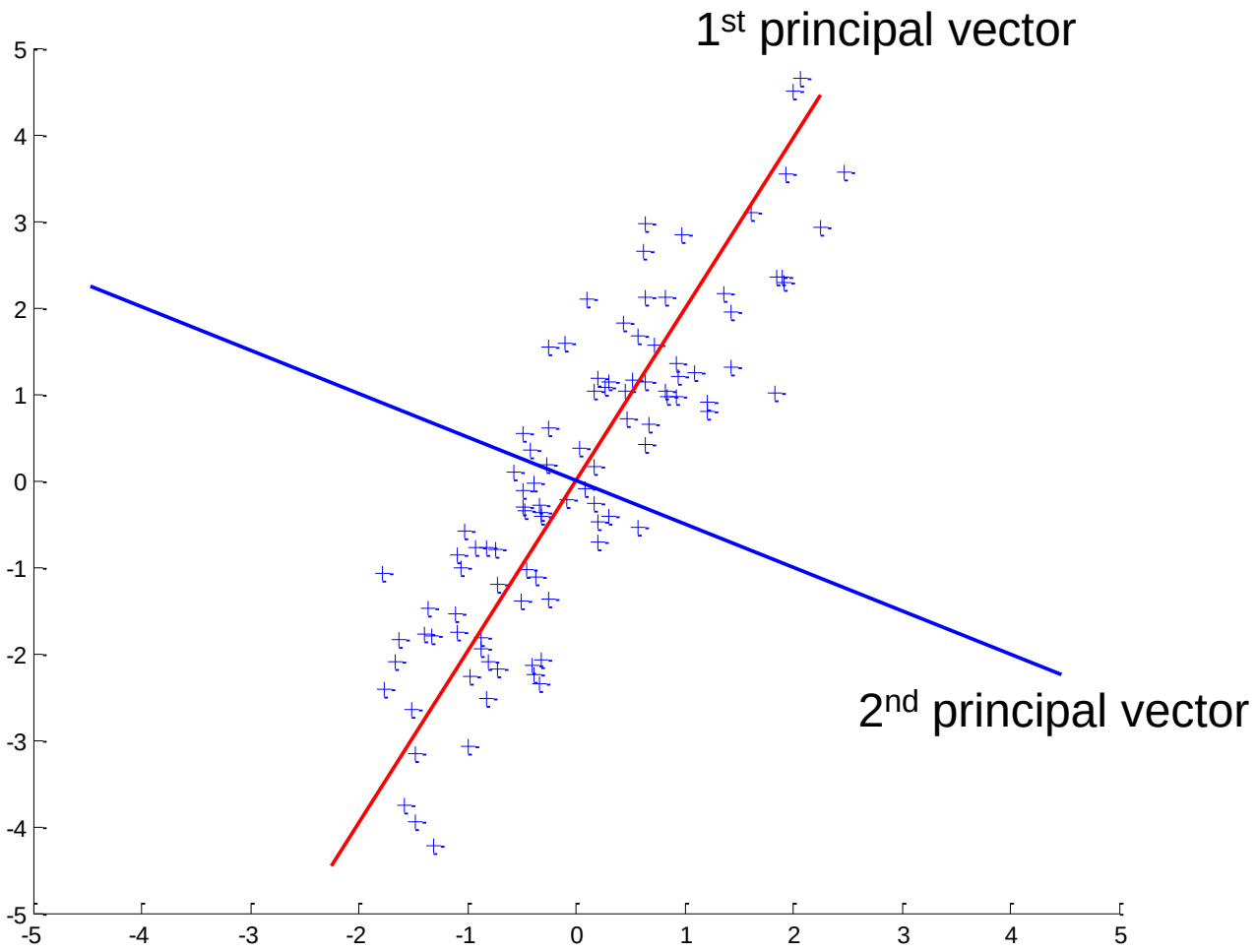
For red points, the Euclidean distance is 14.7, Mahalanobis distance is 6.

Curse of Dimensionality

- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality
  - For high dimensional data distances are all very large and the discrimination between classes can be obscured by noise.
  - Feature selection and feature extraction are two approaches for dimensionality reduction
  - (PCA can be used): see Matlab example

Principal Component Analysis (PCA)

- Gives best axis to project
- Minimum RMS error
- Principal vectors are orthogonal



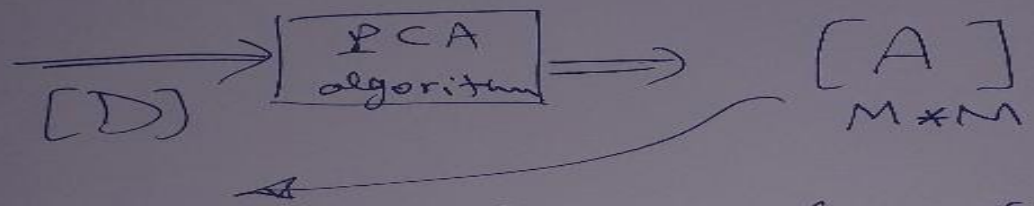
Principal Component Analysis (PCA)

Given : Training data array

$[D]$

$N * M$

examples — # Features



we choose to keep only 1st (n) of the principal components ($n \ll M$)

$\underset{n*M}{A'}$ ⟶ we keep only 1st (n) rows of $[A]$.

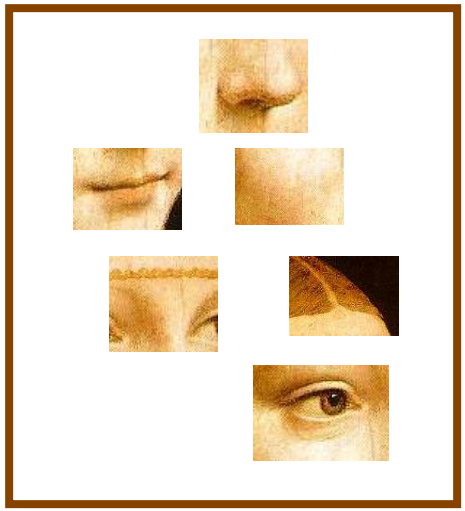
Then, any vector $\underset{M*1}{X}$

$$\underset{n*1}{X'} = \underset{n*M}{A'} * \underset{M*1}{X}$$

⟶ reduced dim.

Bags of features for image classification

1. Extract features



Bags of features for image classification

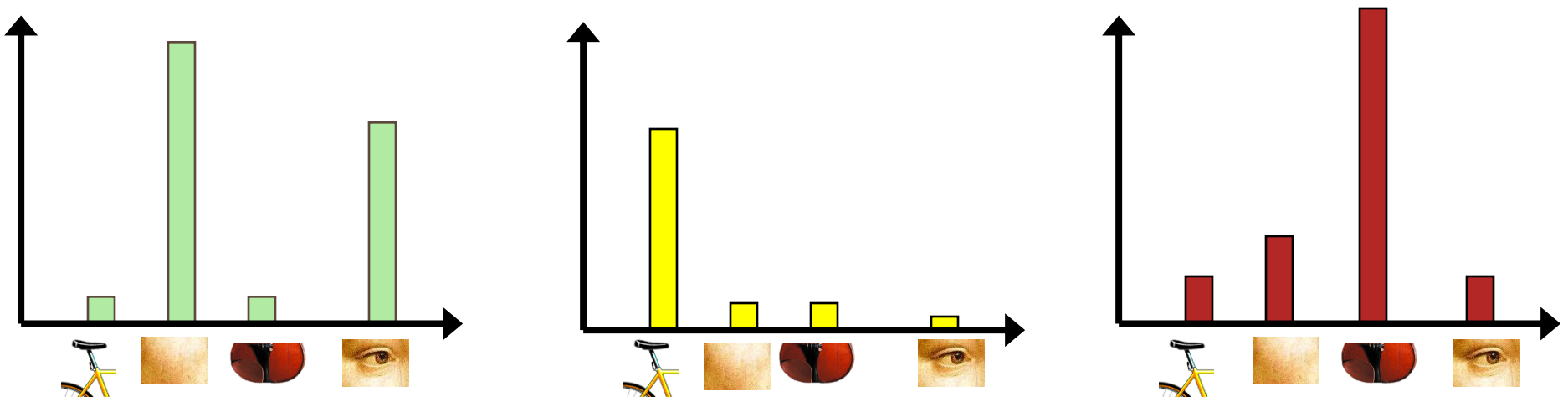
1. Extract features
2. Learn “visual vocabulary”

Bags of features for image classification

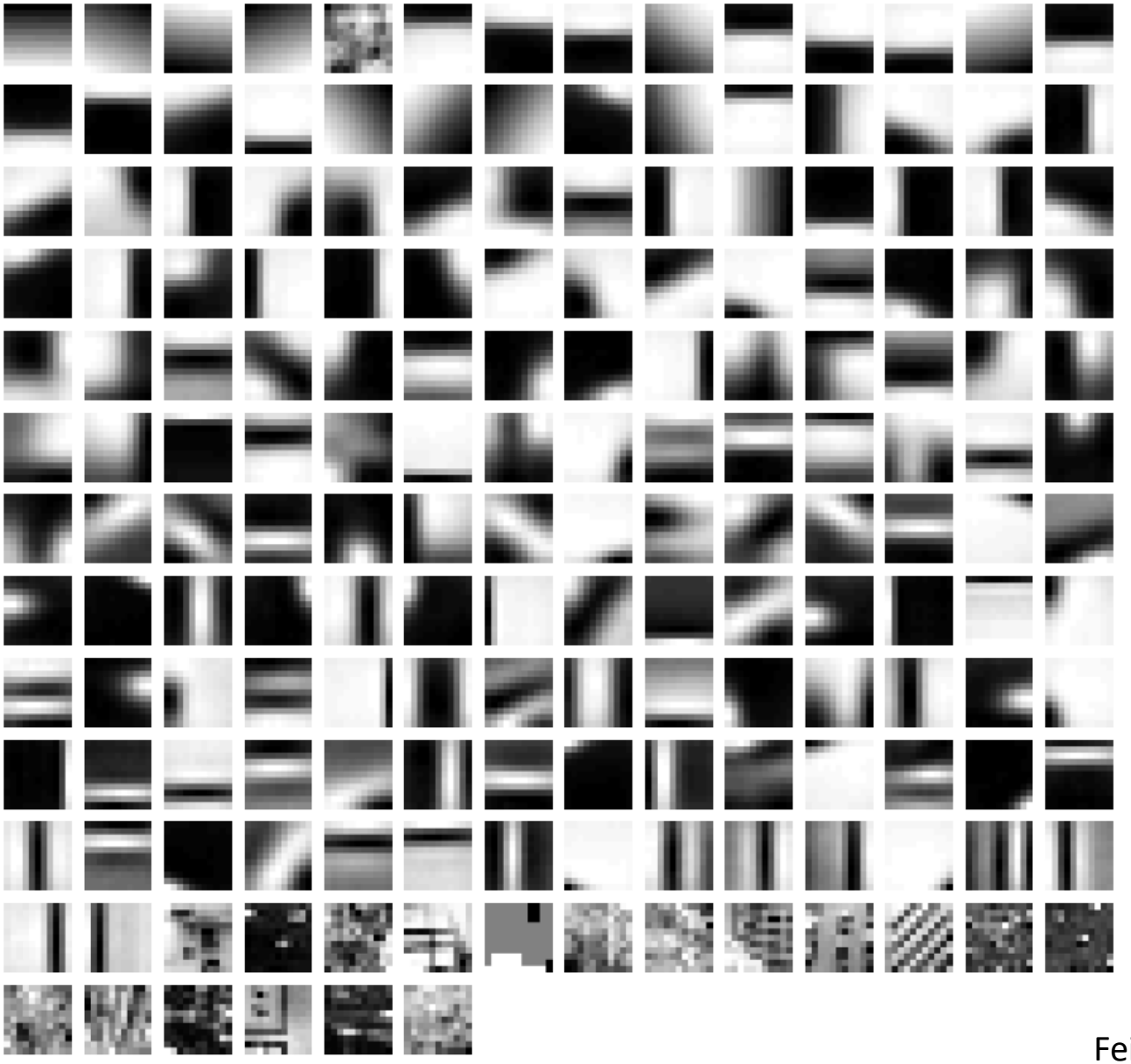
1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary

Bags of features for image classification

1. Extract features (e.g., the SIFT features)
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary (K-means clustering)
4. Represent images by normalized frequencies of “visual words” (normalized histograms)



Example visual vocabulary



Fei-Fei et al. 2005

Image representation by histogram

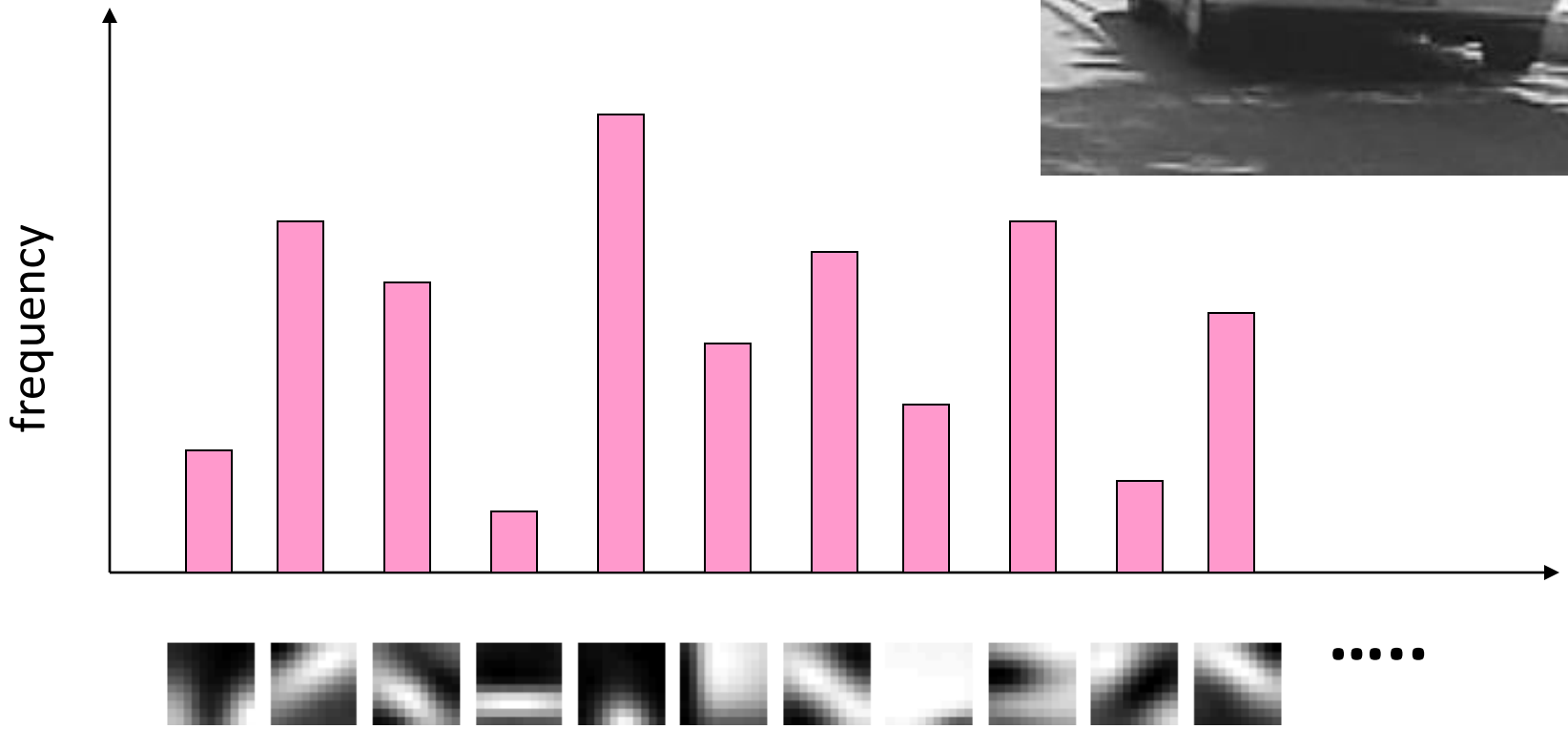
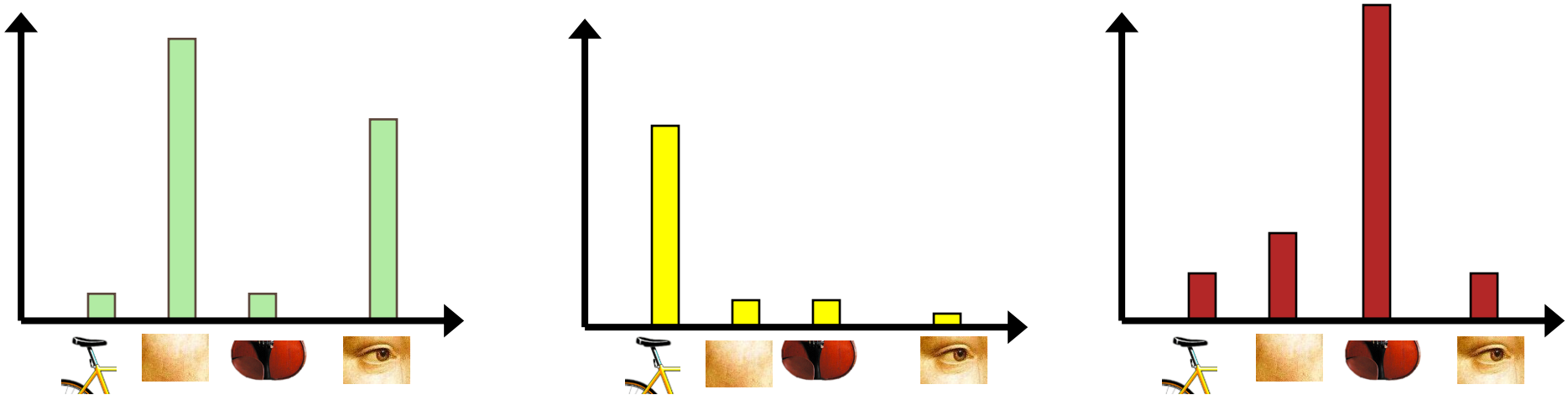


Image classification

- Given the bag-of-features representations of images from different classes, how do we learn a model for distinguishing them?



- SIFT
- PCA-SIFT
- GLOH
- HOG
- SURF
- DAISY
- LBP
- Shape Contexts
- Color Histograms

RANSAC and image stitching using SIFT features

Random sample consensus

- 3 steps algorithm:
- 1- sample randomly minimum number of data points to create the model (e.g. for a line we sample 2 points)
- 2- count how many other data points are considered Inliers for that model (based on a certain threshold of closeness)→ this is the score for that model
- 3- Repeat steps 1 and 2 and finally select the model with the highest score and the associated Inliers. The rest of the data are treated as outliers

Example 1: Fitting a Line

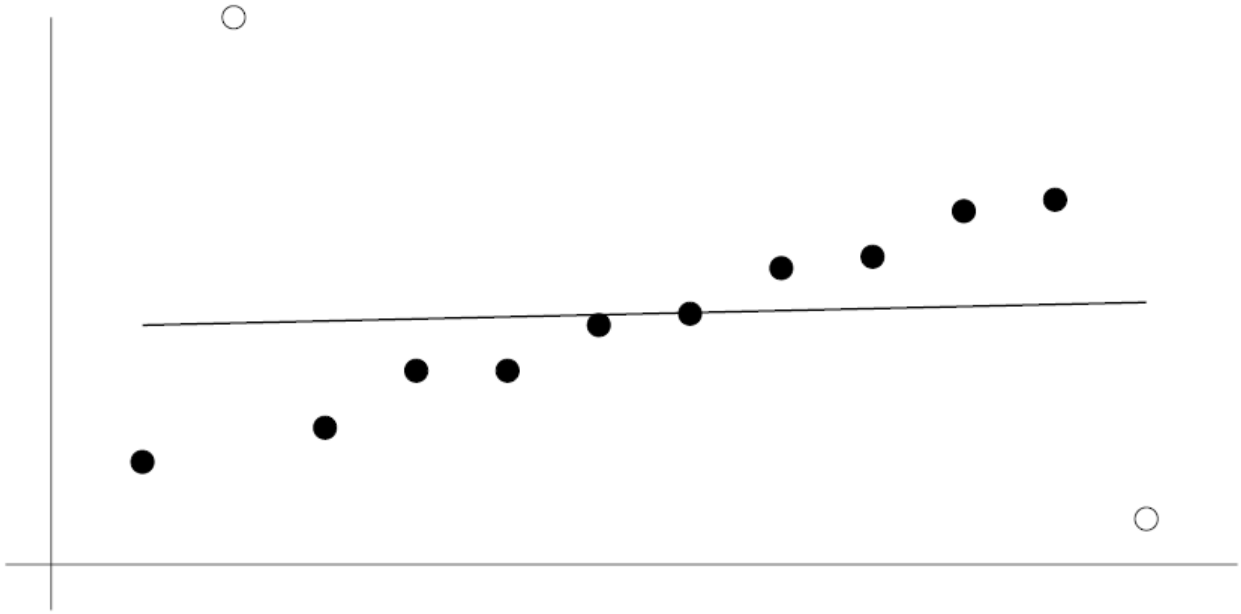


Figure Credit: Hartley & Zisserman

Example 1: Fitting a Line

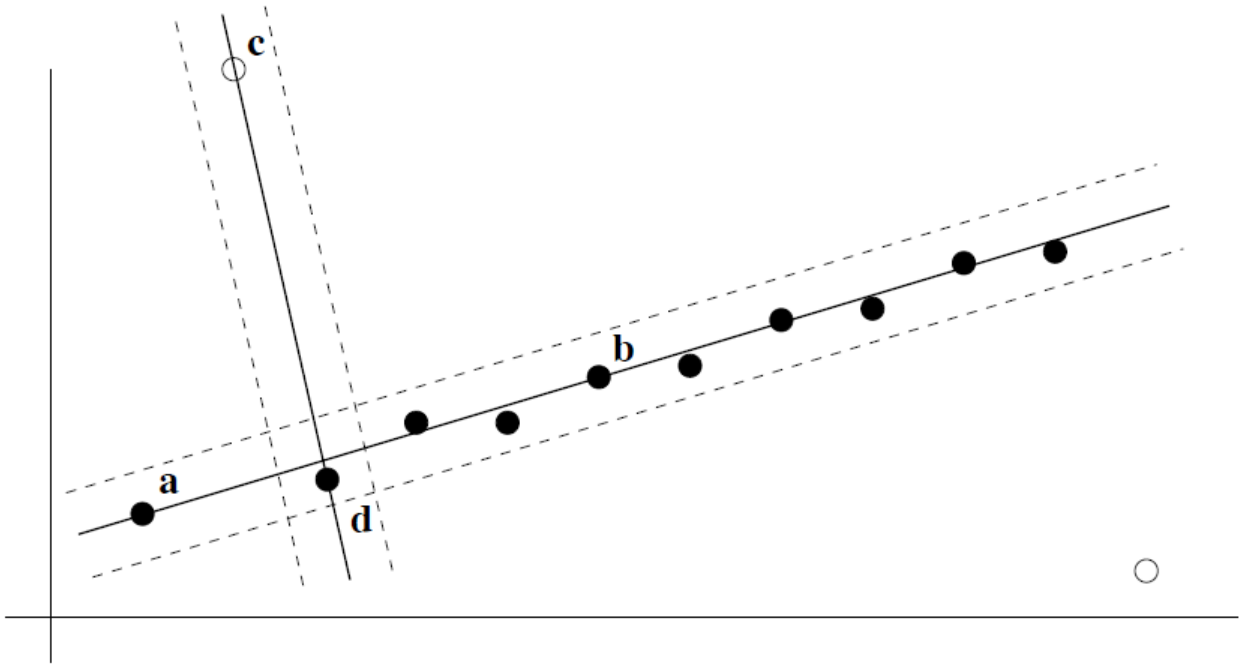


Figure Credit: Hartley & Zisserman

a with b ← 1st line, c with d ← 2nd line, etc.

Example 1: Fitting a Line

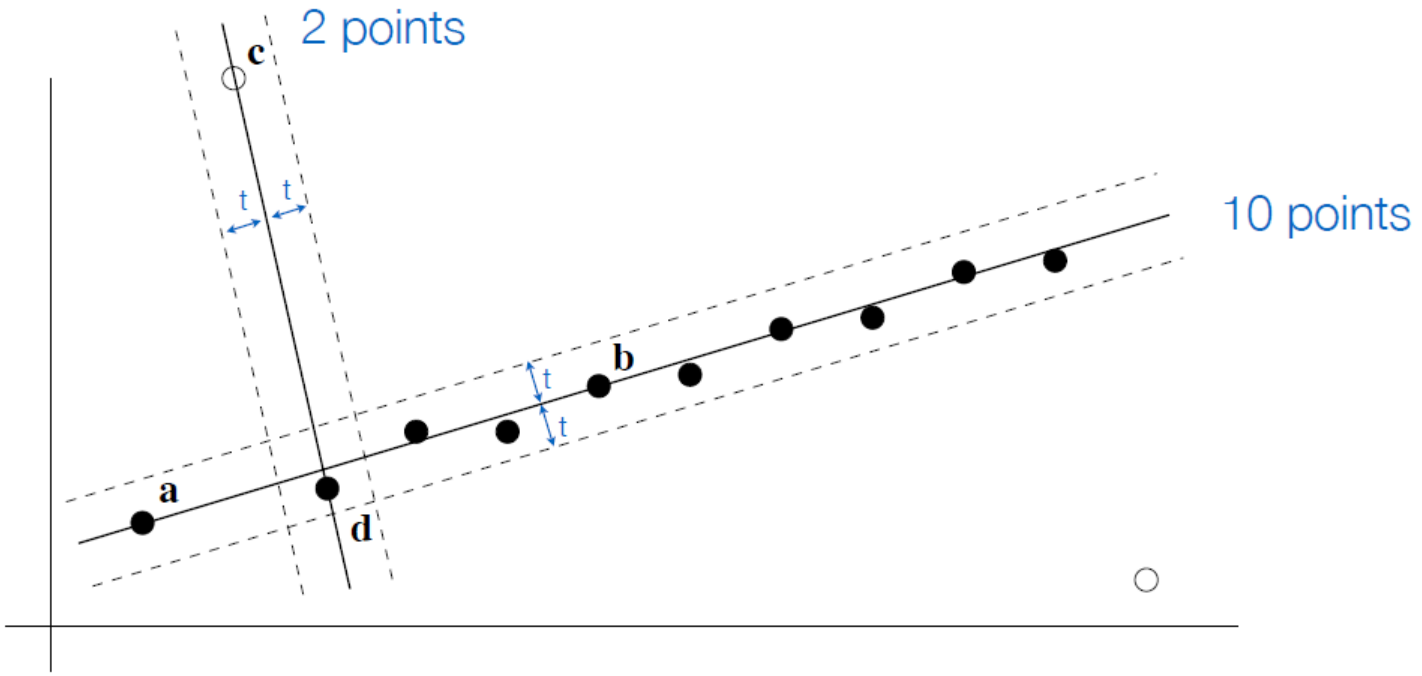


Figure Credit: Hartley & Zisserman

How many times to Sample??

- $T = \frac{\log(1-p)}{\log(1-(1-e)^s)}$
- If you want to succeed with probability ***p*** and the outlier ratio in the data is ***e*** and the number of data points required for the model is ***s*** then we need to make ***T*** *Trials*

Example 3: Automatic Matching of Images

— How to get correct correspondences without human intervention?

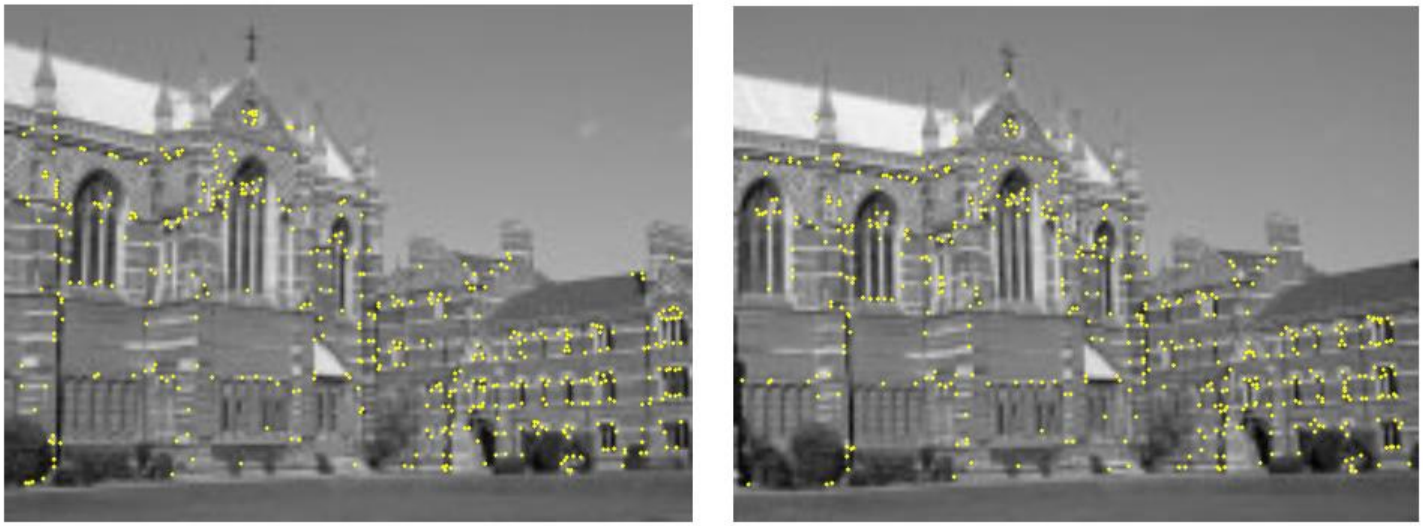
— Can be used for image stitching or automatic determination of epipolar geometry

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

SIFT is better since it is also scale-invariant

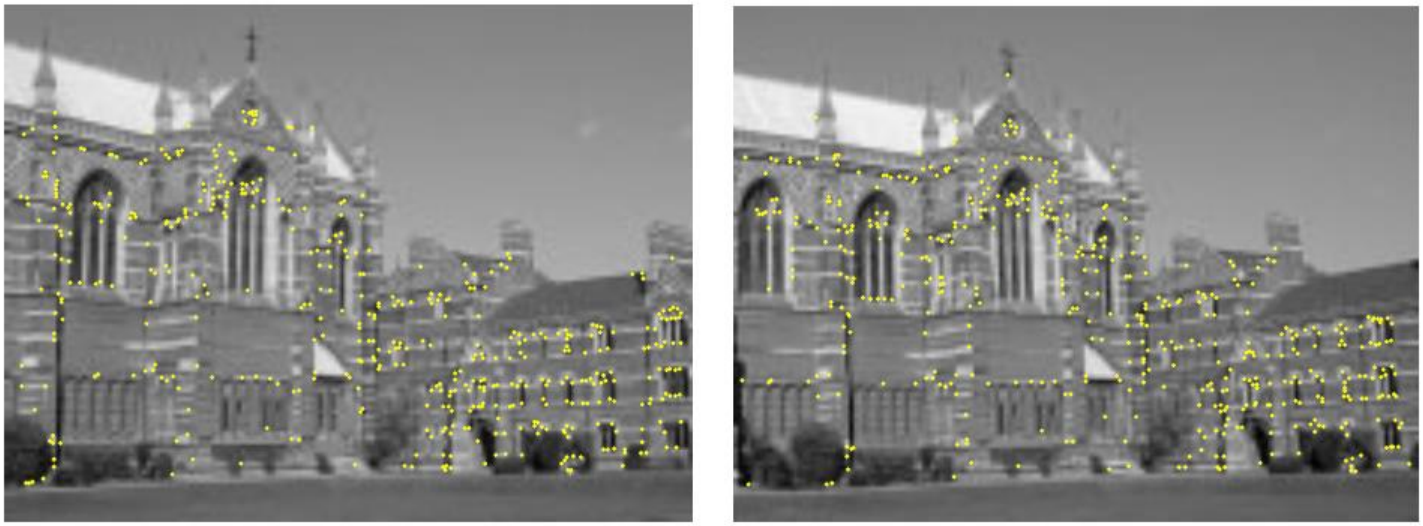


≈ 500 corner features found in each image

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

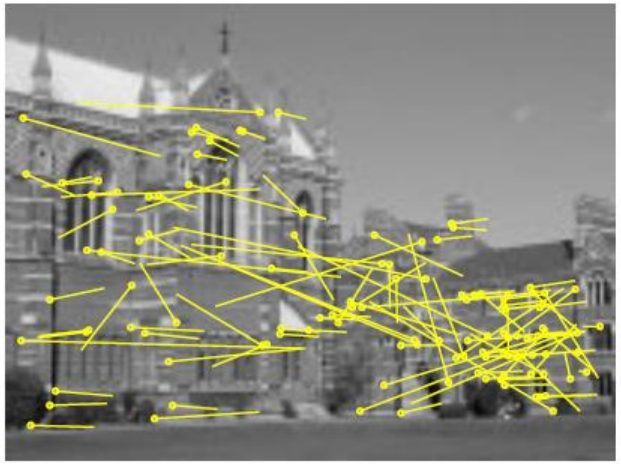
Example 3: Initial Match Hypothesis

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

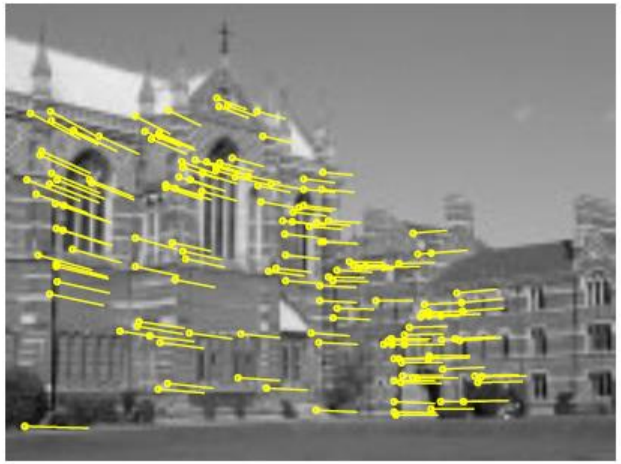
Figure Credit: Hartley & Zisserman

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**



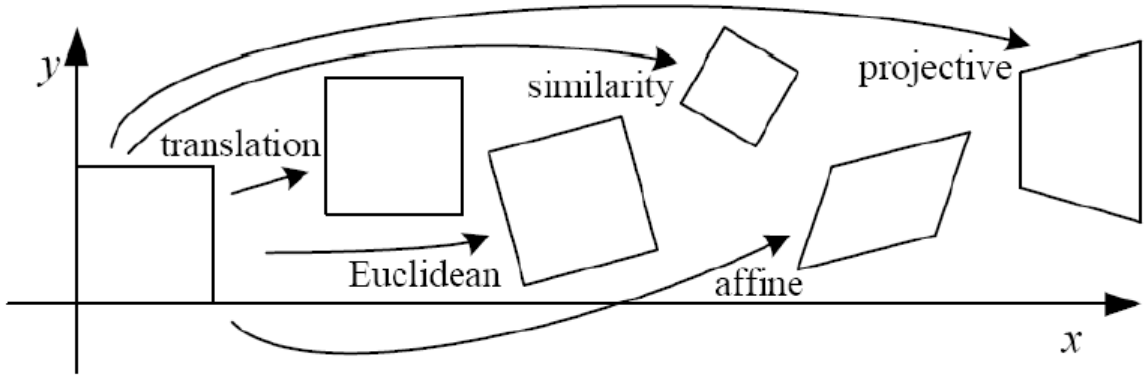
151 **inliers**

Figure Credit: Hartley & Zisserman

Image Homography

- How to transform one image to another based on the matching points and the RANSAC algorithm?
- This is called Homography (warping one image and aligning it with another reference image)
- This is used in image stitching (2 or more images e.g. panoramic view), and in image registration.
- So, we want to compute the Homography matrix that projects one image to another

2D image transformations



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times3}$	2	orientation + ···	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times3}$	3	lengths + ···	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times3}$	4	angles + ···	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times3}$	6	parallelism + ···	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times3}$	8	straight lines	

See Hartley and Zisserman, p. 44

These transformations are a nested set of groups

- Closed under composition and inverse is a member

Basic 2D Transformations via 3x3 matrices

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Shear

all of the above are special cases of a general Affine Transformation:

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Affine Transformations

Affine transformations are combinations of …

- Linear 2D transformations, and
- Translations

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Properties of affine transformations:

- Origin does not necessarily map to origin (**new** compared to 2x2 matrices)
- Lines map to lines
- Parallel lines remain parallel
- Length/distance ratios are preserved on parallel lines
- Ratios of areas are preserved
- Closed under composition

Projective Transformations (a.k.a. *homographies*)

transformations in homogeneous coordinate space via general 3x3 matrices

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Projective transformations …

- Affine transformations, and
- Projective warps

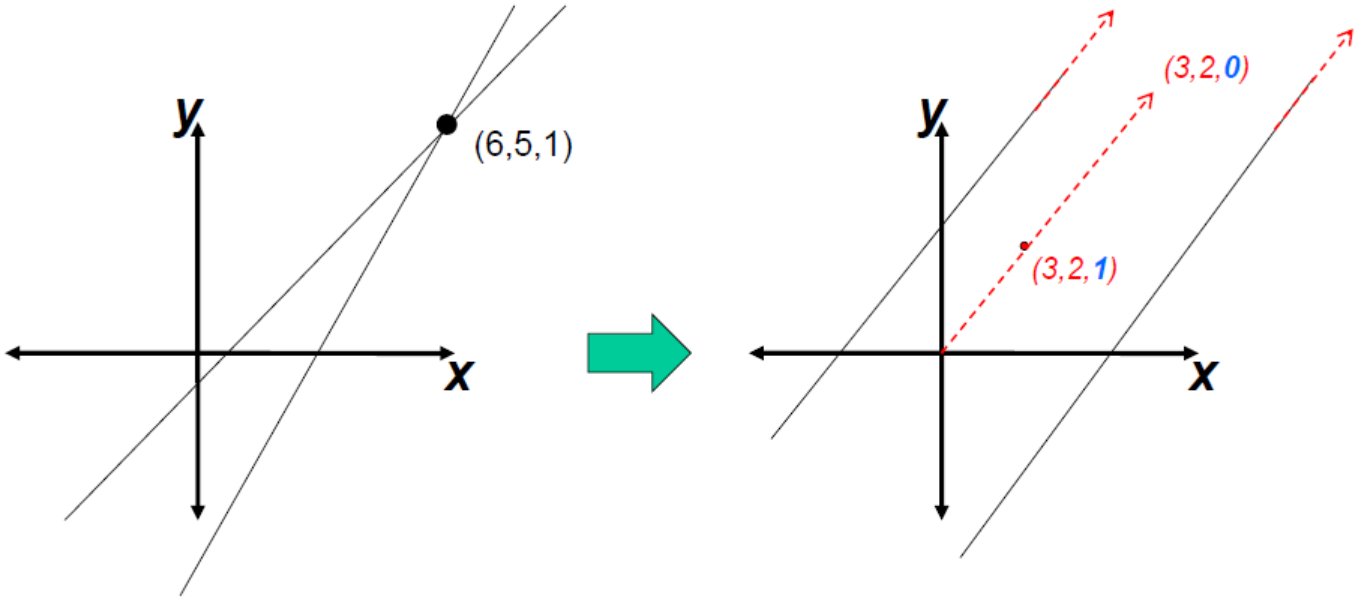
Properties of projective transformations:

- Origin does not necessarily map to origin
- Lines map to lines (indeed, line of hom. points **p** means $\mathbf{a}\cdot\mathbf{p}=0$ for some **a**. Then, $\mathbf{b}\cdot H\mathbf{p}=0$ for $\mathbf{b}=\mathbf{a}H^{-1}$)
- Parallel lines do not necessarily remain parallel
- Non-parallel lines may become parallel
- Distance/length or area ratios are not preserved
- Closed under composition

Projective Transformations (a.k.a. *homographies*)

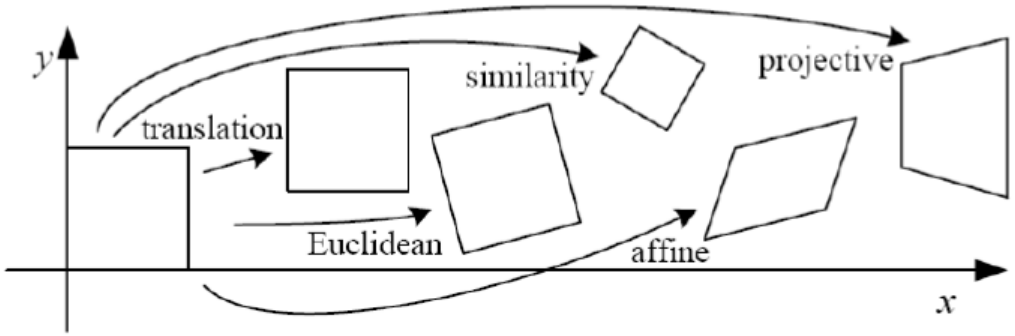
- Parallel lines do not necessarily remain parallel
- Non-parallel lines may become parallel

$$\begin{bmatrix} 3 \\ 2 \\ 0 \end{bmatrix} = \overbrace{\begin{bmatrix} a & b & c \\ d & e & f \\ -1 & 1 & 1 \end{bmatrix}}^{H} \begin{bmatrix} 6 \\ 5 \\ 1 \end{bmatrix}$$



NOTE: “finite” point may transform to “point at infinity”

2D image transformations



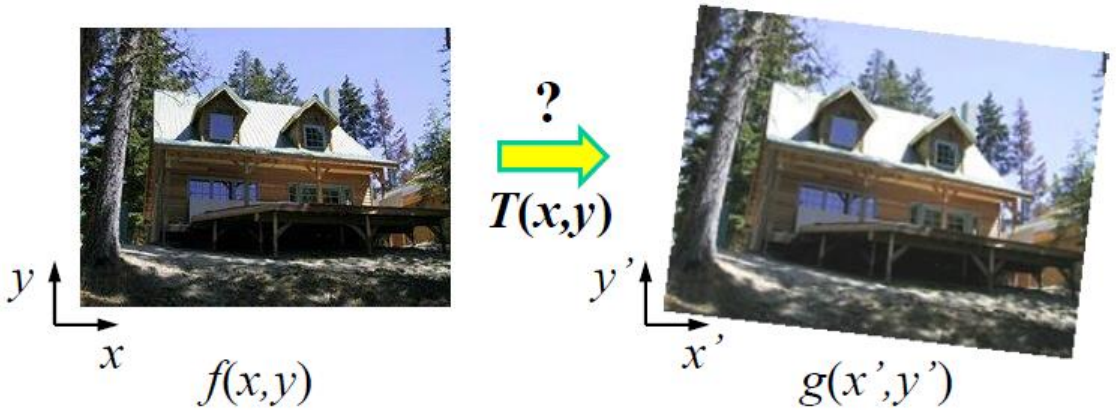
Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	4	angles $+\cdots$	
affine	$\left[\boldsymbol{A}\right]_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\left[\tilde{\boldsymbol{H}}\right]_{3\times 3}$	8	straight lines	

See Hartley and Zisserman, p. 44

These transformations are a nested set of groups

- Closed under composition and inverse is a member

Recovering Parametric Transformations

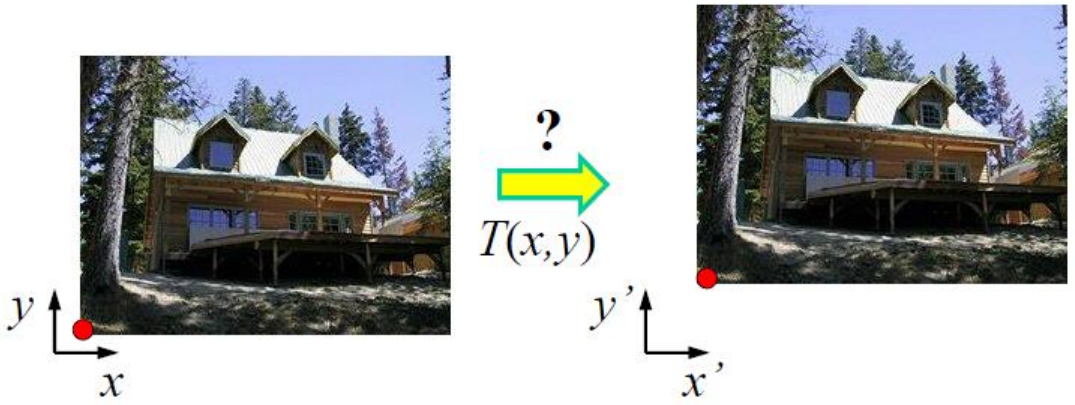


What if we know *f* and *g* and want to recover transform T?

- e.g. to better align images (**image registration**)
- willing to let user provide correspondences

Q: How many pairs of corresponding points do we need?

Translation: # correspondences?



How many correspondences needed for translation?

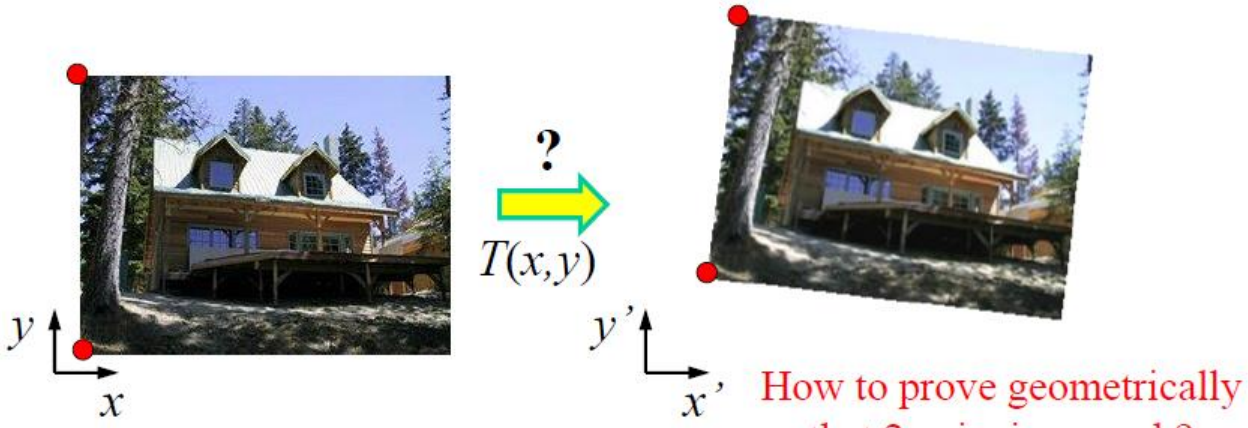
How many Degrees of Freedom (DOF)?

What is the transformation matrix?

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & c_x \\ 0 & 1 & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

2 parameters: 1 pair

Euclidian: # correspondences?



How to prove geometrically that 2 pairs is enough?
(use rigid transformation invariants to map an arbitrary point)

How many correspondences needed for translation+rotation?

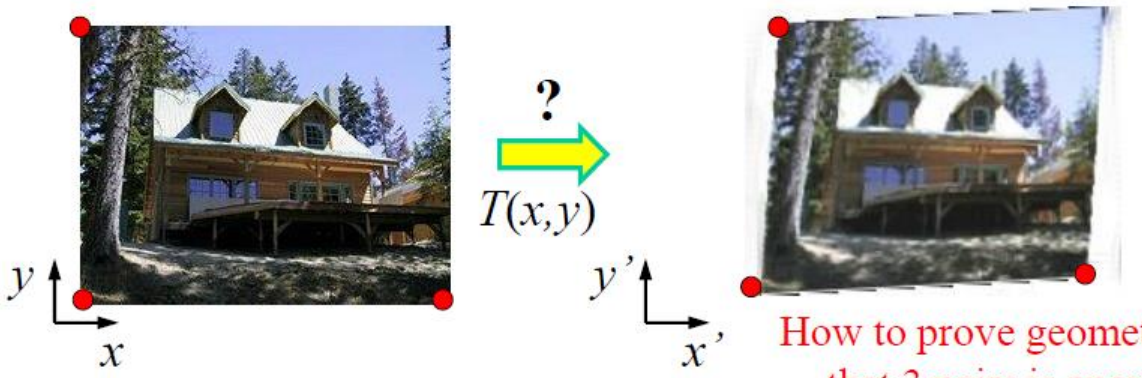
How many DOF?

Transformation matrix?

$$\mathbf{M} = \begin{bmatrix} \cos\theta & -\sin\theta & c_x \\ \sin\theta & \cos\theta & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

3 parameters: 2 pairs

Affine: # correspondences?



How to prove geometrically that 3 pairs is enough?
(use affine transformation invariants to map an arbitrary point)

How many correspondences needed for affine?

How many DOF?

Transformation matrix?

$$\mathbf{M} = \begin{bmatrix} a & b & c \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

6 parameters: 3 pairs

Algebraic point of view

For Any Affine Transformations

$$\begin{matrix} \mathbf{p}'_i & & M & & \mathbf{p}_i \end{matrix}$$

$$\mathbf{p}'_i = M\ \mathbf{p}_i \qquad \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

for any given pair of corresponding points $(\mathbf{p}_i, \mathbf{p}'_i)$

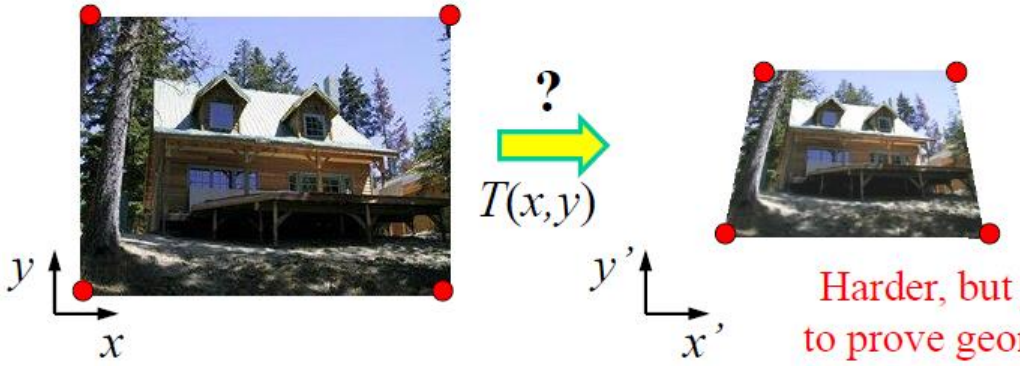
$$\Rightarrow \quad \begin{cases} x'_i = ax_i + by_i + c \\ y'_i = dx_i + ey_i + f \end{cases}$$

6 unknown parameters (variables)

Each pair of corresponding points $(\mathbf{p}_i, \mathbf{p}'_i)$ gives **two linear equations** w.r.t 6 unknown coefficients of matrix M with known point coordinates for $\mathbf{p}_i$ and $\mathbf{p}'_i$

3 pairs of corresponding points give 3x2 (=6) linear equations allowing to **resolve 6 unknown parameters**

Projective: # correspondences?



Harder, but possible to prove geometrically that 4 pairs is enough.
(can use only *line preservation*)

How many correspondences needed for projective?

How many DOF?

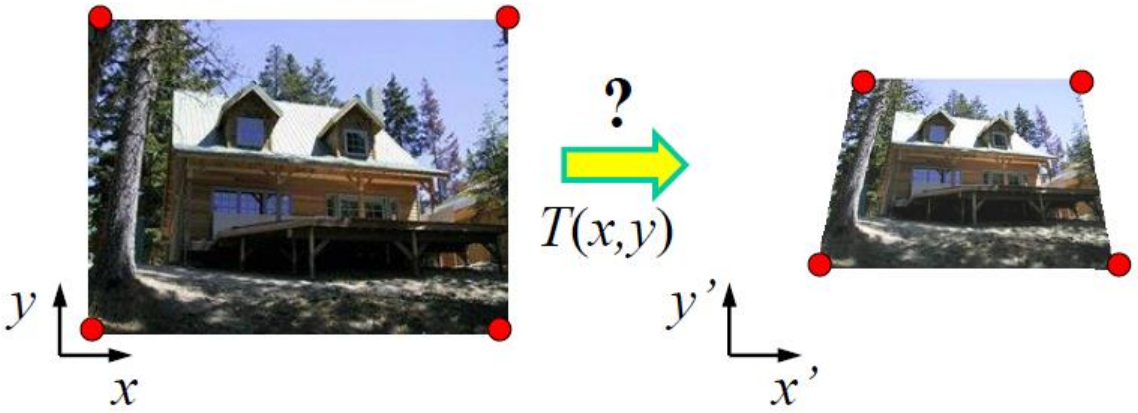
Transformation matrix?

Most general case: Projective Transformation: the 0 0 1 can take any values

UNIVERSITY OF WATERLOO

Projective: # correspondences?

Parallelism not preserved



How many correspondences needed for projective? $=4$

How many DOF?

Transformation matrix?

Easy to check that 4 pairs give only 4x2 (=8) equations!
What about 9 unknowns?

Homographies have only 8 DOF since scale is irrelevant
(multiplying M by any factor does not change the actual transformation).

More on estimating homographies
from 4 matching pairs of point - later in Topic 5.

$$\mathbf{M} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

8 parameters: 4 pairs

Rotation + Translation example

Image **Alignment + RANSAC**

In practice we have many noisy correspondences + **outliers**

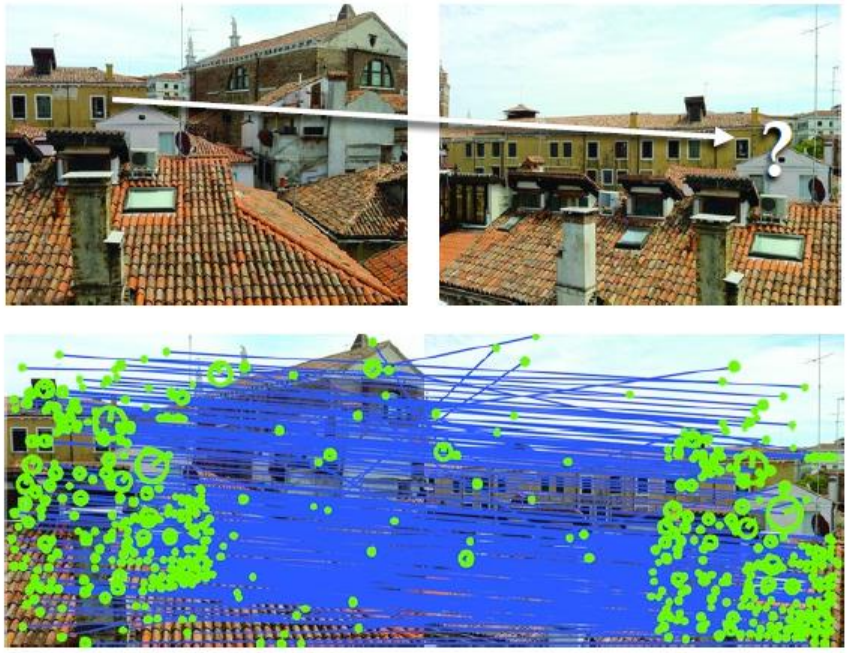


Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

4 inliers (red, yellow, orange, brown),

When the transform is created, and when these 4 points are transformed from left to right images, they fit well between both images

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

4 outliers (blue, light blue, purple, pink)

When the transform is created, and when these 4 points are transformed from left to right images, they Don’t fit well between both images

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

4 inliers (red, yellow, orange, brown),
4 outliers (blue, light blue, purple, pink)

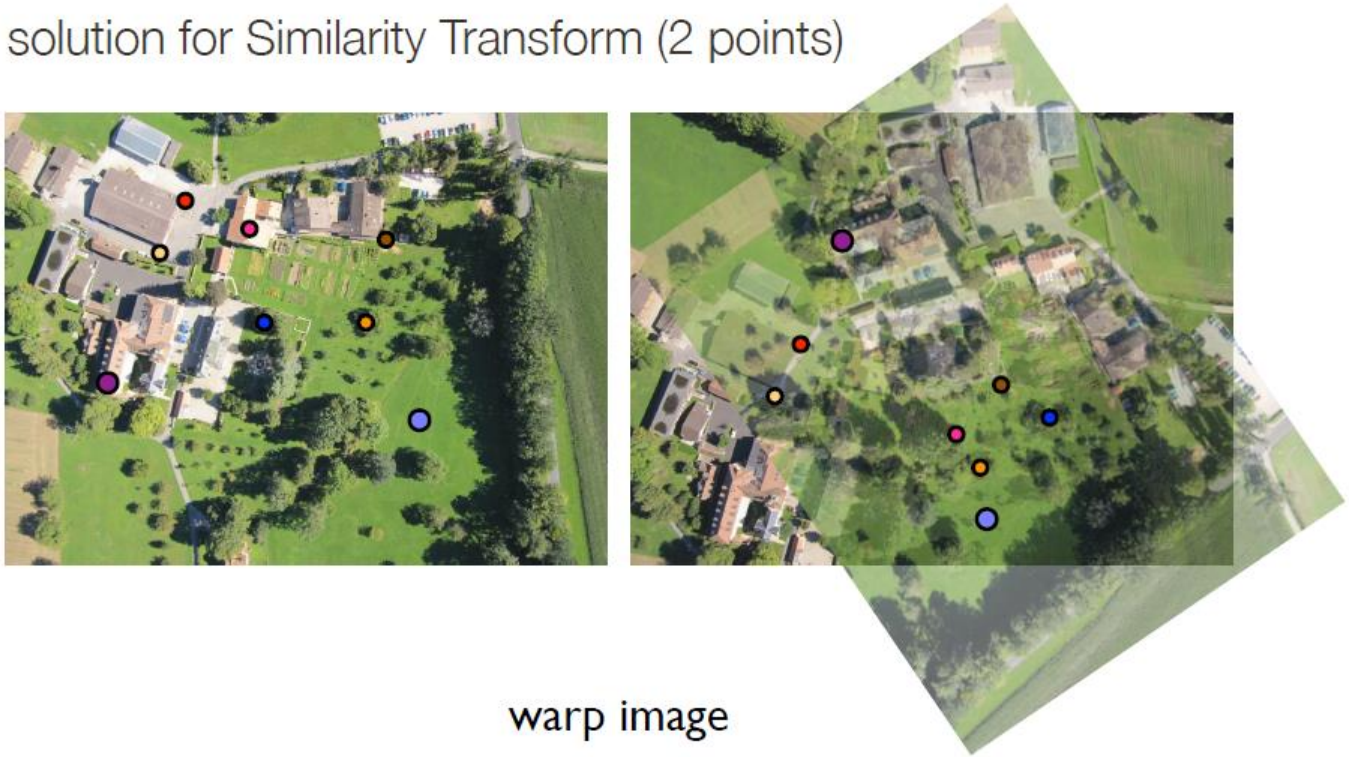
Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

choose light blue, purple

Image **Alignment + RANSAC**

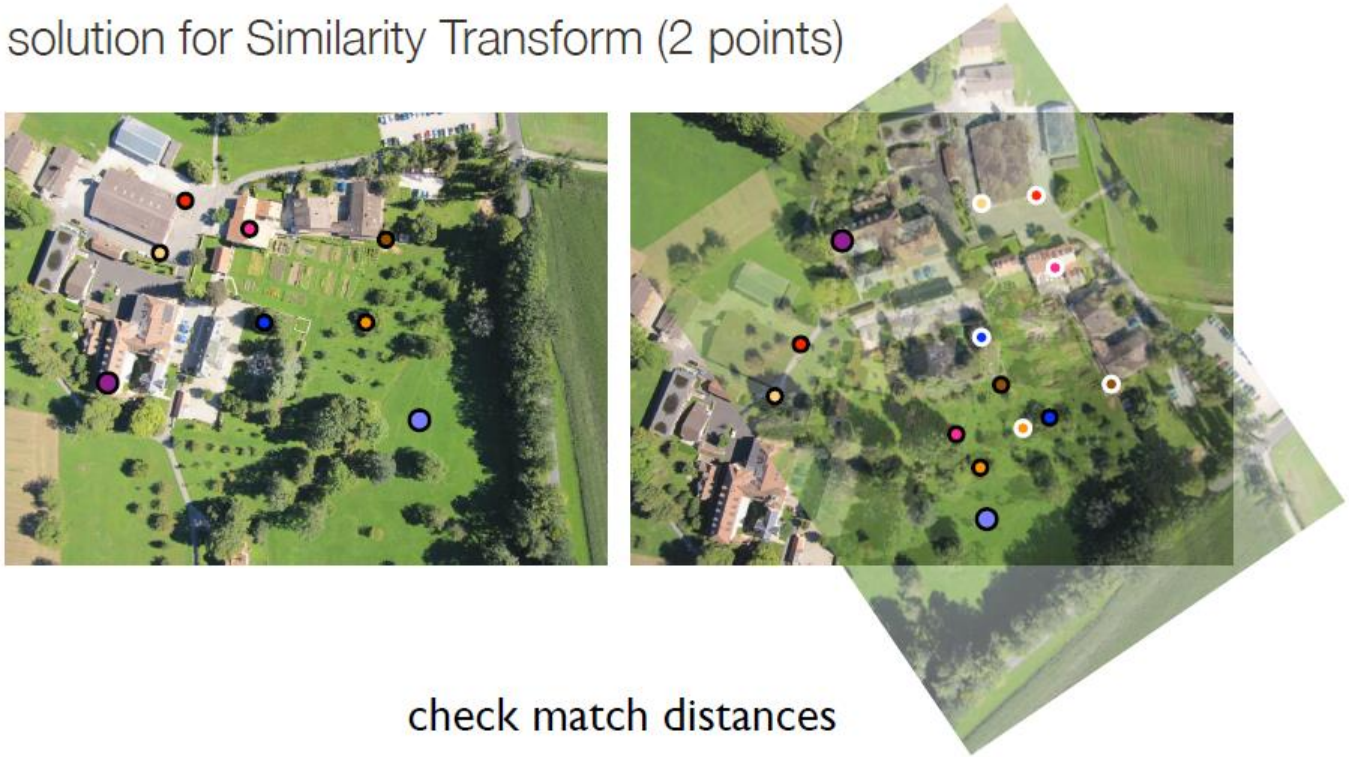
RANSAC solution for Similarity Transform (2 points)



warp image

Image **Alignment + RANSAC**

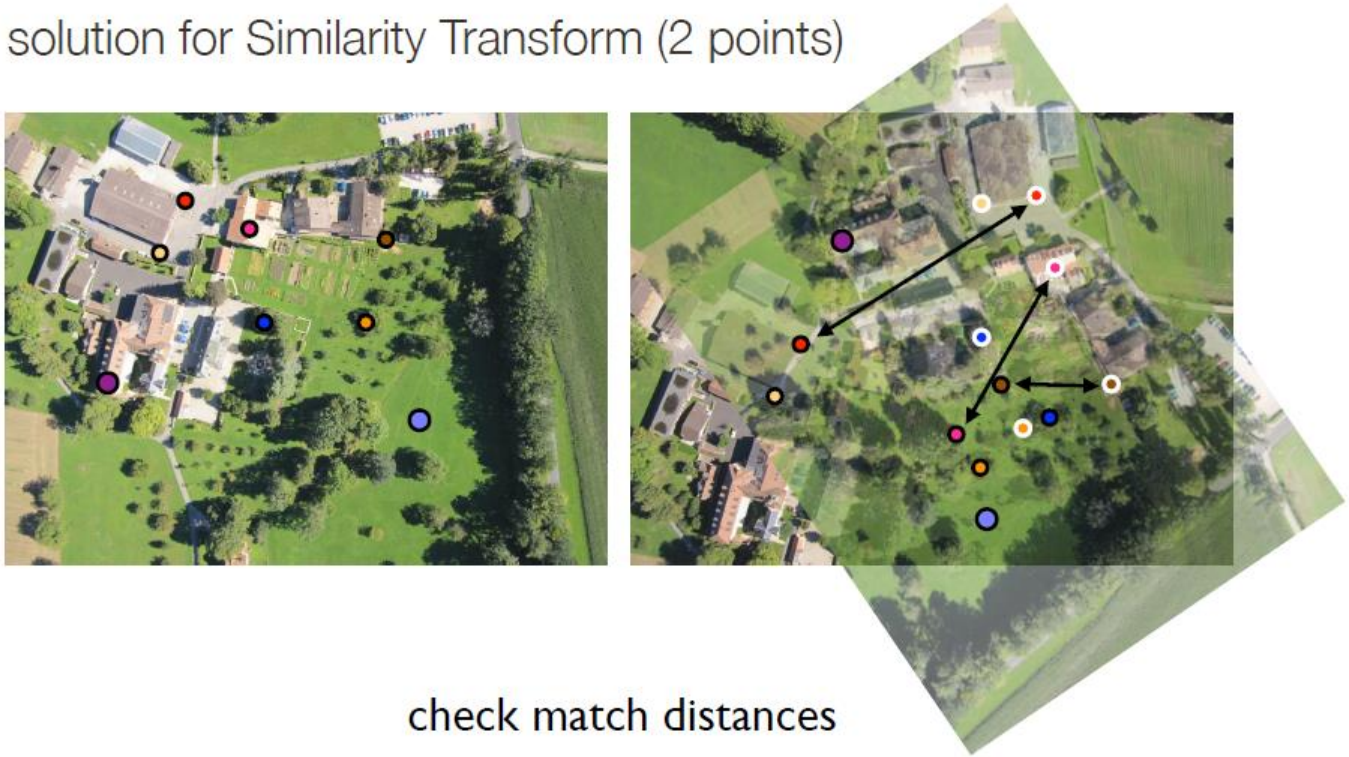
RANSAC solution for Similarity Transform (2 points)



check match distances

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)



check match distances

#inliers = 2

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

choose red, orange

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

warp image

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

check match distances

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

check match distances

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

check match distances

#inliers = 4

Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

Same approach can be done for affine or projective transformation but with 3 pairs and 4 pairs respectively