Visual Navigation for Flying Robots

Structure From Motion

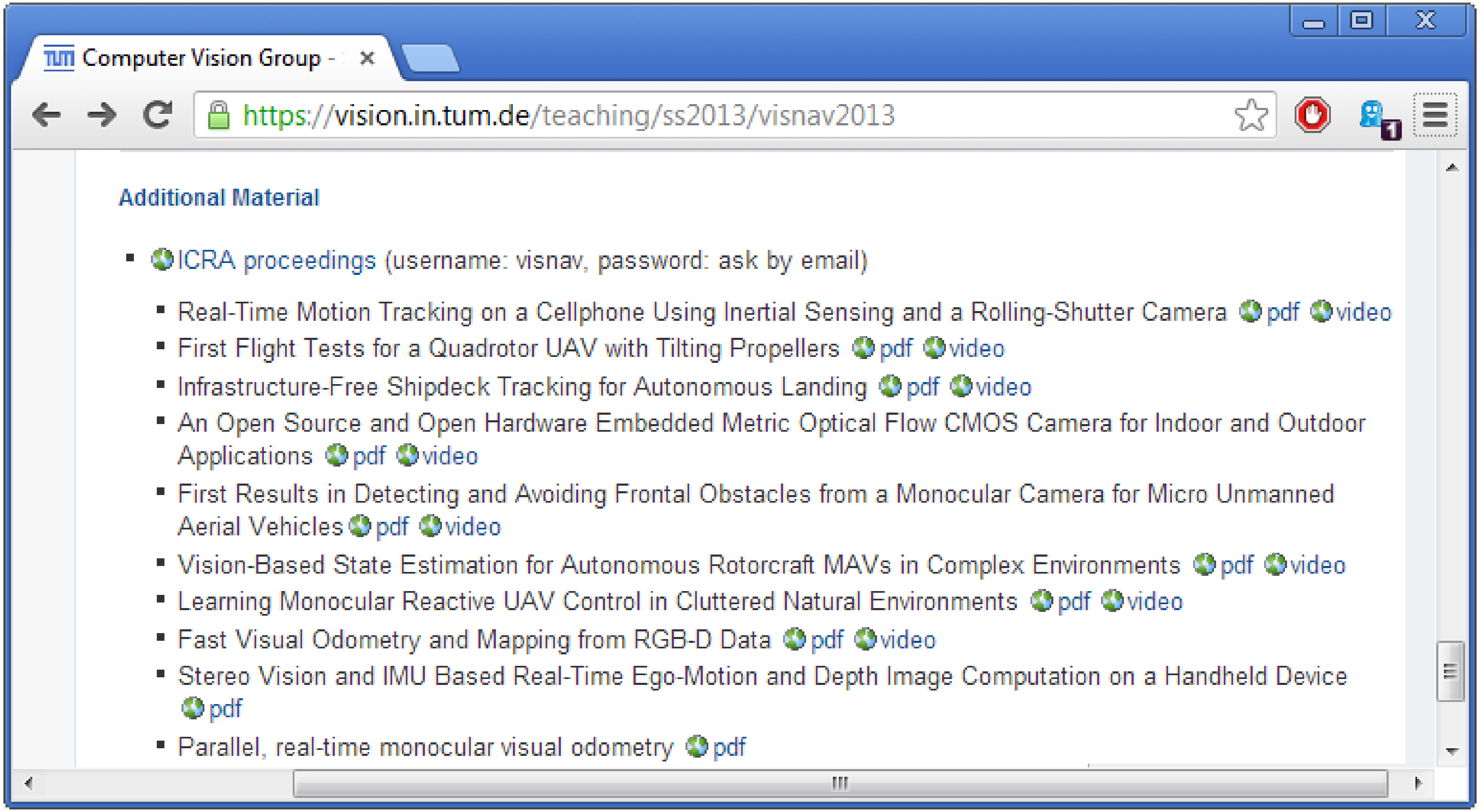
Dr. Jürgen Sturm

VISNAV Oral Team Exam

Date and Time	Student Name	Student Name	Student Name
Mon, July 29, 10am			
Mon, July 29, 11am			
Mon, July 29, 2pm			
Mon, July 29, 3pm			
Mon, July 29, 4pm			
Tue, July 30, 10am			
Tue, July 30, 11am			
Tue, July 30, 2pm			
Tue, July 30, 3pm			
Tue, July 30, 4pm			

Will put up this list in front of our secretary's office (02.09.052)

ICRA Papers+Videos are Online



Additional Material

- ICRA proceedings (username: visnav, password: ask by email)
  - Real-Time Motion Tracking on a Cellphone Using Inertial Sensing and a Rolling-Shutter Camera pdf video
  - First Flight Tests for a Quadrotor UAV with Tilting Propellers pdf video
  - Infrastructure-Free Shipdeck Tracking for Autonomous Landing pdf video
  - An Open Source and Open Hardware Embedded Metric Optical Flow CMOS Camera for Indoor and Outdoor Applications pdf video
  - First Results in Detecting and Avoiding Frontal Obstacles from a Monocular Camera for Micro Unmanned Aerial Vehicles pdf video
  - Vision-Based State Estimation for Autonomous Rotorcraft MAVs in Complex Environments pdf video
  - Learning Monocular Reactive UAV Control in Cluttered Natural Environments pdf video
  - Fast Visual Odometry and Mapping from RGB-D Data pdf video
  - Stereo Vision and IMU Based Real-Time Ego-Motion and Depth Image Computation on a Handheld Device pdf
  - Parallel, real-time monocular visual odometry pdf

Agenda for Today

- **This week:** basic ingredients of a visual SLAM system
  - Feature detection, descriptors and matching
  - Place recognition
  - 3D motion estimation
- **Next week:** bundle adjustment, graph SLAM, stereo cameras, Kinect
- **In two weeks:** map representations, mapping and (dense) 3D reconstruction

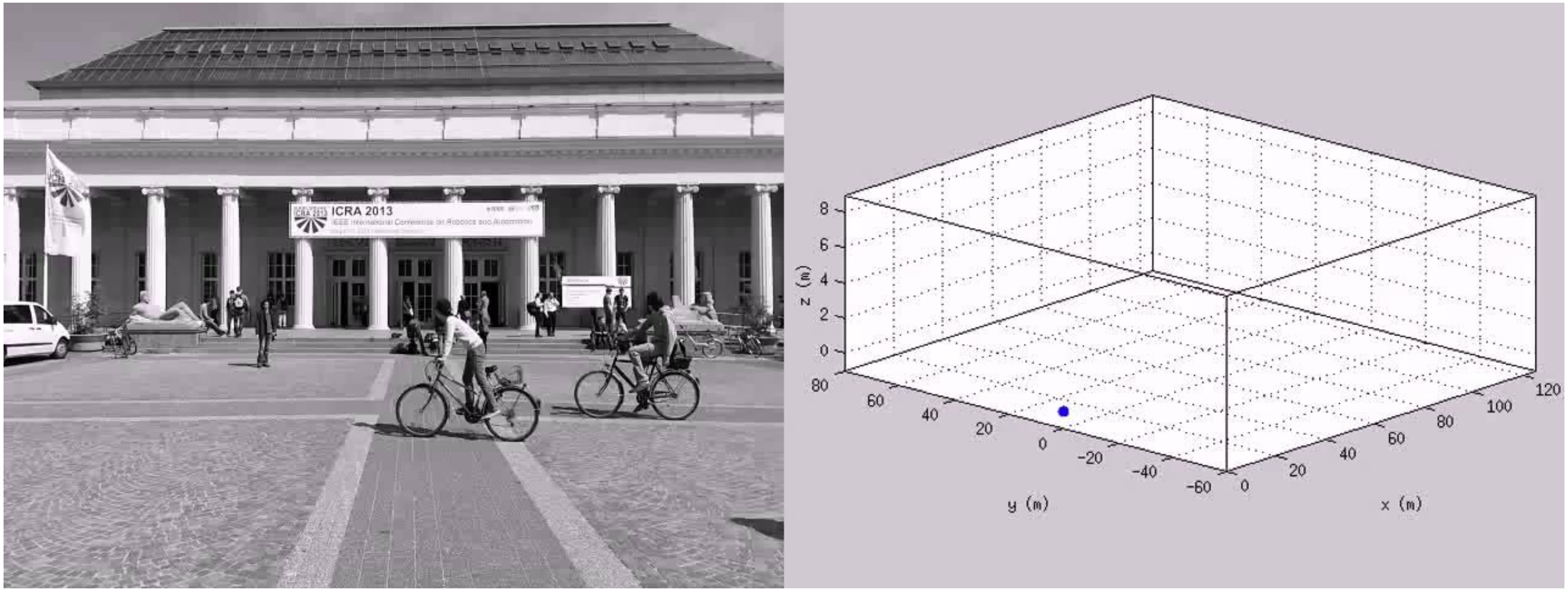
Last week: KLT Tracker

Kanade-Lucas-Tomasi (KLT) Tracker

- Algorithm
  1. Find (Shi-Tomasi) corners in first frame and initialize tracks
  2. Track from frame to frame
  3. Delete track if error exceeds threshold
  4. Initialize additional tracks when necessary
  5. Repeat step 2-4
- KLT tracker is highly efficient (real-time on CPU) but provides only sparse motion vectors
- Can use coarse-to-fine for larger motions

Visual Odometry

[Li et al., ICRA ‘13]



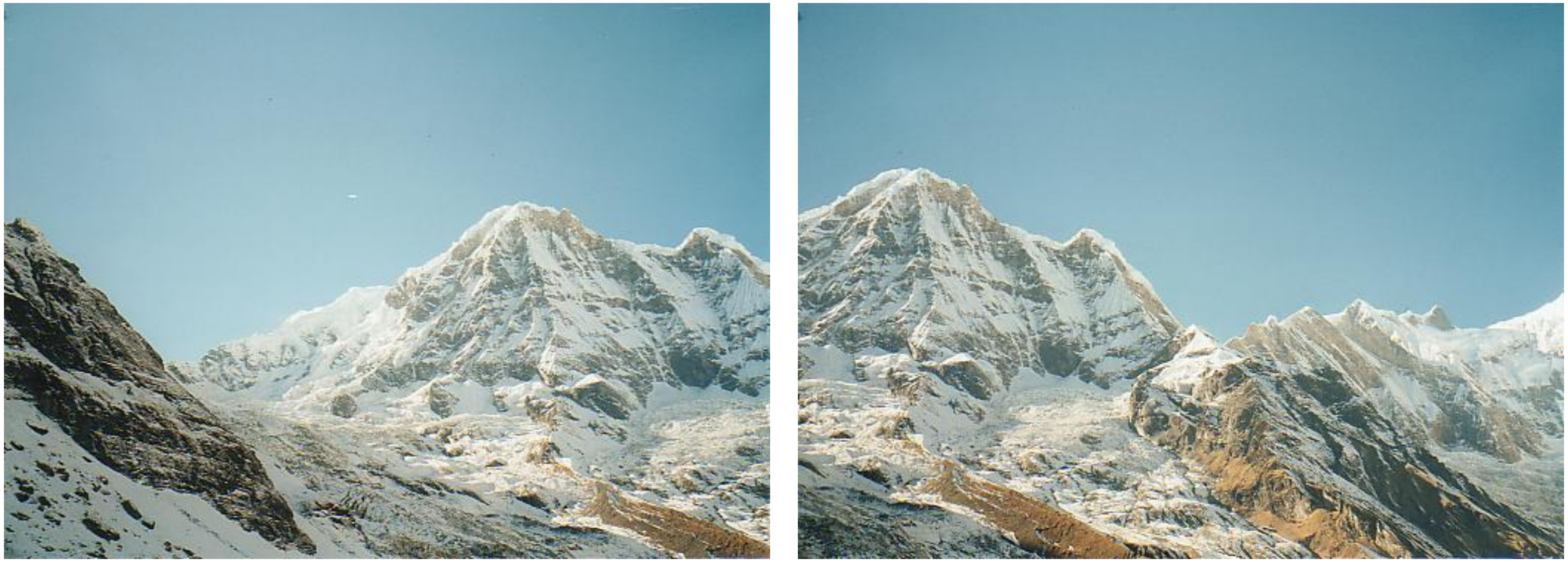
Limitations

- Tracking is based on image gradients (dx/dy/dt)
  - Only works for small motions
  - Preferably high frame rate
- Cannot recover when tracks are lost

- How can we recognize previously seen patches?

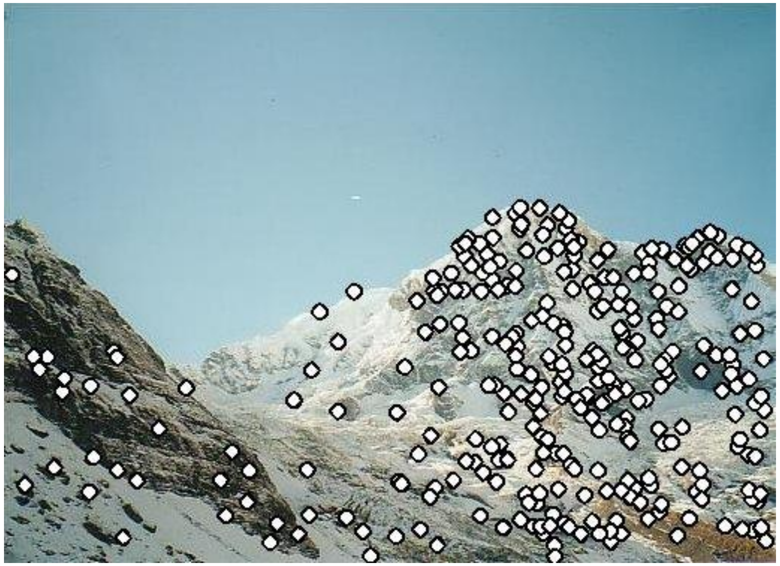
Example: How to Build a Panorama Map

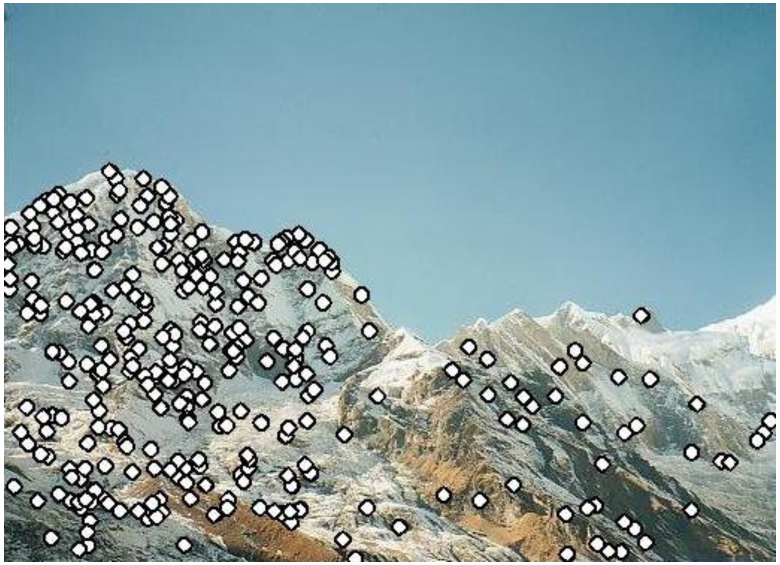
- We need to match (align) images
- Global methods sensitive to occlusion, lighting, parallax effects
- How would you do it by eye?



Matching with Features

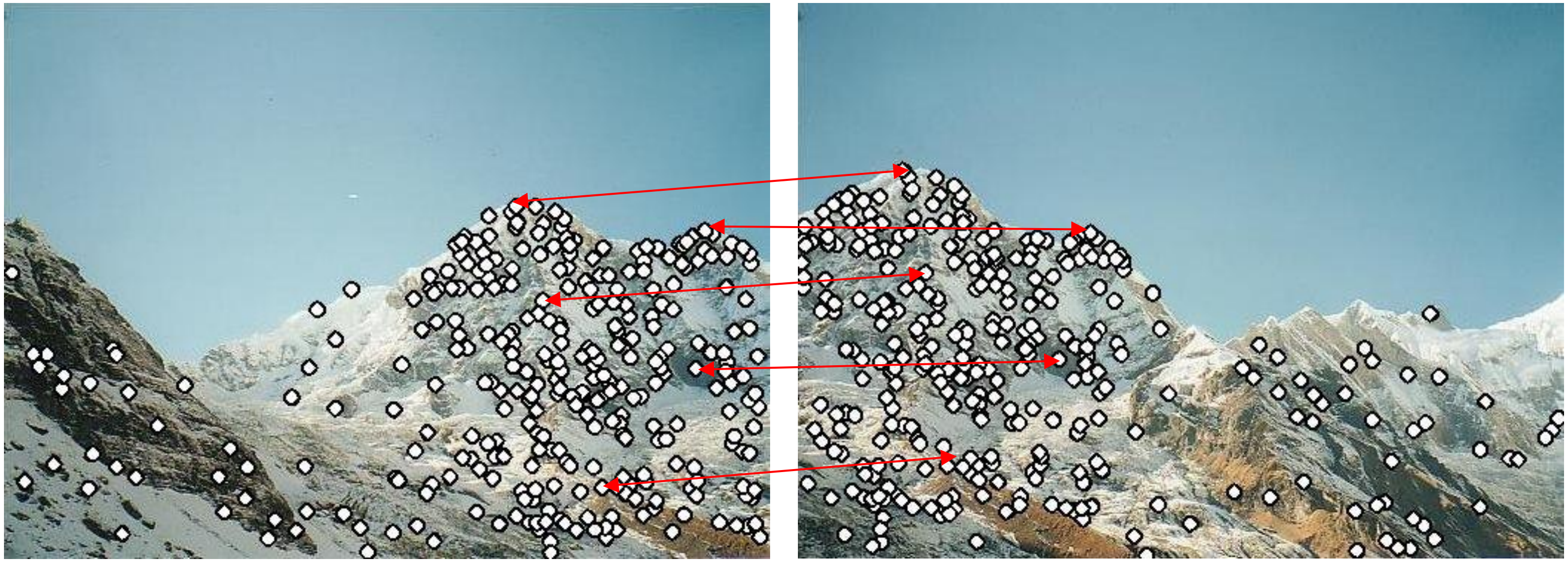
- Detect features in both images





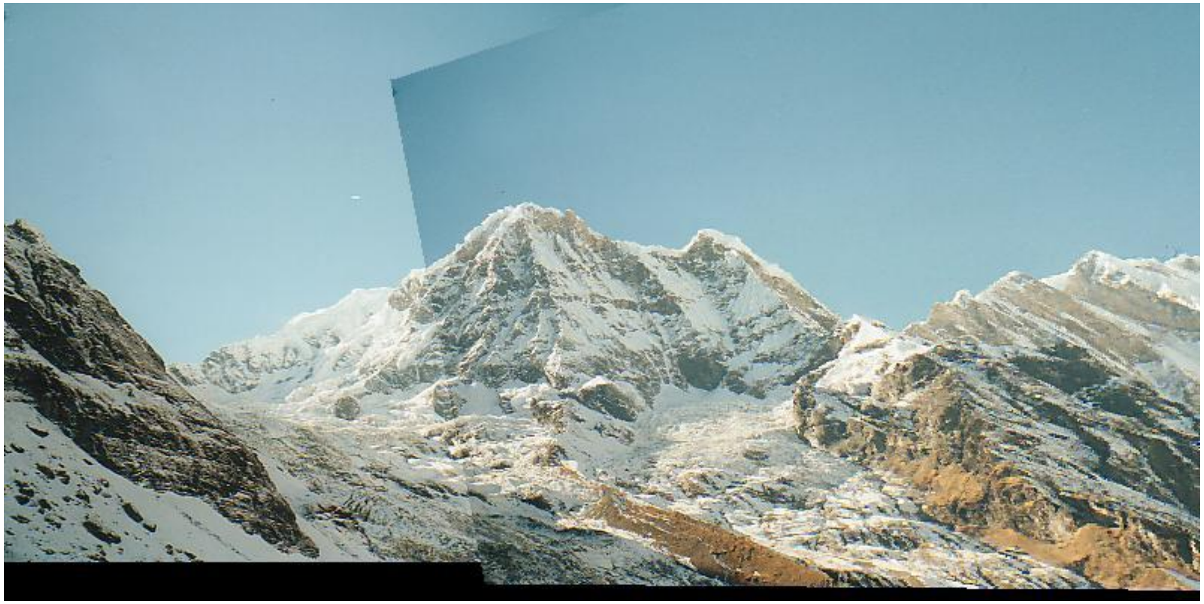
Matching with Features

- Detect features in both images
- Find corresponding pairs



Matching with Features

- Detect features in both images
- Find corresponding pairs
- Use these pairs to align images



Matching with Features

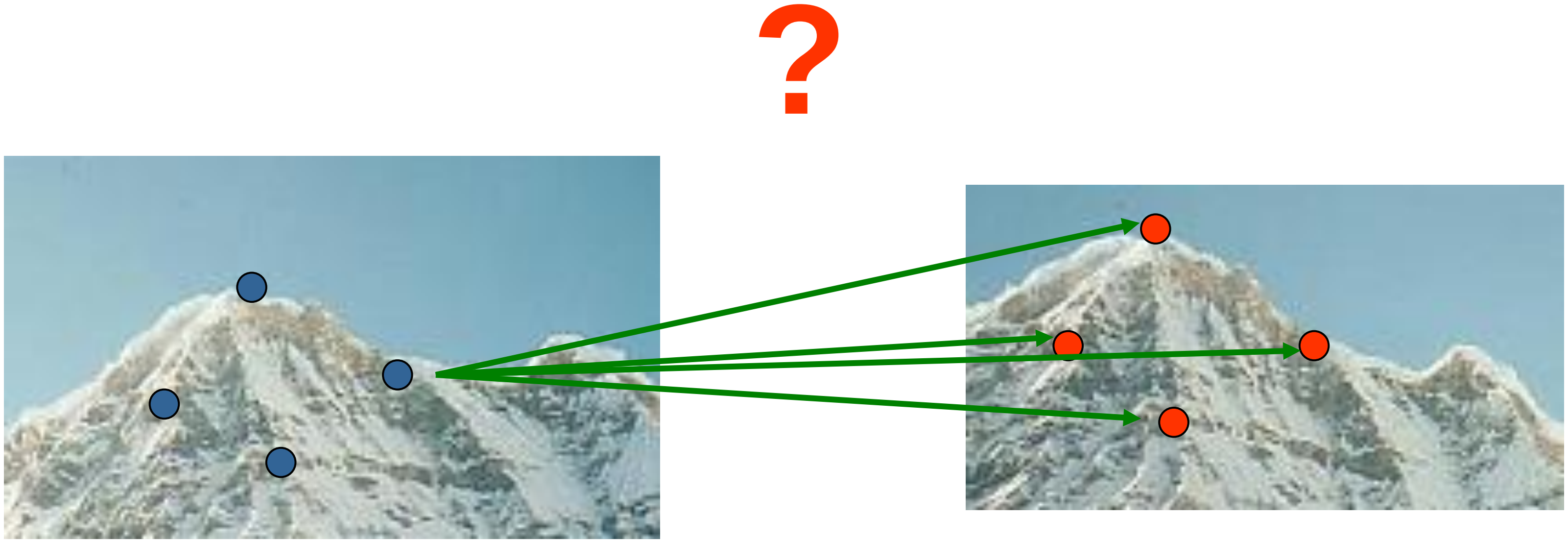
- Problem 1:
  We need to detect the **same** point **independently** in both images

no chance to match!

→ We need a reliable detector

Matching with Features

- Problem 2:
  For each point correctly recognize the corresponding one



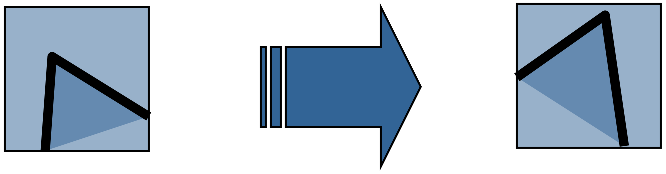
→ We need a reliable and distinctive descriptor

Ideal Feature Detector

- Always finds the same point on an object, regardless of changes to the image
- Insensitive (invariant) to changes in:
  - Scale
  - Lightning
  - Perspective imaging
  - Partial occlusion

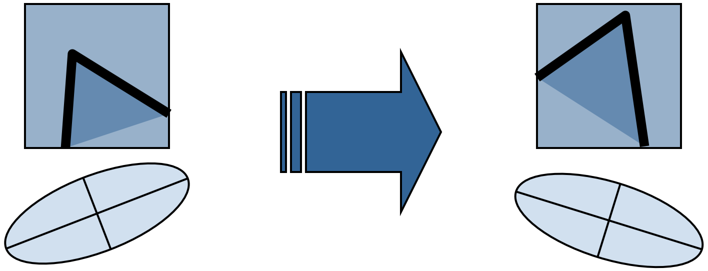
Harris Detector

- Rotation invariance?



Harris Detector

- Rotation invariance?

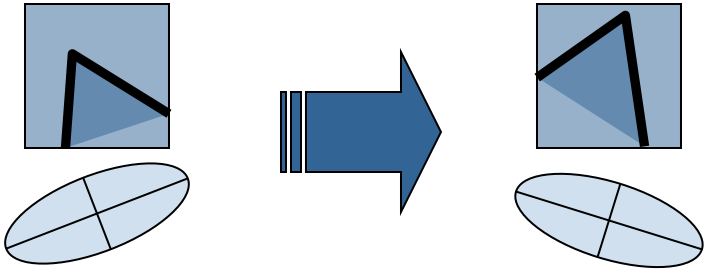


- Remember from last week

$$A = \begin{pmatrix} \sum f_x^2 & \sum f_x f_y \\ \sum f_x f_y & \sum f_y^2 \end{pmatrix} \qquad R = \lambda_1 \lambda_2 - \kappa \left(\lambda_1 + \lambda_2\right)^2$$

Harris Detector

- Rotation invariance



- Remember from last week

$$A = \begin{pmatrix} \sum f_x^2 & \sum f_x f_y \\ \sum f_x f_y & \sum f_y^2 \end{pmatrix} \qquad R = \lambda_1 \lambda_2 - \kappa \left(\lambda_1 + \lambda_2\right)^2$$

- Ellipse rotates but its shape (i.e. eigenvalues) remains the same

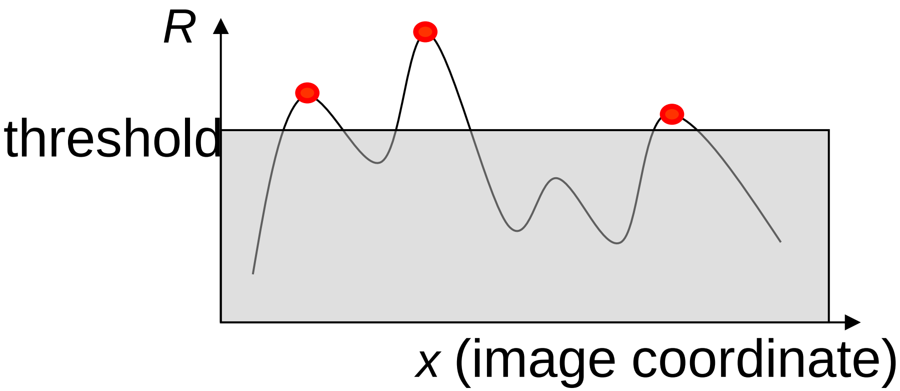
→ Corner response R is invariant to rotation

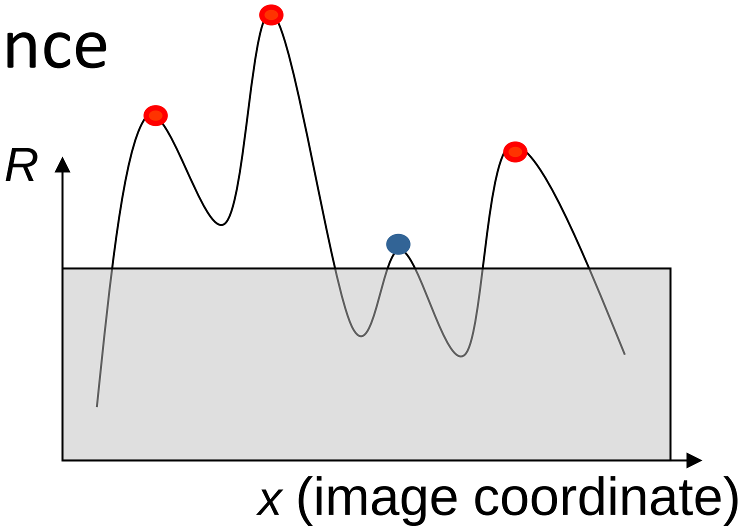
Harris Detector

- Invariance to intensity change?

Harris Detector

- Partial invariance to additive and multiplicative intensity changes
  - Only derivatives are used → invariance to intensity shift $I \rightarrow I + b$
  - Intensity scale $I \rightarrow aI$:
    Because of fixed intensity threshold on local maxima, only partial invariance



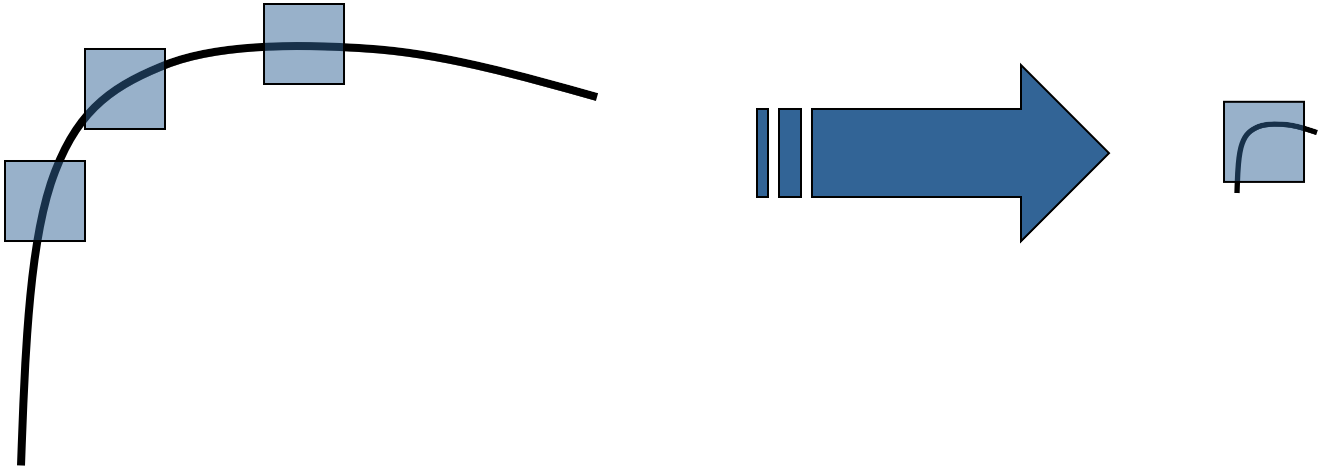


Harris Detector

- Invariant to scaling?

Harris Detector

- Not invariant to image scale

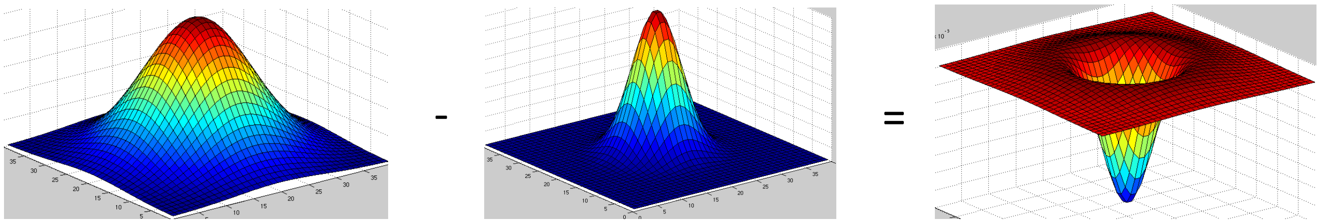


All points classified as edge

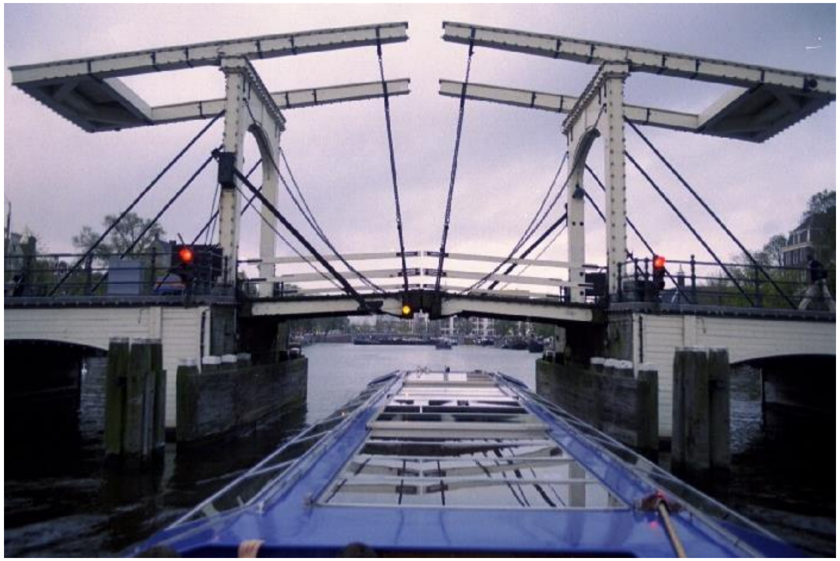
Point classified as corner

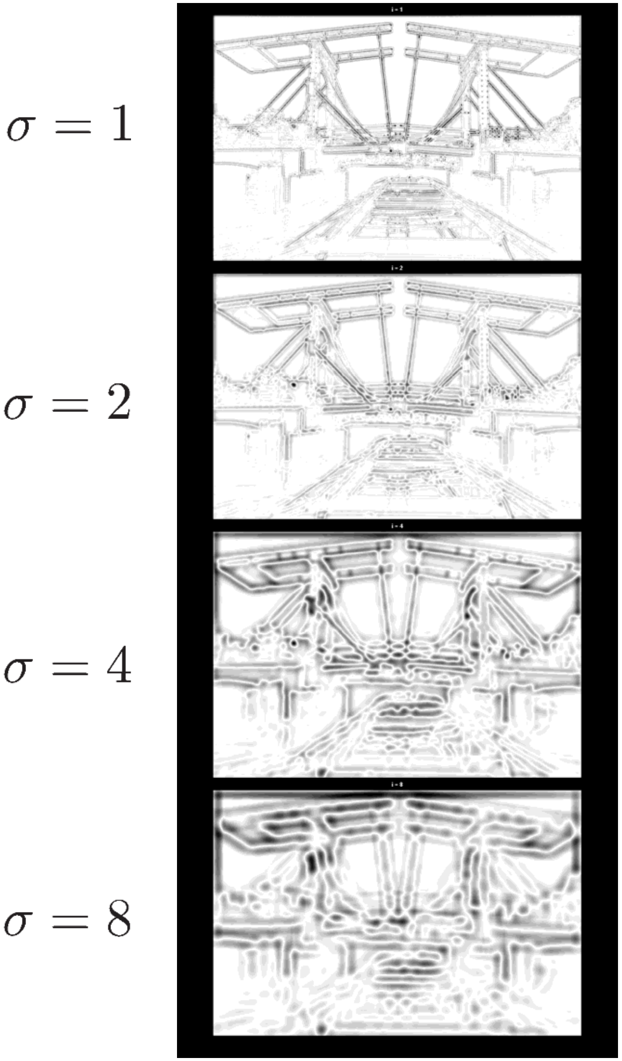
Difference Of Gaussians (DoG)

- Alternative corner detector that is additionally invariant to scale change
- Approach:
  - Run linear filter (diff. of two Gaussians, $\sigma_1 = 2\sigma_2$)
  - Do this at different scales
  - Search for a maximum both in space and scale



Example: Difference of Gaussians





$\sigma = 1$

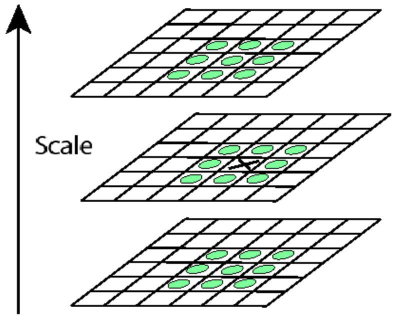
$\sigma = 2$

$\sigma = 4$

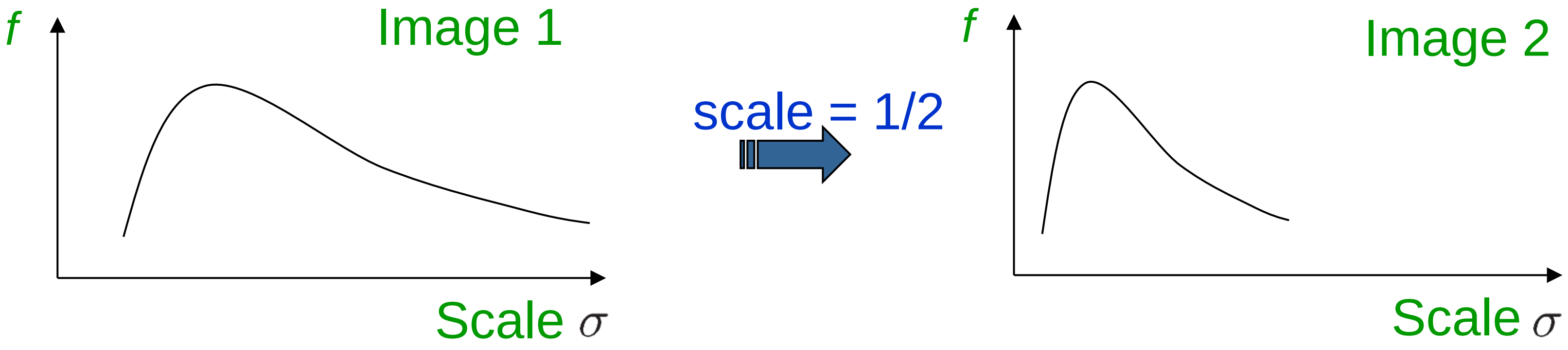
$\sigma = 8$

SIFT Detector

- Search for local maximum in space and scale



- Corner detections are invariant to scale change



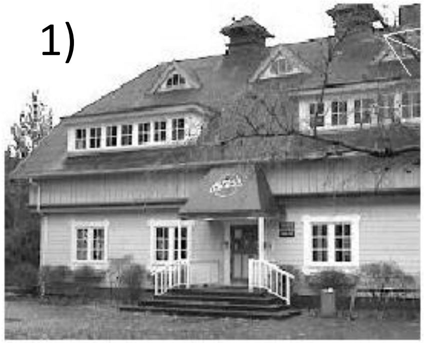
SIFT Detector

1. Detect maxima in scale-space
2. Non-maximum suppression
3. Eliminate edge points (check ratio of eigenvalues)
4. For each maximum, fit quadratic function and compute center at sub-pixel accuracy

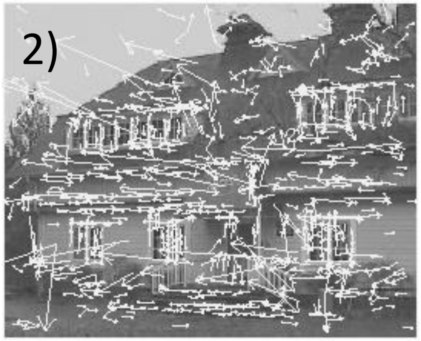
Example

1. Input image 233x189 pixel
2. 832 candidates DoG minima/maxima (visualization indicate scale, orient., location)
3. 536 keypoints remain after thresholding on minimum contrast and principal curvature

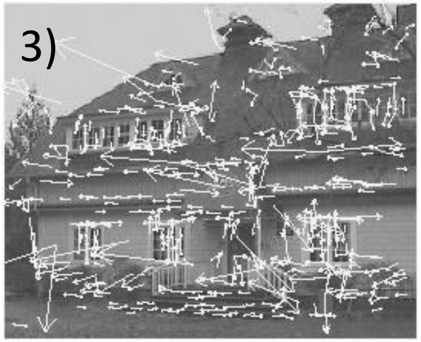
1)



2)

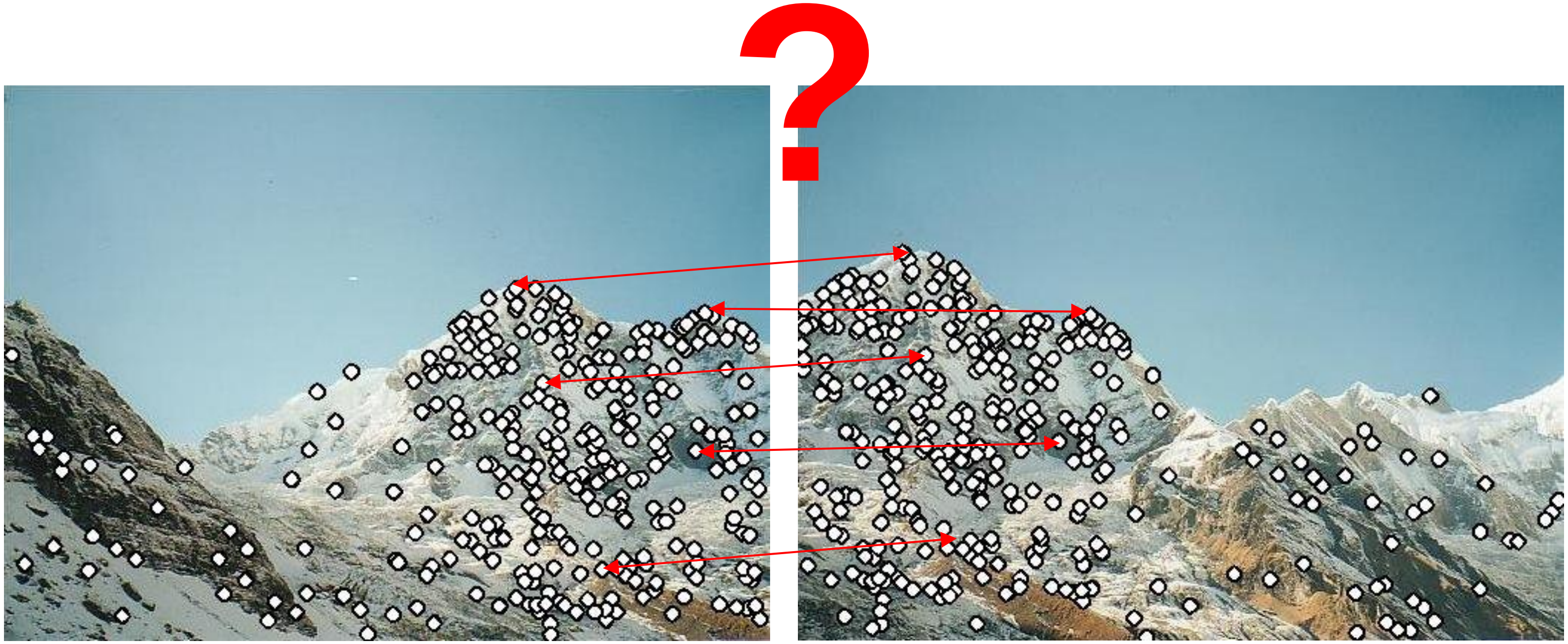


3)



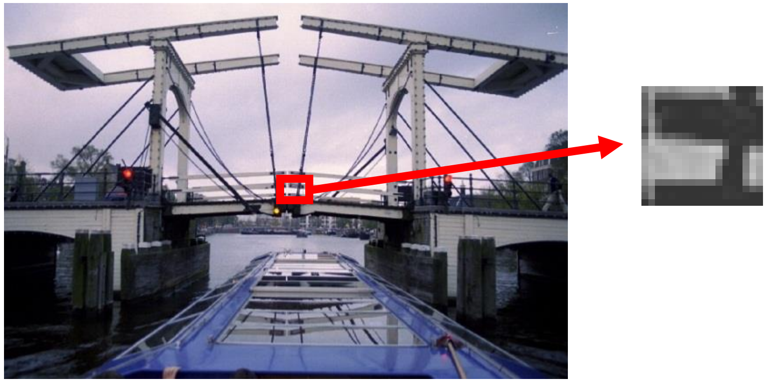
Feature Matching

- Now, we know how to find **repeatable** corners
- Next question: How can we match them?

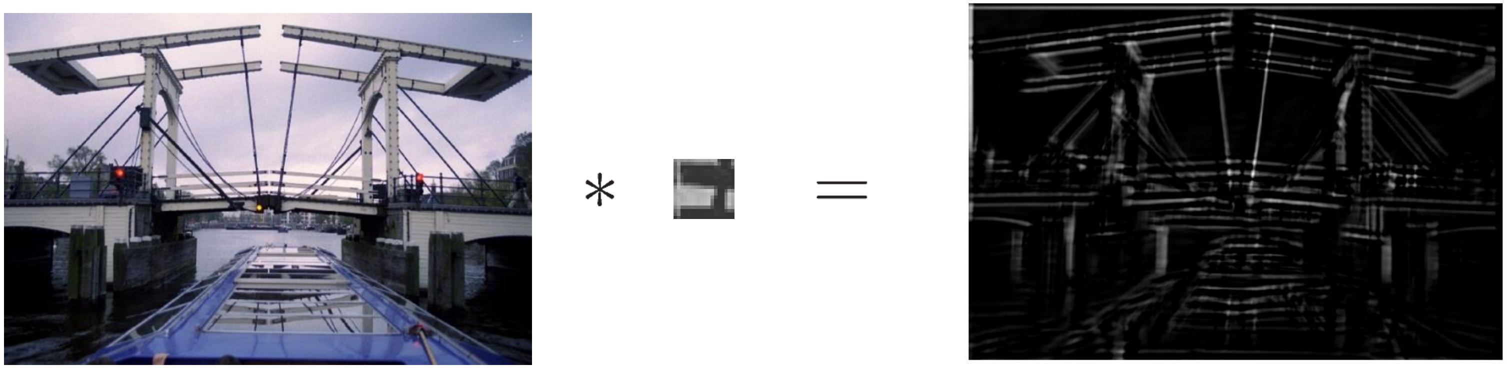


Template Convolution

- Extract a small as a template



- Convolve image with this template



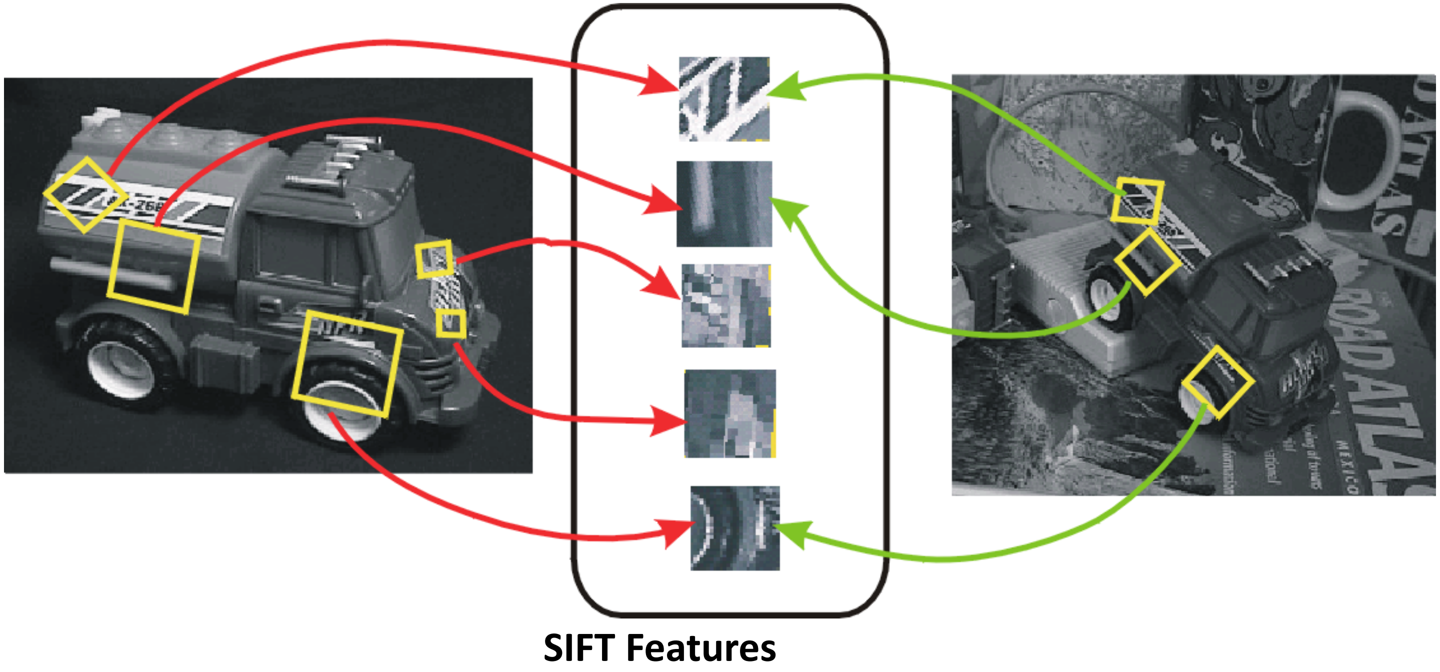
Template Convolution

Invariances

- Scaling: No
- Rotation: No (maybe rotate template?)
- Illumination: No (use bias/gain model?)
- Perspective projection: Not really

Scale Invariant Feature Transform (SIFT)

- Lowe, 2004: Transform patches into a canonical form that is invariant to translation, rotation, scale, and other imaging parameters



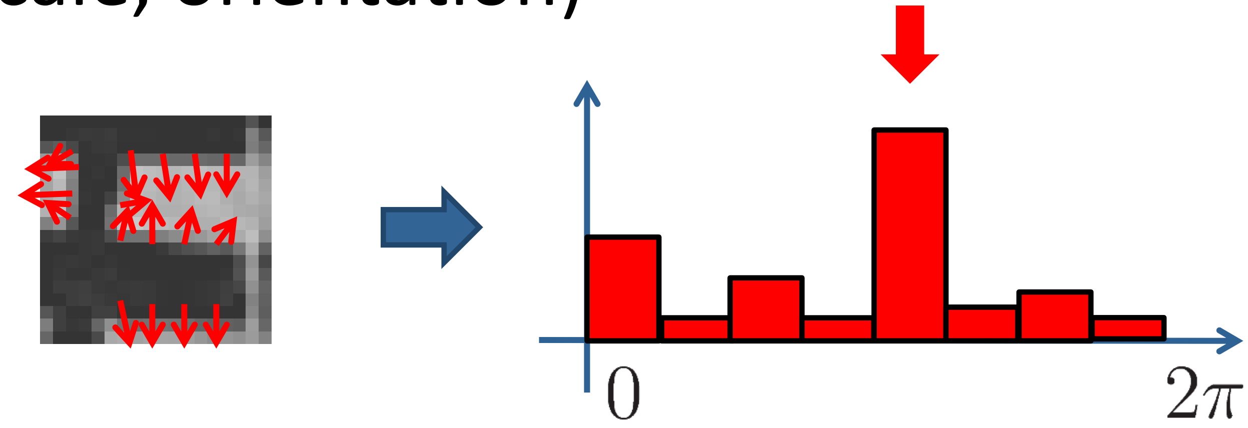
Scale Invariant Feature Transform (SIFT)

Approach

1. Find SIFT corners (position + scale)
2. Find dominant orientation and de-rotate patch
3. Extract SIFT descriptor (histograms over gradient directions)

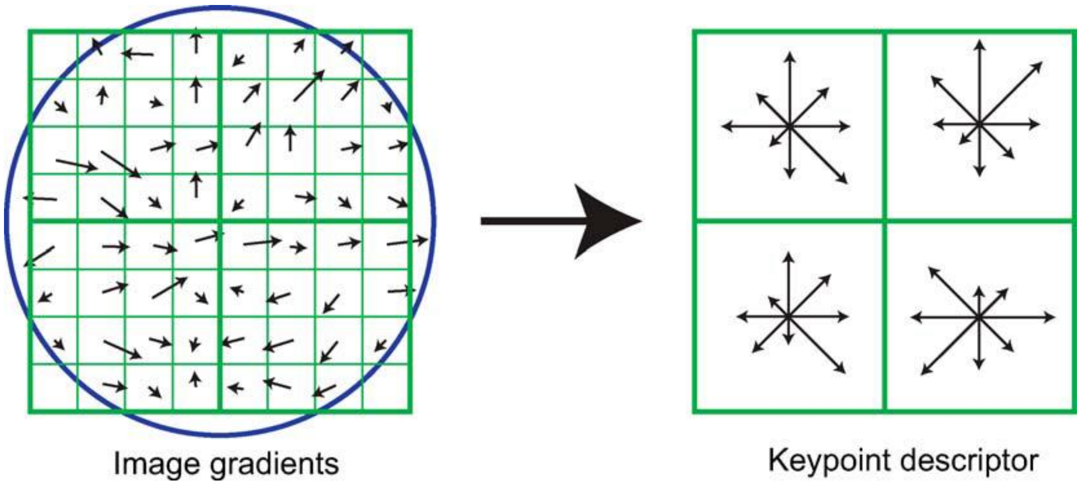
Select Dominant Orientation

- Create a histogram of local gradient directions computed at selected scale (36 bins)
- Assign canonical orientation at peak of smoothed histogram
- Each key now specifies stable 2D coordinates (x, y, scale, orientation)



SIFT Descriptor

- Compute image gradients over 16x16 window (green), weight with Gaussian kernel (blue)
- Create 4x4 arrays of orientation histograms, each consisting of 8 bins
- In total, SIFT descriptor has 128 dimensions



Feature Matching

Given features in I_1, how to find best match in I_2?

- Define distance function that compares two features
- Test all the features in I_2, find the one with the minimal distance

Feature Distance

How to define the difference between features?

- Simple approach is Euclidean distance (or SSD)

$$\mathrm{d}(\mathbf{d}_1, \mathbf{d}_2) = \|\mathbf{d}_1 - \mathbf{d}_2\|$$

Feature Distance

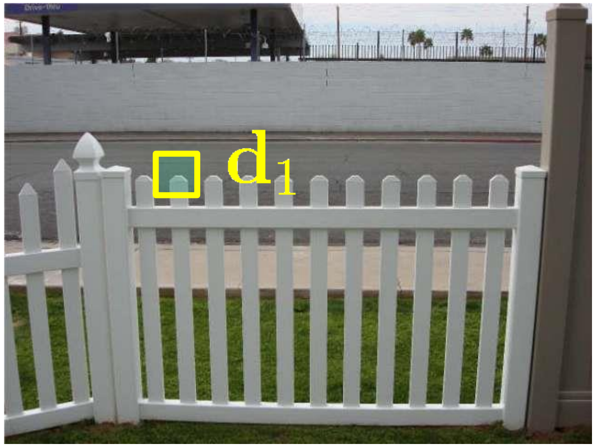
How to define the difference between features?

- Simple approach is Euclidean distance (or SSD)

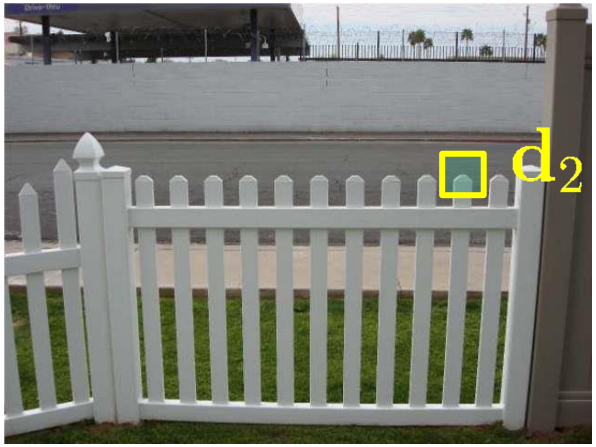
$$\mathrm{d}(\mathbf{d}_1, \mathbf{d}_2) = \|\mathbf{d}_1 - \mathbf{d}_2\|$$

- Problem: can give good scores to ambiguous (bad) matches

I_1



I_2

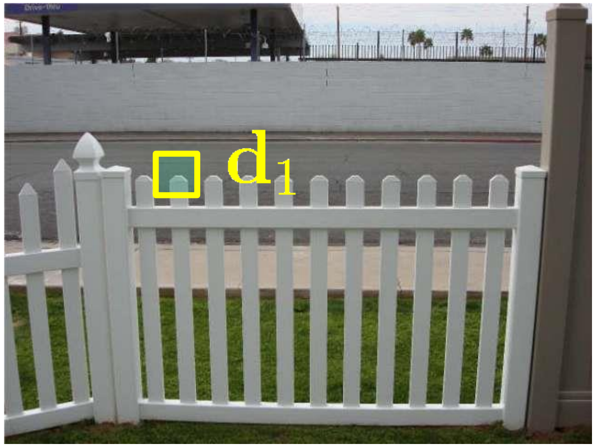


Feature Distance

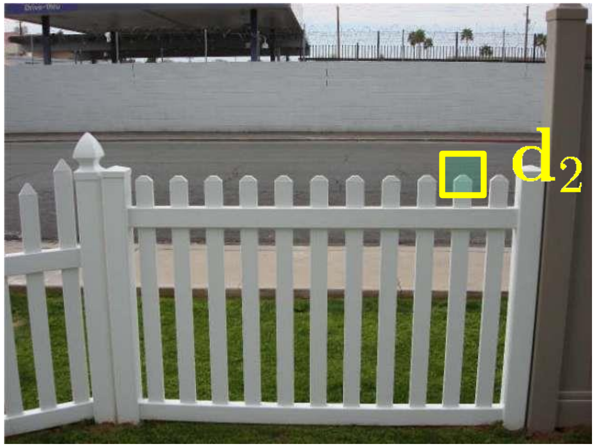
How to define the difference between features?

- Better approach $\mathrm{d}(\mathbf{d}_1, \mathbf{d}_2) = \|\mathbf{d}_1 - \mathbf{d}_2\| / \|\mathbf{d}_1 - \mathbf{d}'_2\|$
  with $\mathbf{d}_2$ best matching feature from I_2
  $\mathbf{d}'_2$ second best matching feature from I_2
- Gives small values for ambiguous matches

I_1



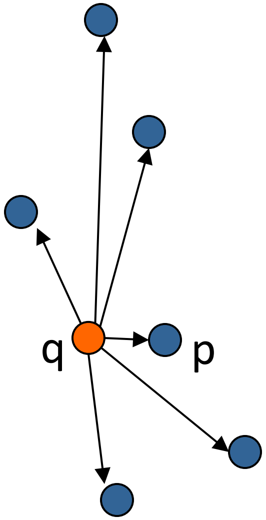
I_2



Efficient Matching

For feature matching, we need to answer a large number of **nearest neighbor queries**

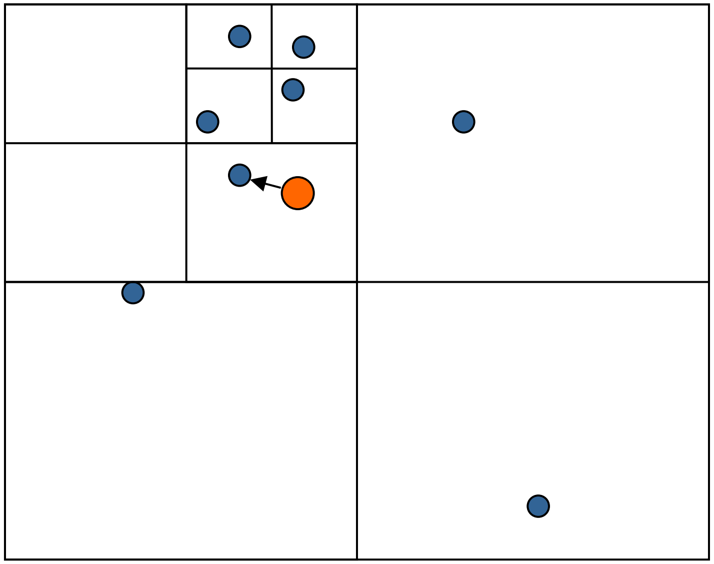
- Exhaustive search $\mathrm{O}(n^2)$



Efficient Matching

For feature matching, we need to answer a large number of **nearest neighbor queries**

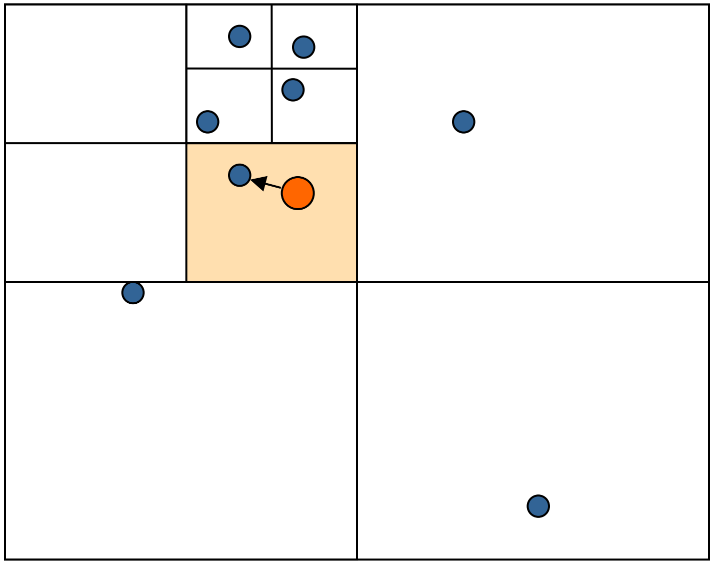
- Exhaustive search $\mathrm{O}(n^2)$
- Indexing (k-d tree)



Efficient Matching

For feature matching, we need to answer a large number of **nearest neighbor queries**

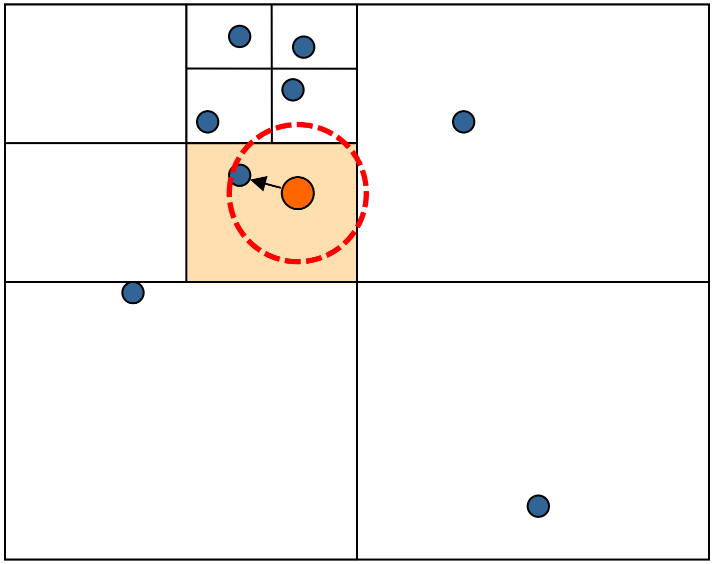
- Exhaustive search $O(n^2)$
- Indexing (k-d tree)
    - Localize query in tree
    - Search nearby leaves until nearest neighbor is guaranteed found



Efficient Matching

For feature matching, we need to answer a large number of **nearest neighbor queries**

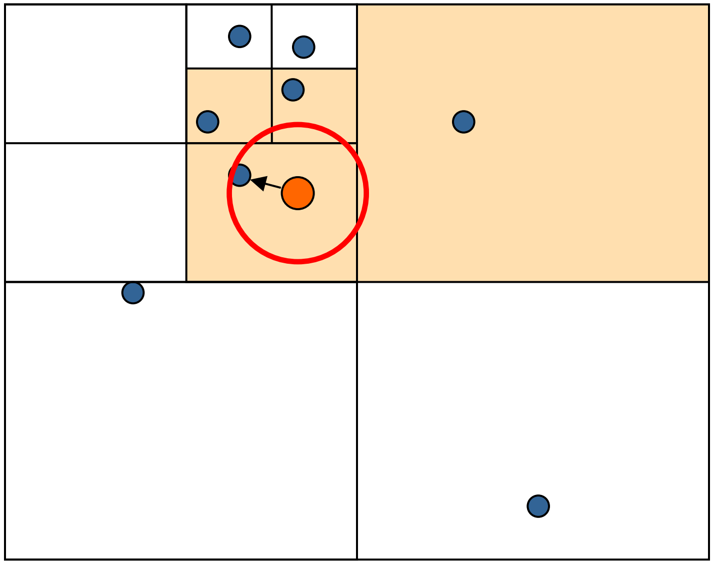
- Exhaustive search $\mathrm{O}(n^2)$
- Indexing (k-d tree)
  - Localize query in tree
  - Search nearby leaves until nearest neighbor is guaranteed found



Efficient Matching

For feature matching, we need to answer a large number of **nearest neighbor queries**

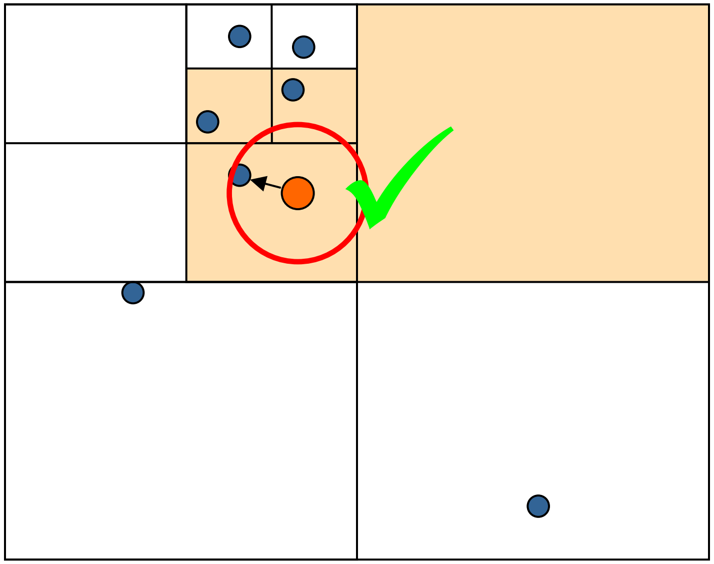
- Exhaustive search $O(n^2)$
- Indexing (k-d tree)
    - Localize query in tree
    - Search nearby leaves until nearest neighbor is guaranteed found



Efficient Matching

For feature matching, we need to answer a large number of **nearest neighbor queries**

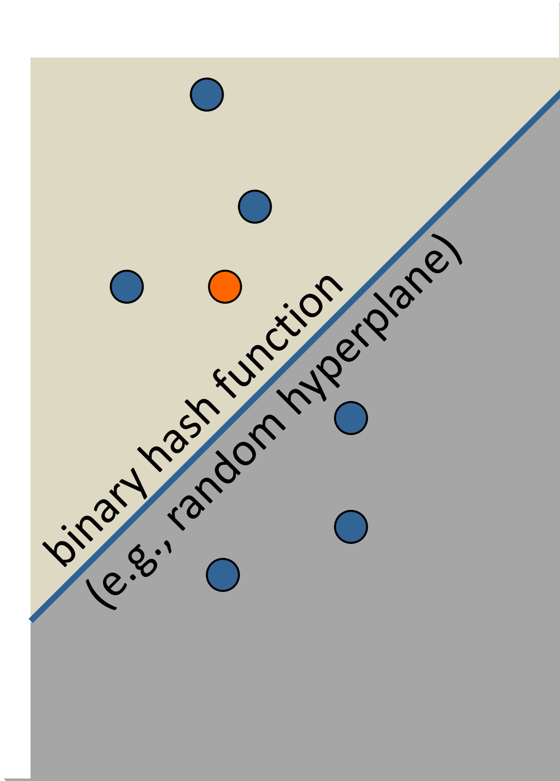
- Exhaustive search $\mathrm{O}(n^2)$
- Indexing (k-d tree)
  - Localize query in tree
  - Search nearby leaves until nearest neighbor is guaranteed found
  - Best-bin-first: use priority queue for unchecked leafs



Efficient Matching

For feature matching, we need to answer a large number of **nearest neighbor queries**

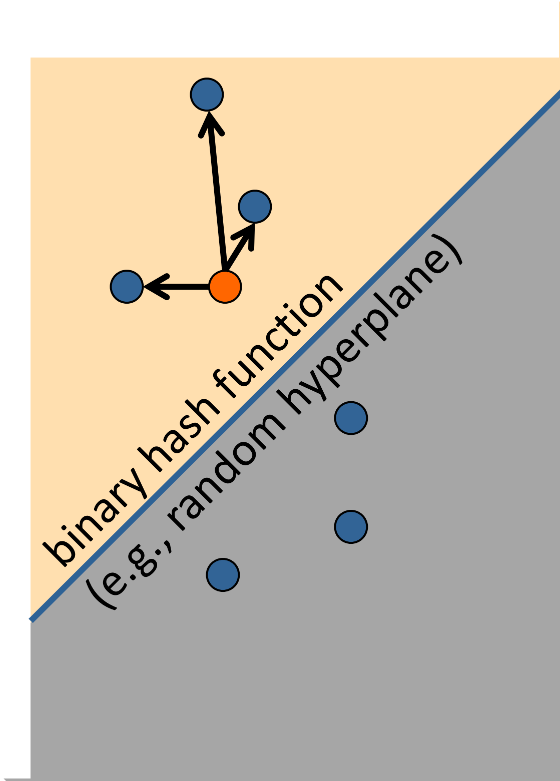
- Exhaustive search $O(n^2)$
- Indexing (k-d tree)
- Approximate search
  - Locality sensitive hashing
  - Approximate nearest neighbor



Efficient Matching

For feature matching, we need to answer a large number of **nearest neighbor queries**

- Exhaustive search $\mathrm{O}(n^2)$
- Indexing (k-d tree)
- Approximate search
  - Locality sensitive hashing
  - Approximate nearest neighbor



Efficient Matching

For feature matching, we need to answer a large number of **nearest neighbor queries**

- Exhaustive search $\mathrm{O}(n^2)$
- Indexing (k-d tree)
- Approximate search
- Vocabulary trees

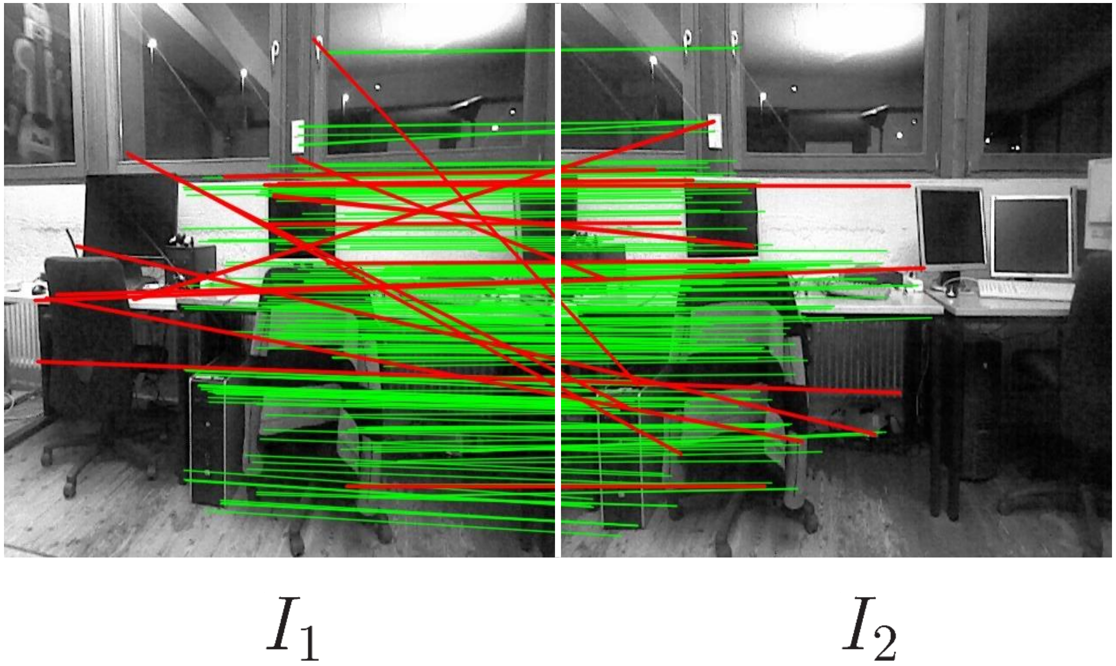
Other Descriptors (for intensity images)

- SIFT (Scale Invariant Feature Transform)
  [Lowe, 2004]
- SURF (Speeded Up Robust Feature)
  [Bay et al., 2008]
- BRIEF (Binary robust independent elementary features)
  [Calonder et al., 2010]
- ORB (Oriented FAST and Rotated Brief)
  [Rublee et al, 2011]
- …

Example: RGB-D SLAM

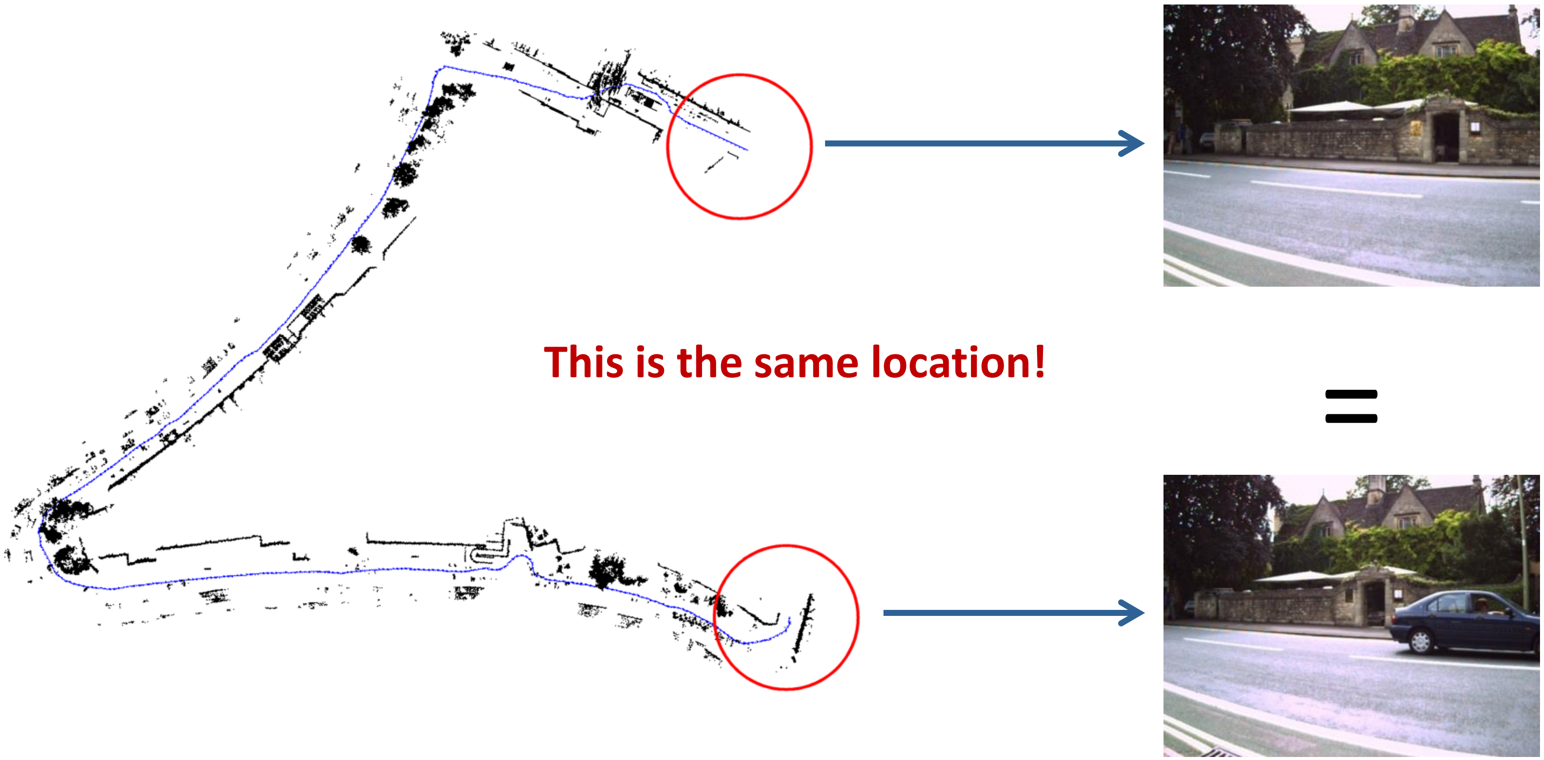
[Engelhard et al., 2011; Endres et al. 2012]

- Feature descriptor: SURF
- Feature matching: FLANN (approximate nearest neighbor)



Appearance-based Place Recognition

- How can we recognize that we have been visiting the same place before?

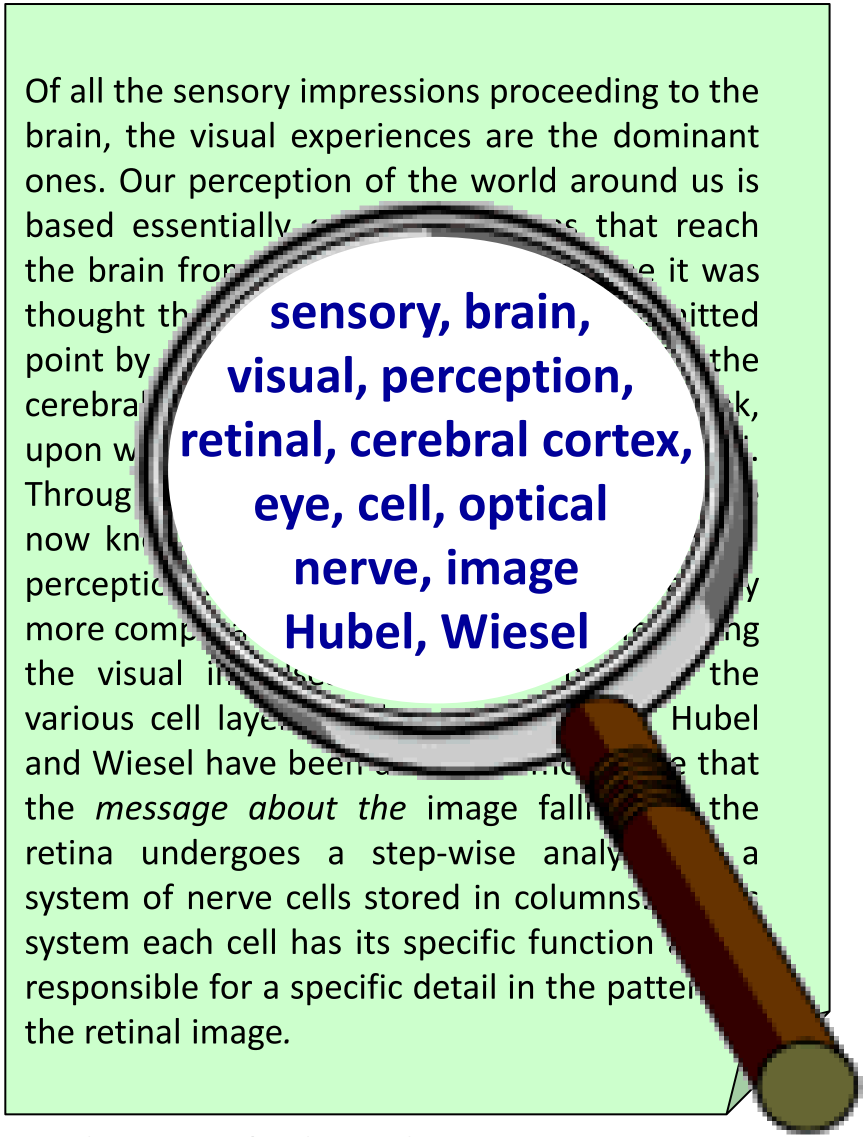


This is the same location!

Appearance-based Place Recognition

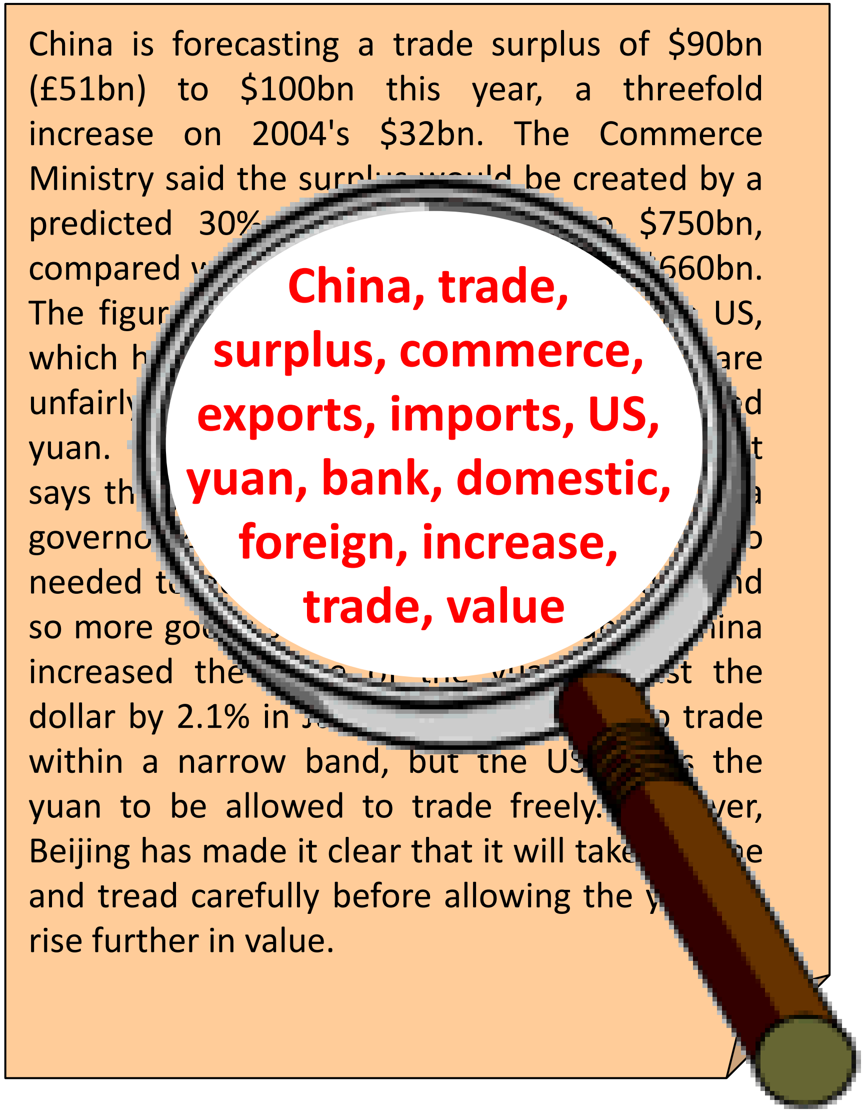
- Brute-force matching with all previous images is slow (why?)
- How can we do this faster?

Analogy to Document Retrieval



Of all the sensory impressions proceeding to the
brain, the visual experiences are the dominant
ones. Our perception of the world around us is
based essentially s that reach
the brain fro e it was
thought th itted
point by the
cerebra k,
upon w .
Throug
now kn
perceptio y
more comp ng
the visual i the
various cell laye Hubel
and Wiesel have been e that
the *message about the* image falli the
retina undergoes a step-wise analy a
system of nerve cells stored in columns
system each cell has its specific function
responsible for a specific detail in the patter
the retinal image.

sensory, brain,
visual, perception,
retinal, cerebral cortex,
eye, cell, optical
nerve, image
Hubel, Wiesel



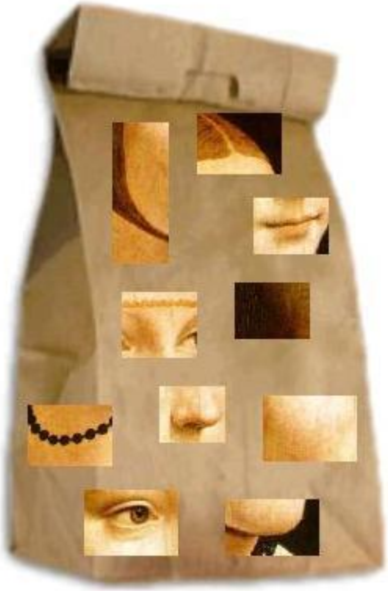
China is forecasting a trade surplus of $90bn
(£51bn) to $100bn this year, a threefold
increase on 2004's $32bn. The Commerce
Ministry said the surpl be created by a
predicted 30% $750bn,
compared 660bn.
The figur US,
which h are
unfairly d
yuan. t
says th a
governo o
needed t nd
so more go ina
increased the st the
dollar by 2.1% in trade
within a narrow band, but the U the
yuan to be allowed to trade freely. ver,
Beijing has made it clear that it will take e
and tread carefully before allowing the y
rise further in value.

China, trade,
surplus, commerce,
exports, imports, US,
yuan, bank, domestic,
foreign, increase,
trade, value

Object/Scene Recognition

- Analogy to documents: The content can be inferred from the frequency of visual words

object

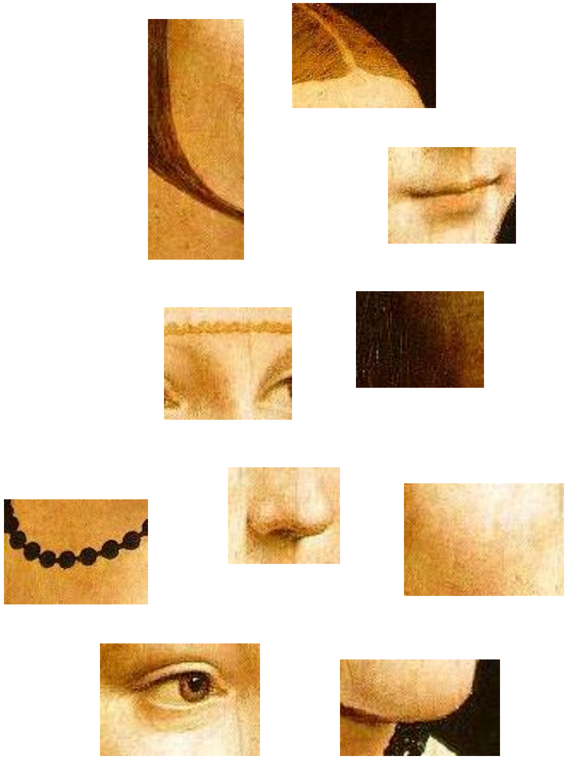


bag of visual words

Bag of Visual Words

- Visual words = (independent) features

face



features

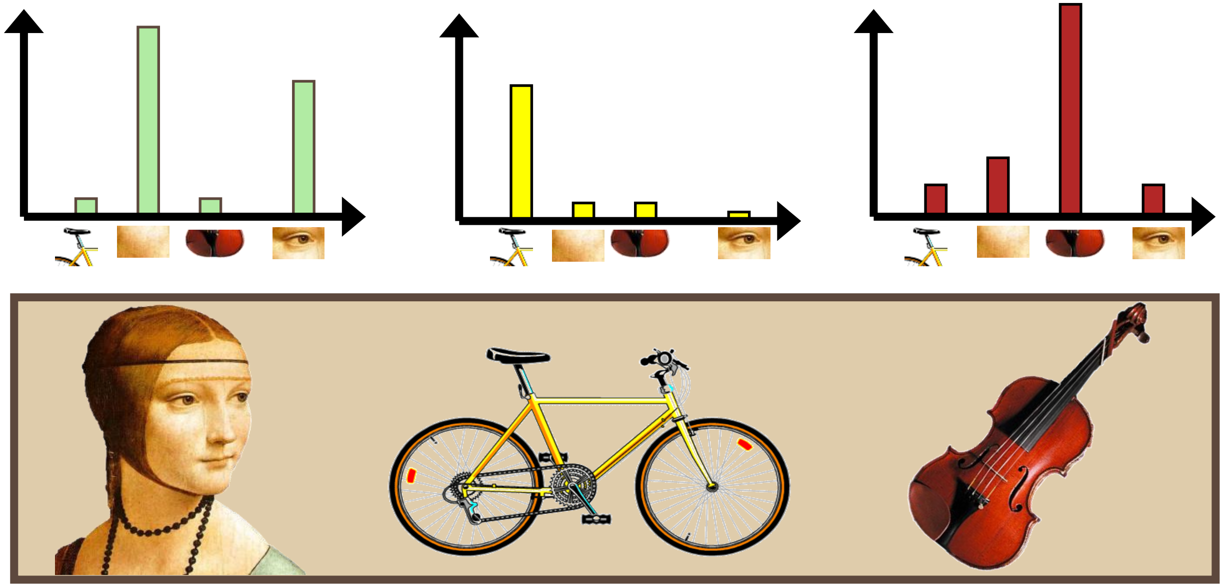
Bag of Visual Words

- Visual words = (independent) features
- Construct a dictionary of representative words

dictionary of visual words (codebook)

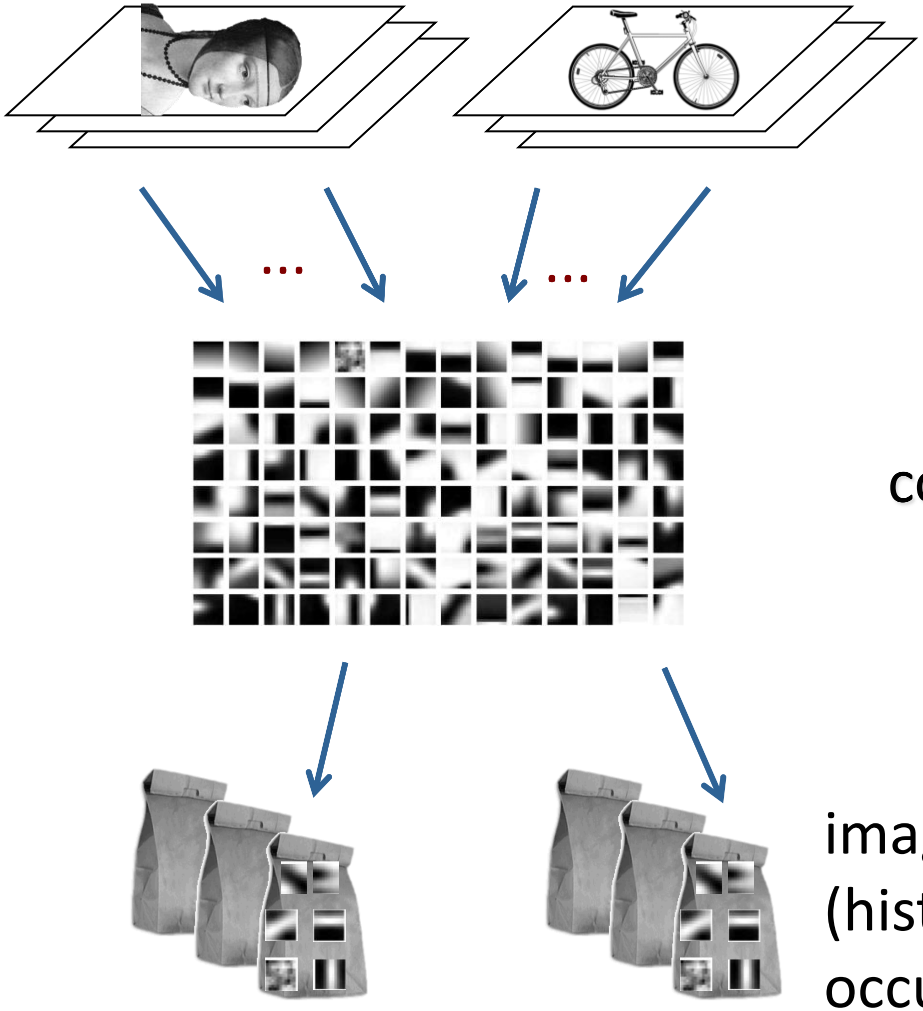
Bag of Visual Words

- Visual words = (independent) features
- Construct a dictionary of representative words
- Represent the image based on a histogram of word occurrences (bag)



Each detected feature is assigned to the closest entry in the codebook

Overview

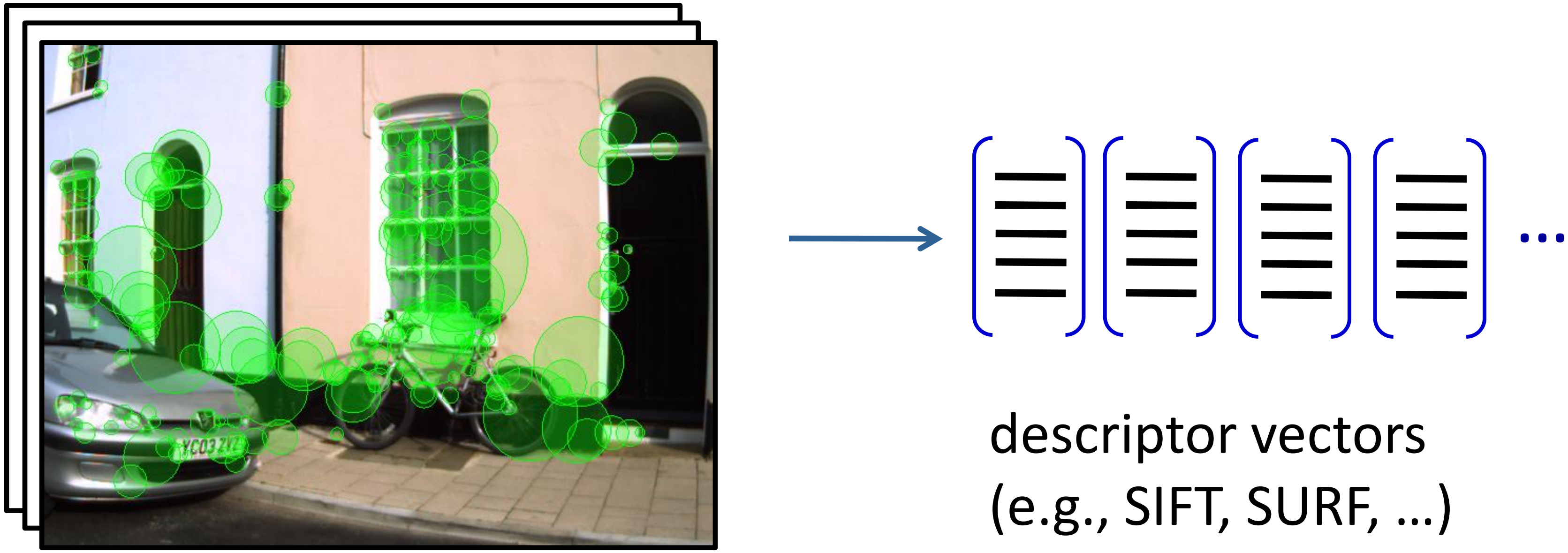


feature detection and extraction (e.g., SIFT, …)

codewords dictionary

image representation (histogram of word occurrences)

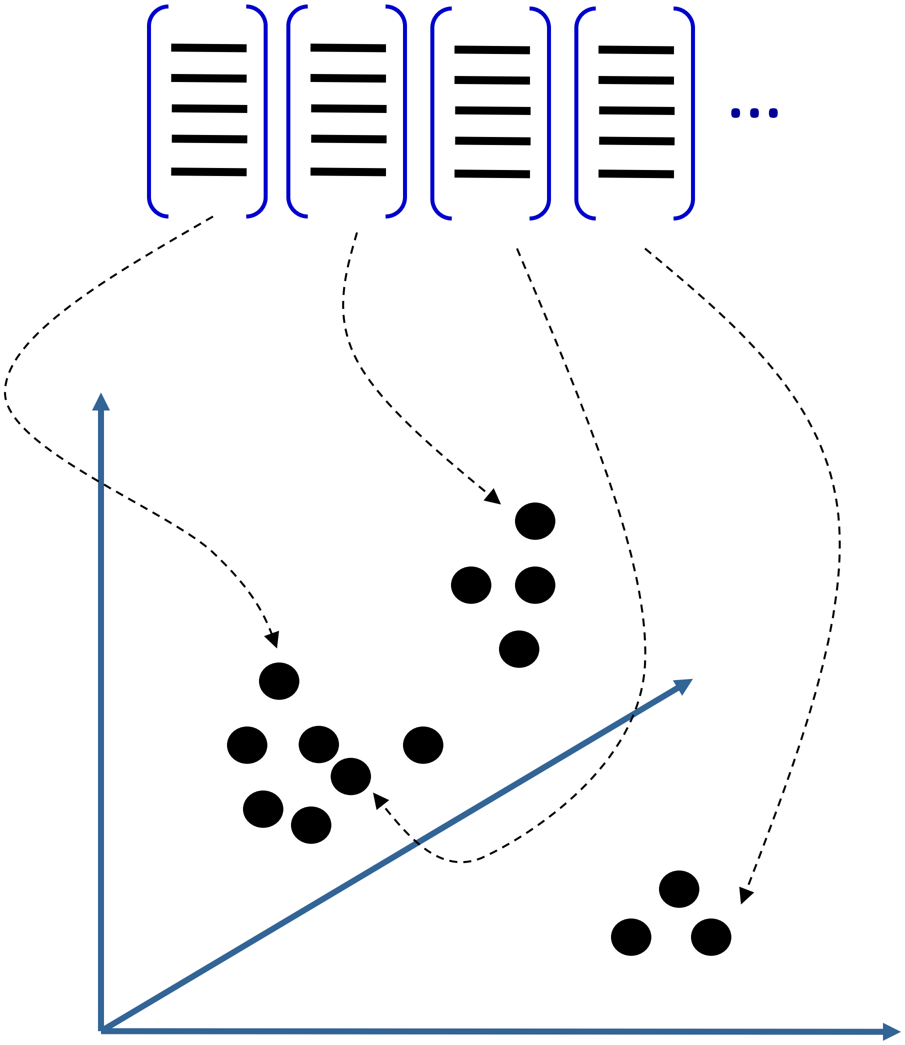
Learning the Dictionary



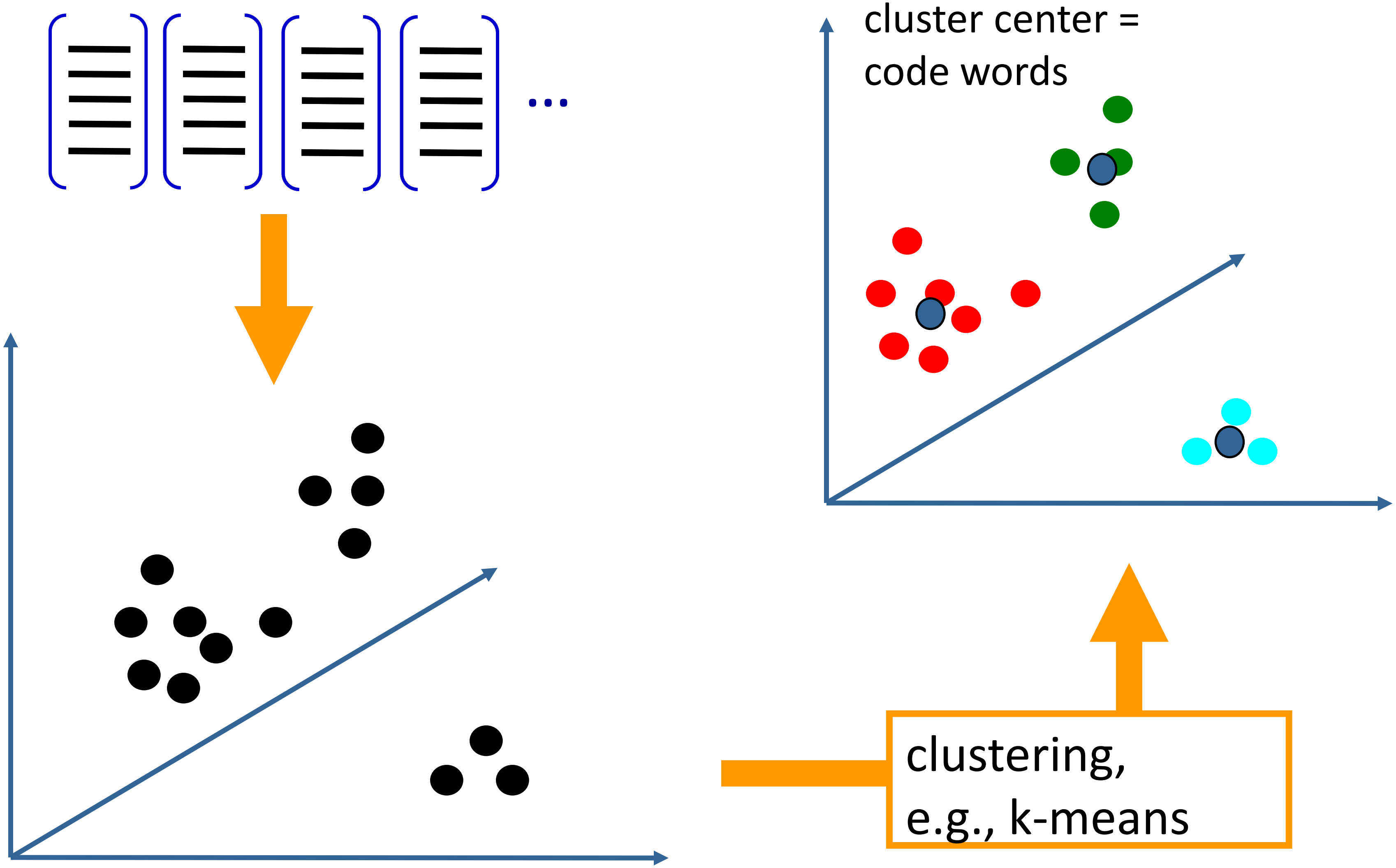
descriptor vectors
(e.g., SIFT, SURF, …)

example patch

Learning the Dictionary



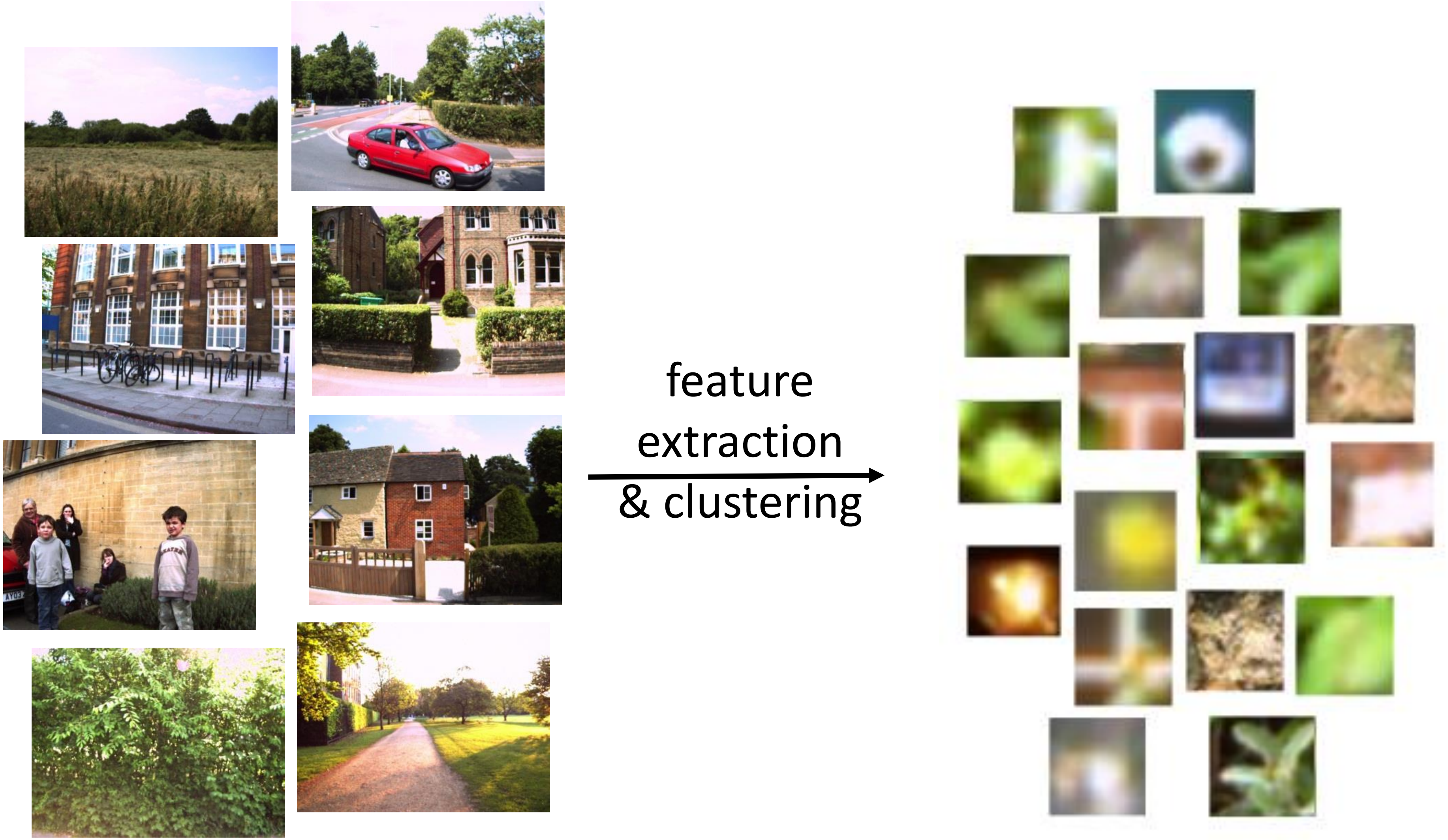
Learning the Dictionary



cluster center = code words

clustering, e.g., k-means

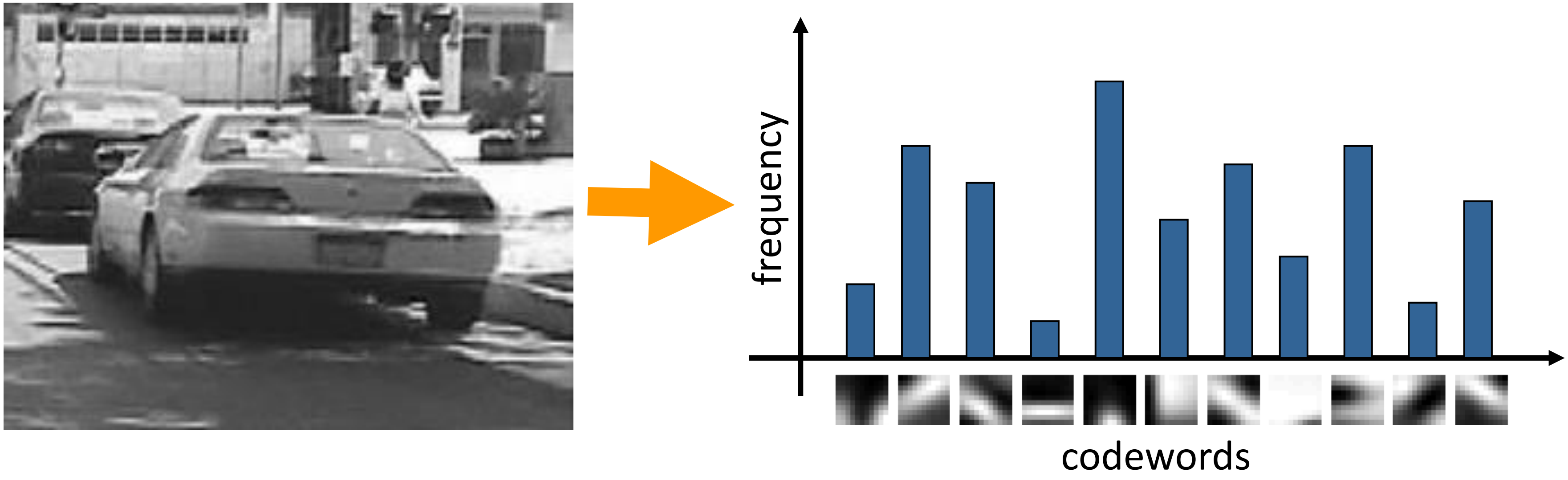
Learning the Visual Vocabulary



feature extraction & clustering

Example Image Representation

- Build the histogram by assigning each detected feature to the closest entry in the codebook

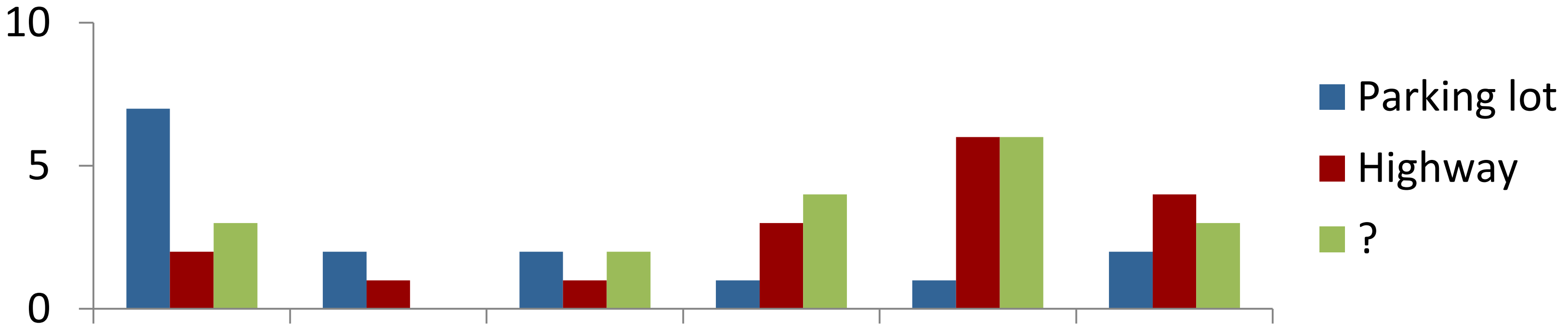


Object/Scene Recognition

- Compare histogram of new scene with those of known scenes, e.g., using
  - simple histogram intersection

$$score(\mathbf{p}, \mathbf{q}) = \sum \min(p_i, q_i)$$

  - naïve Bayes
  - more advanced statistical methods

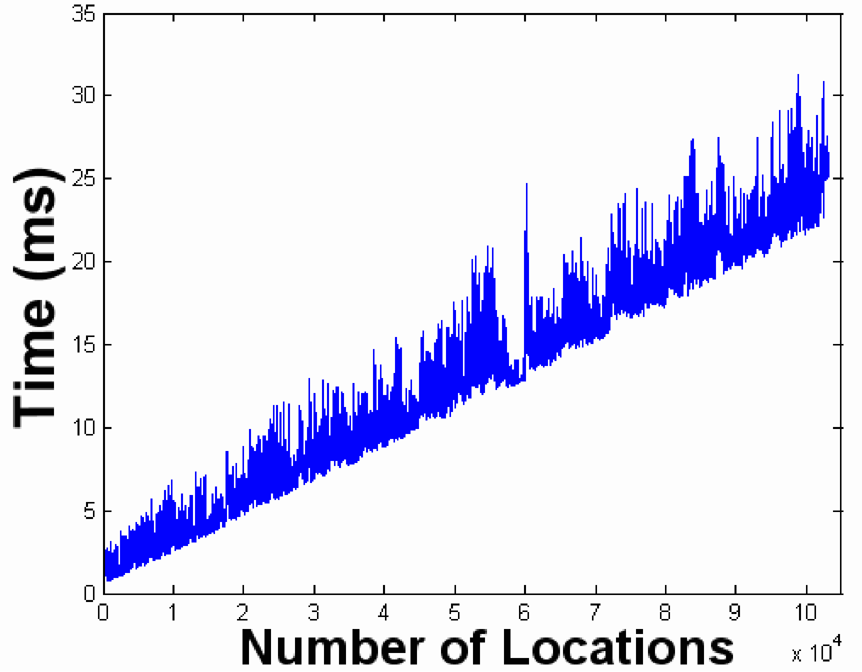


Example: FAB-MAP

[Cummins and Newman, 2008]

Timing Performance

- Inference: 25 ms for 100k locations
- SURF detection + quantization: 483 ms



Summary: Bag of Words

[Fei-Fei and Perona, 2005; Nister and Stewenius, 2006]

- Compact representation of content
- Highly efficient and scalable
- Requires training of a dictionary
- Insensitive to viewpoint changes/image deformations (inherited from feature descriptor)

Structure From Motion (SfM)

- Now we can retrieve relevant images and compute point correspondences between them
- What can we use them for?

Four Important SfM Problems

- Camera calibration / resection

  Known 3D points, observe corresponding 2D points, compute camera pose
- Point triangulation

  Known camera poses, observe 2D point correspondences, compute 3D point
- Motion estimation

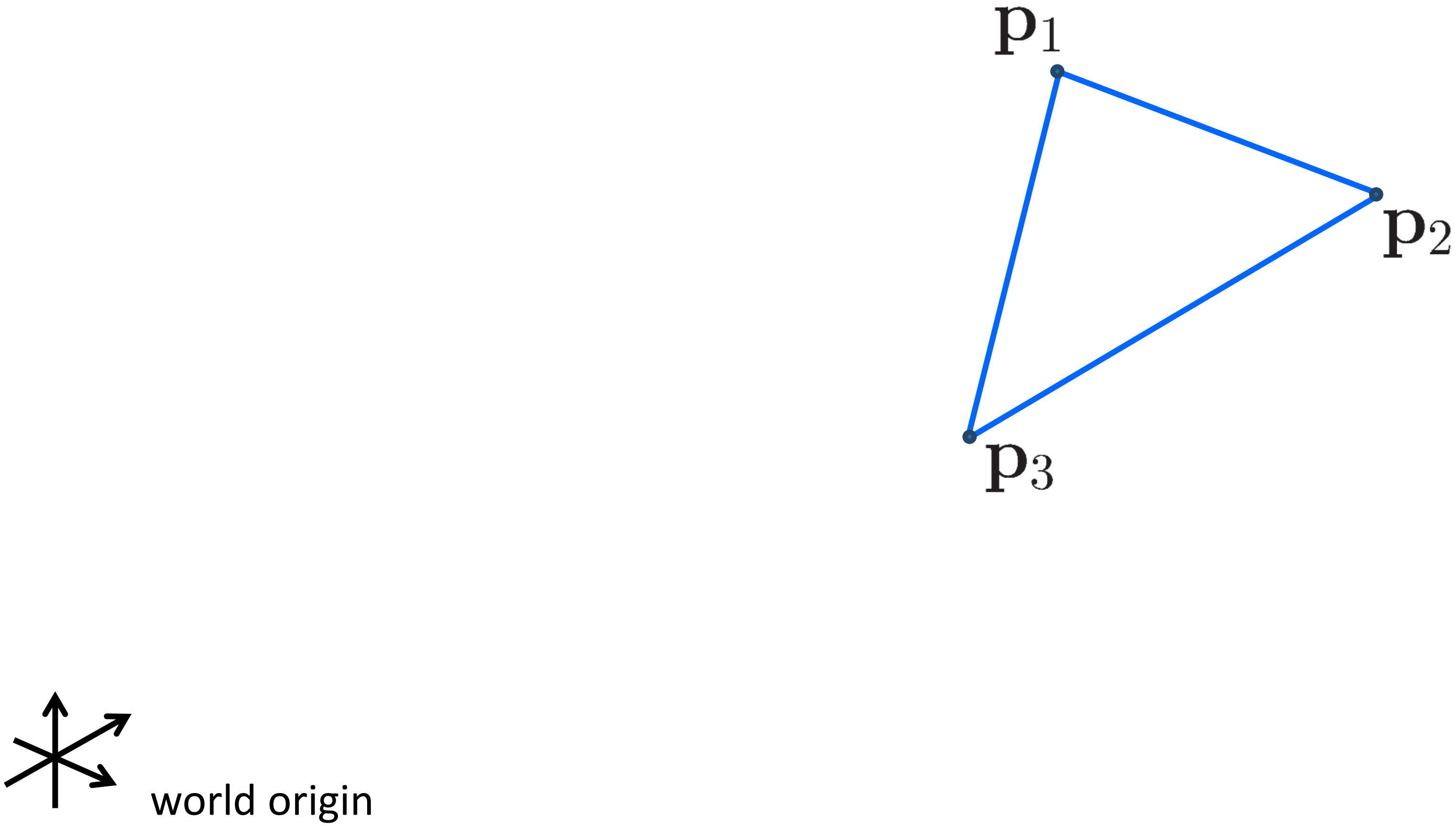
  Observe 2D point correspondences, compute camera pose (up to scale)
- Bundle adjustment / visual SLAM (next week!)

  Observe 2D point correspondences, compute camera pose and 3D points (up to scale)

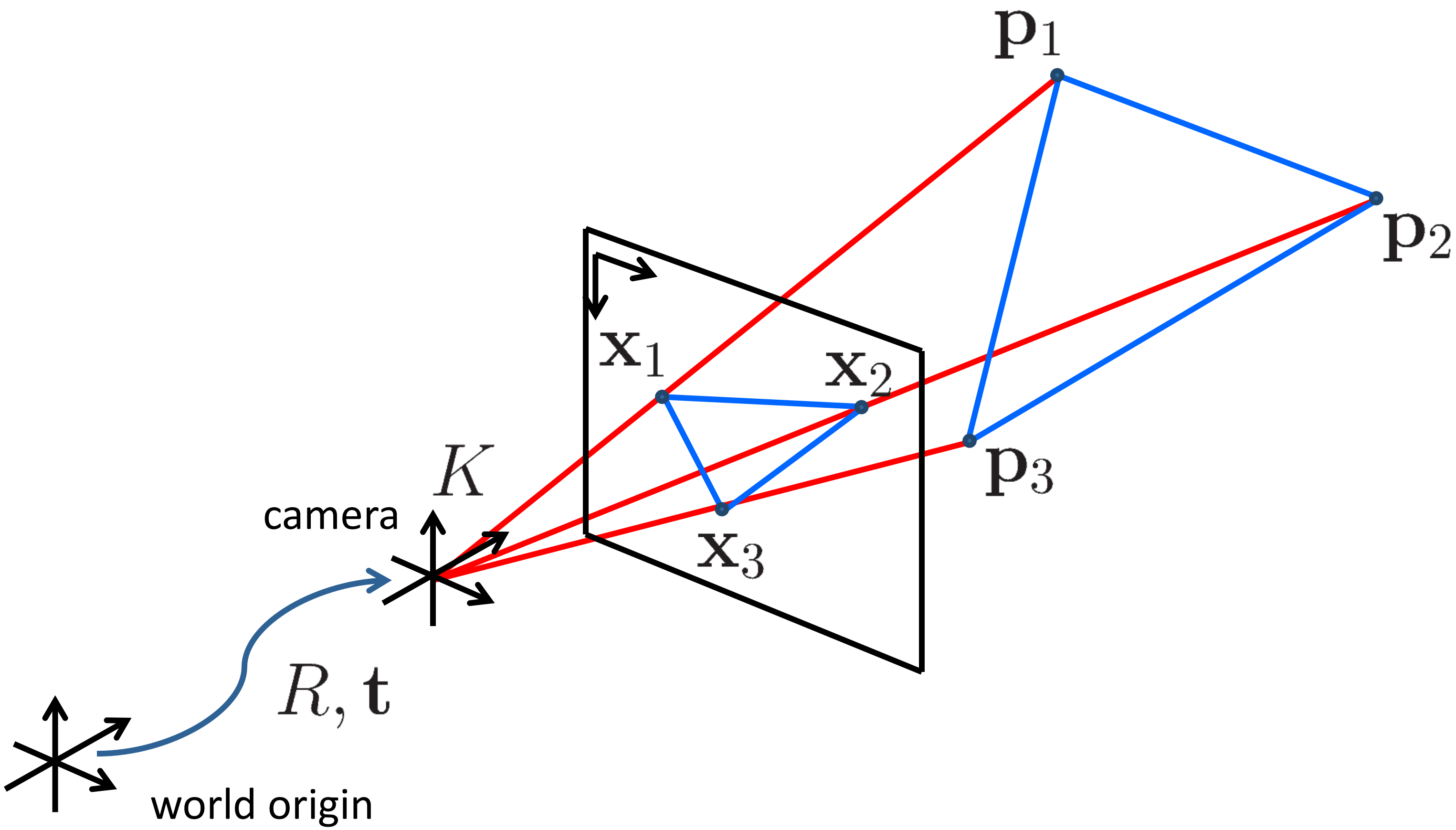
Four Important SfM Problems

- Each of these problems has many solution algorithms
- Approaches differ in:
  - Number of minimum points and assumptions of their configuration
  - Effect of noise (bias)
  - Conditioning
  - Simplicity vs. accuracy (linear vs. non-linear)

Camera Calibration (Perspective n-Point Problem)



Camera Calibration (Perspective n-Point Problem)



Camera Calibration

- **Given:** n 2D/3D correspondences $\mathbf{x}_i \leftrightarrow \mathbf{p}_i$
- **Wanted:** $M = K(R \;\; \mathbf{t})$
  such that $\tilde{\mathbf{x}}_i = M\mathbf{p}_i$

- Question: How many DOFs does M have?
- The algorithm has two parts:
  1. Compute $M \in \mathbb{R}^{3\times 4}$
  2. Decompose M into $K, R, \mathbf{t}$ via QR decomposition

Step 1: Estimate M

- $\tilde{\mathbf{x}}_i = M\mathbf{p}_i$
- Each correspondence generates two equations

$$x = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}W}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}W} \qquad y = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}W}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}W}$$

- Multiplying out gives equations **linear** in the elements of M

$$(m_{31}X + m_{32}Y + m_{33}Z + m_{34}W)x = m_{11}X + m_{12}Y + m_{13}Z + m_{14}W$$

$$(m_{31}X + m_{32}Y + m_{33}Z + m_{34}W)y_j = m_{21}X + m_{22}Y + m_{23}Z + m_{24}W$$

- Re-arrange in matrix form →

Step 1: Estimate M

- Re-arranged in matrix form

$$\begin{pmatrix} X & Y & Z & 1 & 0 & 0 & 0 & 0 & -xX & -xY & -xZ & -x \\ 0 & 0 & 0 & 0 & X & Y & Z & 1 & -yX & -yY & -yZ & -y \end{pmatrix} \mathbf{m} = \mathbf{0}$$

  with $\mathbf{m} = \begin{pmatrix} m_{11} & m_{12} & \dots & m_{34} \end{pmatrix} \in \mathbb{R}^{12}$

- Concatenate equations for n≥6 correspondences

$$A\mathbf{m} = \mathbf{0}$$

- Wanted vector $\mathbf{m}$ is in the null space of A
- Initial solution using SVD (vector with least singular value), refine using non-linear min.

Step 2: Recover K,R,t

- Remember $M = K(R \;\; \mathbf{t})$
- The first 3x3 submatrix is the product of an upper triangular and orthogonal (rot.) matrix

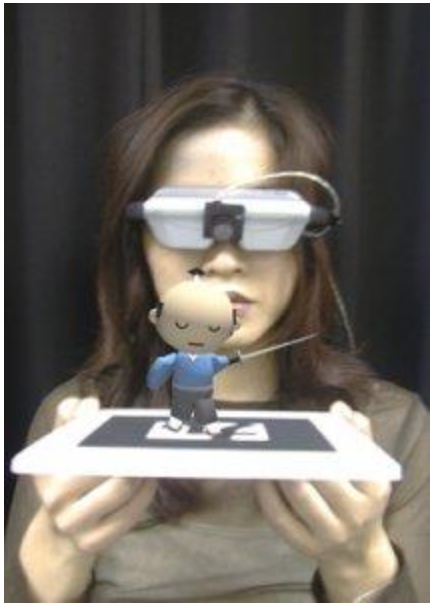
$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

Procedure:

1. Factor M into KR using QR decomposition
2. Compute translation as $\mathbf{t} = K^{-1}(p_{14}, p_{24}, p_{34})^\top$

Example: ARToolkit Markers (1999)

1. Threshold image
2. Detect edges and fit lines
3. Intersect lines to obtain corners
4. Estimate projection matrix M
5. Extract camera pose R,t (assume K is known)

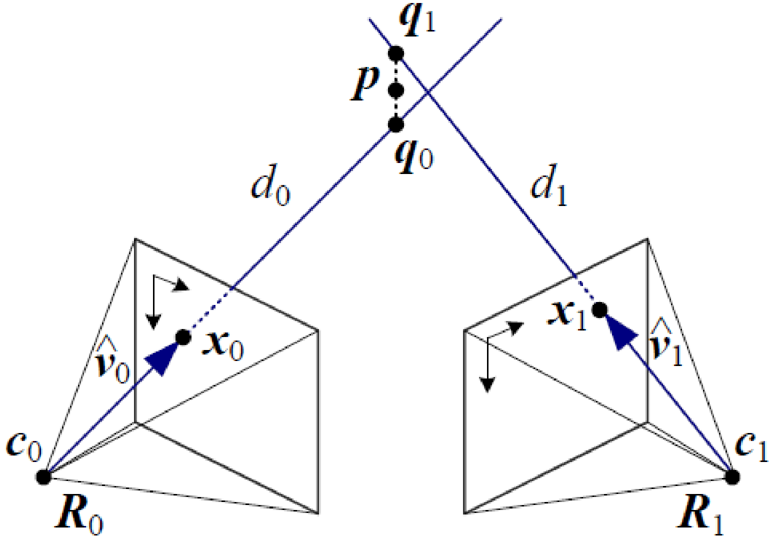


The final error between measured and projected points is typically less than 0.02 pixels

Triangulation

- **Given:** n cameras $\{M_j = K_j(R_j \ \mathbf{t}_j)\}$

  Point correspondence $\mathbf{x}_0, \mathbf{x}_1$
- **Wanted:** Corresponding 3D point $\mathbf{p}$



Triangulation

- Where do we expect to see $\mathbf{p} = (X\ Y\ Z\ W)^\top$?

$$\hat{x} = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}W}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}W} \qquad \hat{y} = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}W}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}W}$$

- Minimize the residuals

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \sum_j d(\mathbf{x}_j, \hat{\mathbf{x}}_j)^2$$

Triangulation

- Multiply with denominator gives

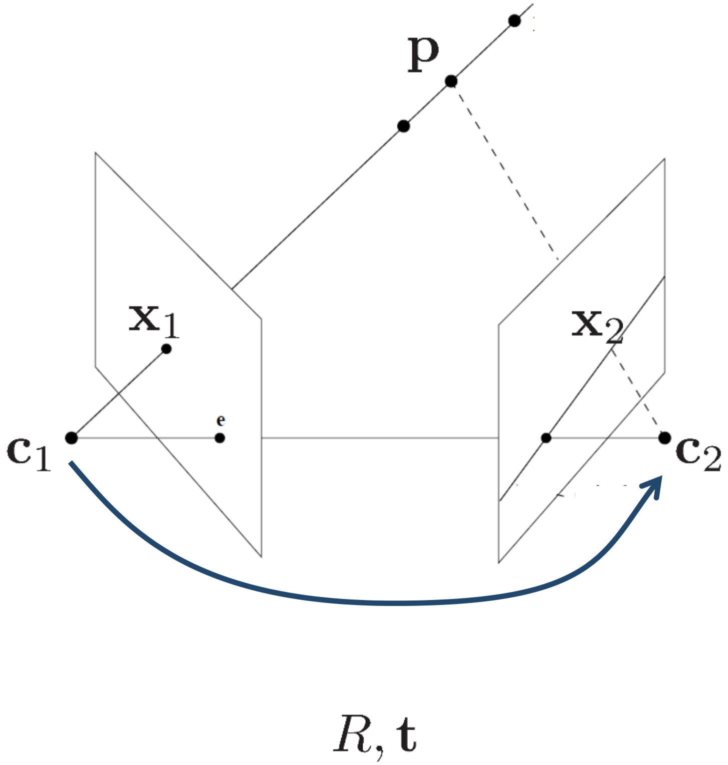
$$0 = (x_j m_{31} - m_{11})X + (x_j m_{32} - m_{12})Y + (x_j m_{33} - m_{13})Z + (x_j m_{34} - m_{14})W$$
$$0 = (y_j m_{31} - m_{21})X + (y_j m_{32} - m_{22})Y + (y_j m_{33} - m_{23})Z + (y_j m_{34} - m_{24})W$$

Solve for $\mathbf{p} = (X \ Y \ Z \ W)^\top$ using:

- Linear least squares with W=1
- Linear least squares using SVD
- Non-linear least squares of the residuals (most accurate)

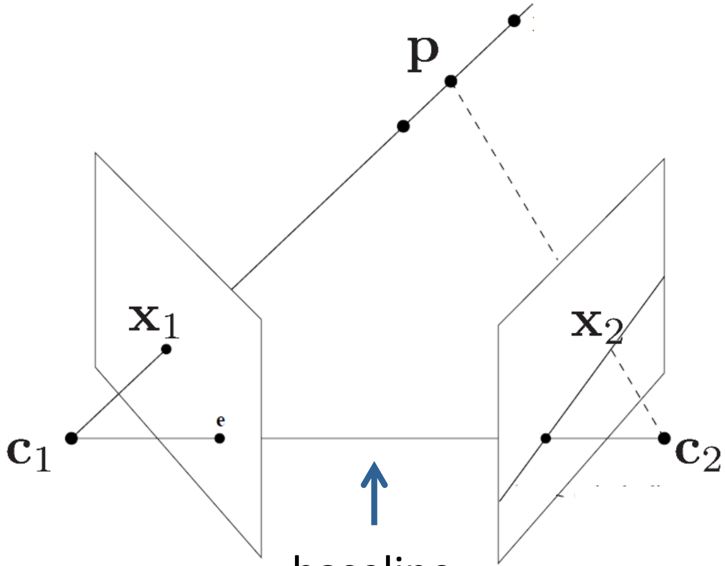
Epipolar Geometry

- Let’s consider two cameras that observe a 3D world point



Epipolar Geometry

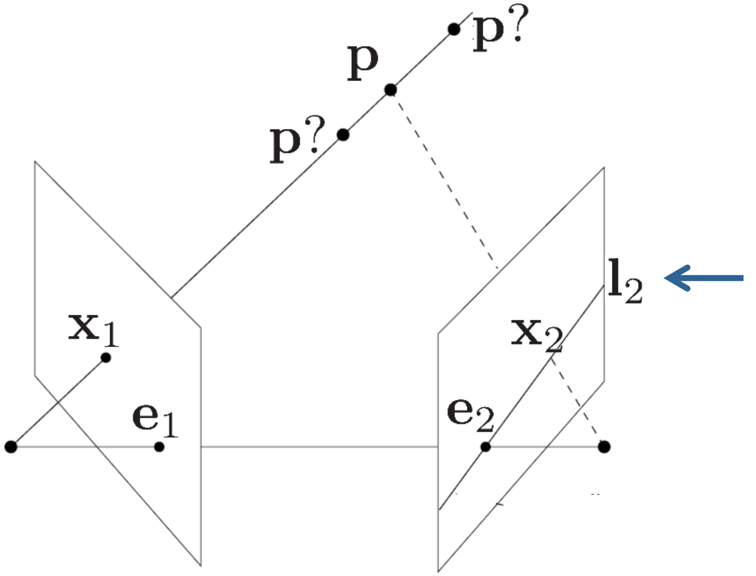
- The line connecting both camera centers is called the **baseline**



baseline
(line joining both camera centers)

Epipolar Geometry

- Given the image of a point in one view, what can we say about its position in another?



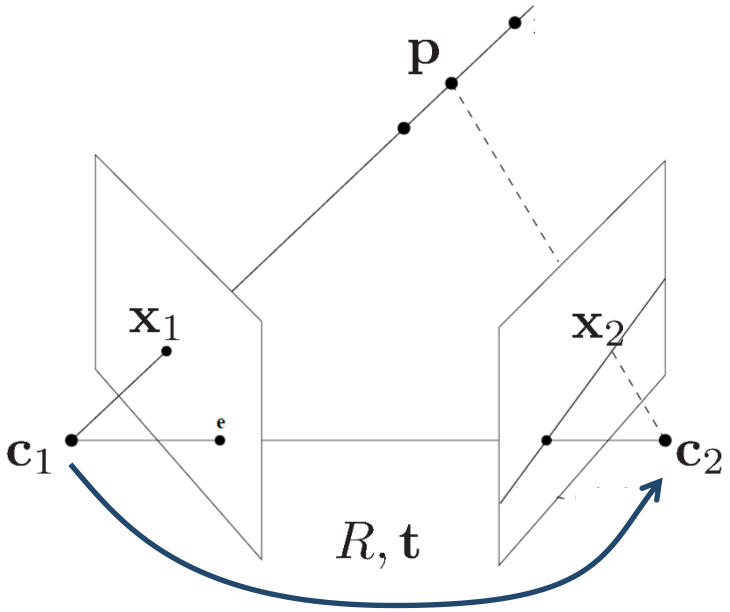
epipolar line of x

- A point in one image "generates" a line in another image (called the **epipolar line**)

Epipolar Geometry

- Left line in left camera frame $\mathbf{p}_1 = d_1 \hat{\mathbf{x}}_1$
- Right line in right camera frame $\mathbf{p}_2 = d_2 \hat{\mathbf{x}}_2$

where $\hat{\mathbf{x}}_\mathbf{j} = K^{-1} \bar{\mathbf{x}}_j$ are the (local) ray directions



Epipolar Geometry

- Left line in **right** camera frame $\mathbf{p}_1' = R d_1 \hat{\mathbf{x}}_1 + t$
- Right line in right camera frame $\mathbf{p}_2 = d_2 \hat{\mathbf{x}}_2$

where $\hat{\mathbf{x}}_\mathbf{j} = K^{-1}\bar{\mathbf{x}}_j$ are the (local) ray directions

- Intersection of both lines

$$d_2\hat{\mathbf{x}}_2 = R d_1 \hat{\mathbf{x}}_1 + \mathbf{t} \qquad \Big| [\mathbf{t}]_\times \cdot$$

$$d_2[\mathbf{t}]_\times \hat{\mathbf{x}}_2 = d_1 [\mathbf{t}]_\times R \hat{\mathbf{x}}_1 + \cancel{[\mathbf{t}]_\times \mathbf{t}} \;\; =0 \qquad \Big| \hat{\mathbf{x}}_2^\top \cdot$$

$$0= \;\; \cancel{d_2 \hat{\mathbf{x}}_2^\top [\mathbf{t}]_\times \hat{\mathbf{x}}_2} = d_1 \hat{\mathbf{x}}_2^\top [\mathbf{t}]_\times R \hat{\mathbf{x}}_1$$

$$0 = \hat{\mathbf{x}}_2^\top [\mathbf{t}]_\times R \hat{\mathbf{x}}_1$$

$$\boxed{0 = \hat{\mathbf{x}}_2^\top E \hat{\mathbf{x}}_1}$$

this is called the epipolar constraint

Epipolar Geometry

Note: The epipolar constraint holds for **every** pair of corresponding points $\mathbf{x}_1, \mathbf{x}_2$

$$\hat{\mathbf{x}}_2^\top E \hat{\mathbf{x}}_1 = 0$$

where E is called the essential matrix

$$E = [\mathbf{t}]_\times R \in \mathbb{R}^{3\times 3}$$

3D Motion Estimation

- **Given:** 2 camera images
  n point correspondences
- **Wanted:** Camera motion R,t (up to scale)

- Solutions:
  - 8-point algorithm
  - normalized 8-point algorithm
  - 6-point algorithm
  - 5-point algorithm

8-Point Algorithm: General Idea

1. Estimate the essential matrix E from at least eight point correspondences
2. Recover the relative pose R,t from E (up to scale)

Step 1: Estimate E

- Epipolar constraint $\hat{\mathbf{x}}_2^\top E \hat{\mathbf{x}}_1 = 0$
- Written out (with $\mathbf{x}_j = (x_j, y_j, 1)^\top$)

$$\begin{array}{ccccccc} x_1 x_2 e_{11} & + & y_1 x_2 e_{12} & + & x_2 e_{13} & + & \\ x_1 y_2 e_{21} & + & y_1 y_2 e_{22} & + & y_2 e_{23} & + & \\ x_1 e_{31} & + & y_1 e_{32} & + & 1 e_{33} & = & 0 \end{array}$$

- Stack the elements into two vectors

$$\left.\begin{aligned} \mathbf{z} &= (x_1 x_2 \;\; y_1 x_2 \;\; \dots \;\; 1)^\top \\ \mathbf{e} &= (e_{11} \;\; e_{12} \;\; \dots \;\; e_{33})^\top \end{aligned}\right\} \; \mathbf{z}^\top \mathbf{e} = 0$$

Step 1: Estimate E

- Each correspondence gives us one constraint

$$\left.\begin{aligned} \mathbf{z}_1^\top \mathbf{e} &= 0 \\ \mathbf{z}_2^\top \mathbf{e} &= 0 \\ &\vdots \\ \mathbf{z}_n^\top \mathbf{e} &= 0 \end{aligned}\right\} \quad Z\mathbf{e} = \mathbf{0}$$

- Linear system with *n* equations
- e is in the null-space of Z
- Solve using SVD (assuming $\|\mathbf{e}\| = 1$)

Normalized 8-Point Algorithm

[Hartley 1997]

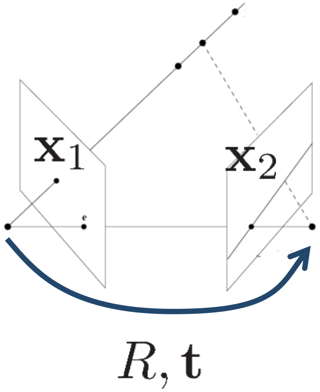
- Noise in the point observations is unequally distributed in the constraints, e.g.,

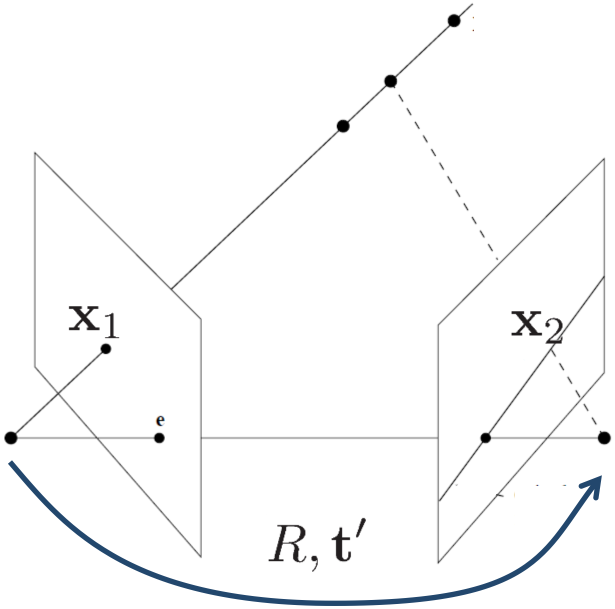
$$\begin{array}{lccccccccc}
\text{double noise} & \boxed{x_1x_2}e_{11} & + & y_1x_2e_{12} & + & x_2e_{13} & + & & \\
 & x_1y_2e_{21} & + & y_1y_2e_{22} & + & y_2e_{23} & + & & \\
\text{normal noise} & \boxed{x_1}e_{31} & + & y_1e_{32} & + & \underset{\text{noise free}}{\boxed{1}e_{33}} & = & 0 &
\end{array}$$

- Estimation is sensitive to scaling
- Normalize all points to have zero mean and unit variance

Step 2: Recover R,t

- **Note:** The absolute distance between the two cameras can never be recovered from pure images measurements alone!!!
- Illustration





- We can only recover the translation $\hat{\mathbf{t}}$ up to scale

Step 2a: Recover t

- Remember: $E = [\mathbf{t}]_\times R$
- Therefore, $\mathbf{t}^\top$ is in the null space of E

$$\mathbf{t}^\top E = \underbrace{\mathbf{t}^\top [\mathbf{t}]_\times}_{=0} R = 0$$

→ Recover $\hat{\mathbf{t}}$ (up to scale) using SVD

$$E = [\hat{\mathbf{t}}]_\times R = U \Sigma V^\top$$

$$= \begin{pmatrix} \mathbf{u}_0 & \mathbf{u}_1 & \boxed{\hat{\mathbf{t}}} \end{pmatrix} \begin{pmatrix} 1 & & \\ & 1 & \\ & & \boxed{0} \end{pmatrix} \begin{pmatrix} \mathbf{v}_0^\top & \mathbf{v}_1^\top & \mathbf{v}_2^\top \end{pmatrix}$$

Step 2b: Recover R

Remember, the cross-product $[\hat{\mathbf{t}}]_\times$

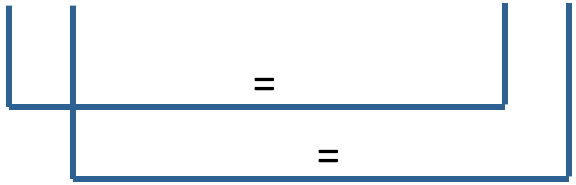
- … projects a vector onto a set of orthogonal basis vectors including $\hat{\mathbf{t}}$
- … zeros out the $\hat{\mathbf{t}}$ component
- … rotates the other two by 90°

$$
\begin{aligned}
[\hat{\mathbf{t}}]_\times &= SZR_{90^\circ}S^\top \\
&= (\mathbf{s}_0\ \mathbf{s}_1\ \hat{\mathbf{t}}) \begin{pmatrix} 1 & & \\ & 1 & \\ & & 0 \end{pmatrix} \begin{pmatrix} 0 & -1 & \\ 1 & 0 & \\ & & 1 \end{pmatrix} \begin{pmatrix} \mathbf{s}_0^\top \\ \mathbf{s}_1^\top \\ \hat{\mathbf{t}}^\top \end{pmatrix}
\end{aligned}
$$

Step 2b: Recover R

- Plug this into the essential matrix equation

$$E = [\mathbf{t}]_\times R = S Z R_{90^\circ} S^\top R = U \Sigma V^\top$$



- By identifying $S = U$ and $Z = \Sigma$, we obtain

$$R_{90^\circ} U^\top R = V^\top$$

$$\boxed{R = U R_{90^\circ}^\top V^\top}$$

Step 2b: Recover R

- Matrices U,V are not guaranteed to be rotations (sign flip still yields a valid SVD)

$$R = \pm U R_{\pm 90^\circ}^\top V^\top$$

- Identify the correct solution using
  - Select those two solutions with $\det R = 1$
  - Triangulate points in 3D
  - Select the solution with the largest number of points **in front** of the camera

Summary: 8-Point Algorithm

Given: Image pair

Find: Camera motion R,t (up to scale)

- Compute correspondences
- Compute essential matrix
- Extract camera motion

Lessons Learned Today

- … how to detect and match feature points
- … how to efficiently recognize places
- … how to estimate the camera pose and to triangulate points