4.1 Feature Detection & Matching: Points & Patches

Outline

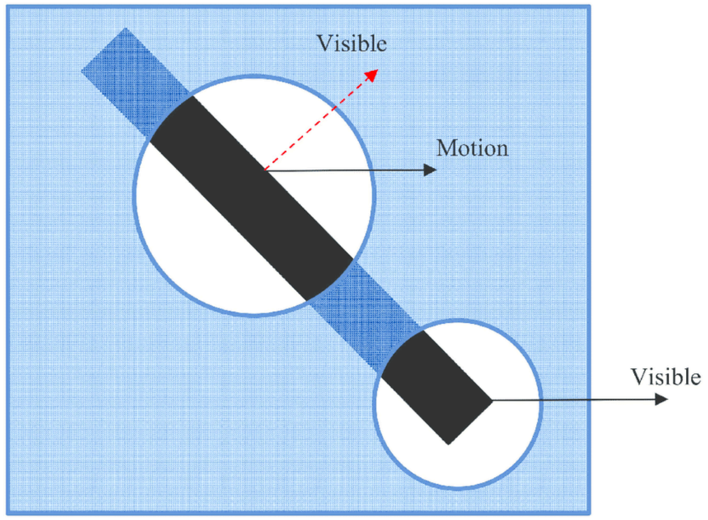
- Feature detectors
- Feature descriptors
- Feature matching
- Feature tracking

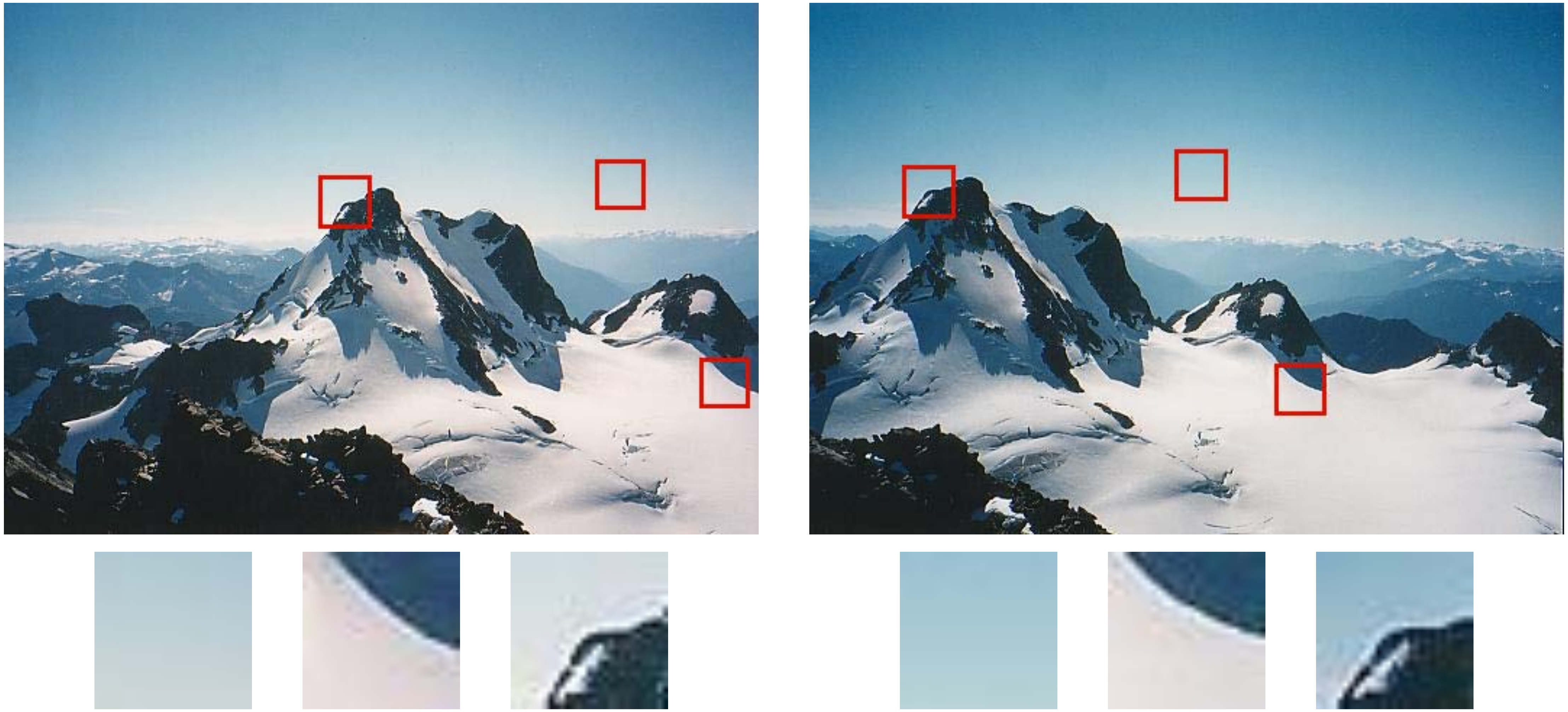
Outline

- **Feature detectors**
- Feature descriptors
- Feature matching
- Feature tracking

What Makes a Good Feature?

- Constant colour
  - Bad - many false matches
- Straight lines or smooth curves
  - Better - but still suffer from the ‘aperture problem’
- Sharp corners
  - Great - often unique!





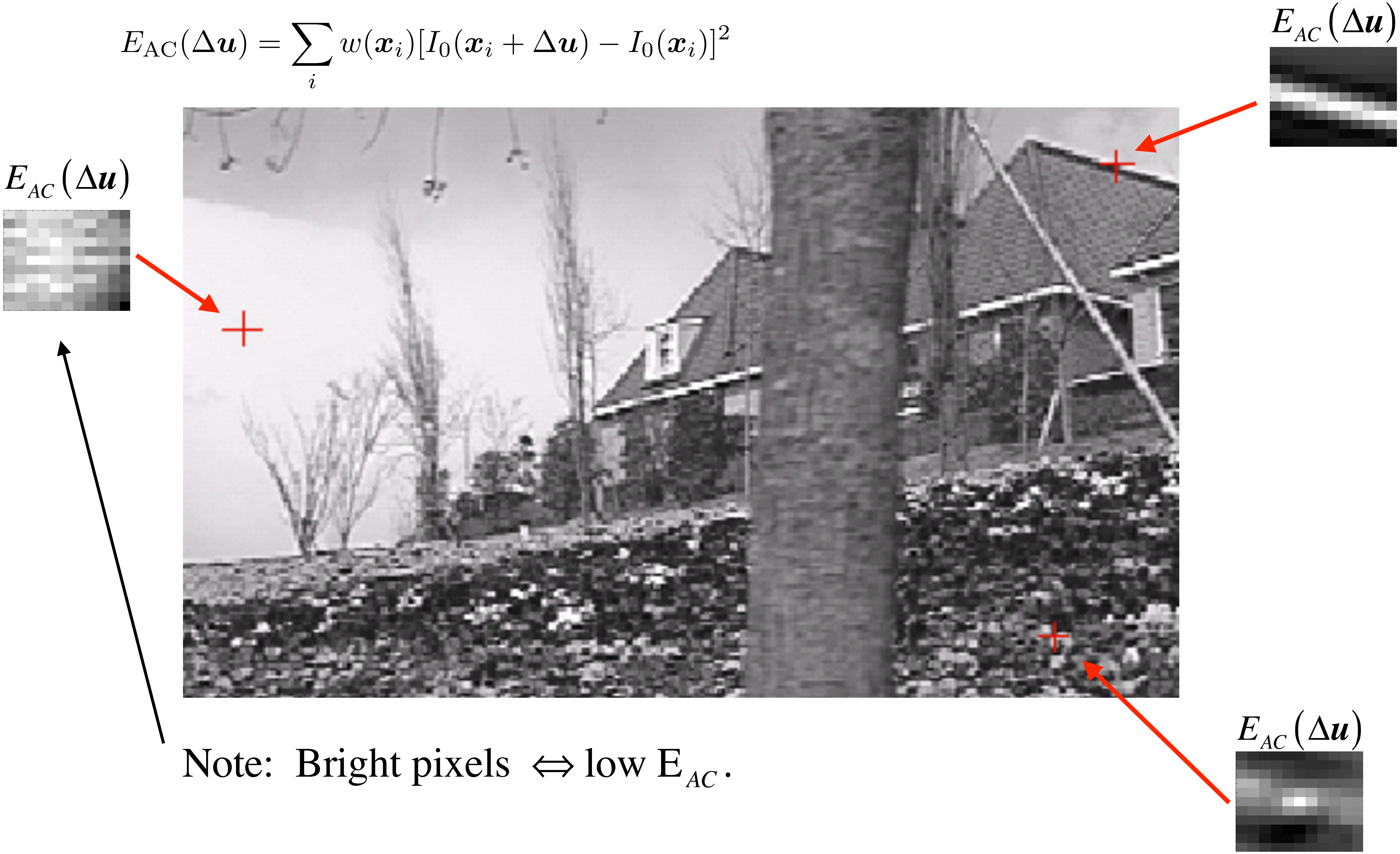
The Barber Pole Illusion

By Sakurambo - Own work (animated 3D model),
CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=798589

Feature Stability

- Local stability of feature can be assessed by computing a local weighted squared deviation of the image patch at the feature location from neighbouring patches:

$$E_{\mathrm{AC}}(\Delta \boldsymbol{u}) = \sum_i w(\boldsymbol{x}_i)[I_0(\boldsymbol{x}_i + \Delta \boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2$$



Note: Bright pixels ⇔ low E_{AC}.

End of Lecture
Oct 17, 2018

Gradient-Based Features

- Taylor series approximation of the local deviation:

$$\begin{aligned} E_{\mathrm{AC}}(\Delta \boldsymbol{u}) &= \sum_i w(\boldsymbol{x}_i)[I_0(\boldsymbol{x}_i + \Delta \boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2 \\ &\approx \sum_i w(\boldsymbol{x}_i)[I_0(\boldsymbol{x}_i) + \nabla I_0(\boldsymbol{x}_i) \cdot \Delta \boldsymbol{u} - I_0(\boldsymbol{x}_i)]^2 \\ &= \sum_i w(\boldsymbol{x}_i)[\nabla I_0(\boldsymbol{x}_i) \cdot \Delta \boldsymbol{u}]^2 \\ &= \Delta \boldsymbol{u}^T \boldsymbol{A} \Delta \boldsymbol{u}, \end{aligned}$$

where

$$\nabla I_0(\boldsymbol{x}_i) = (\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y})(\boldsymbol{x}_i)$$

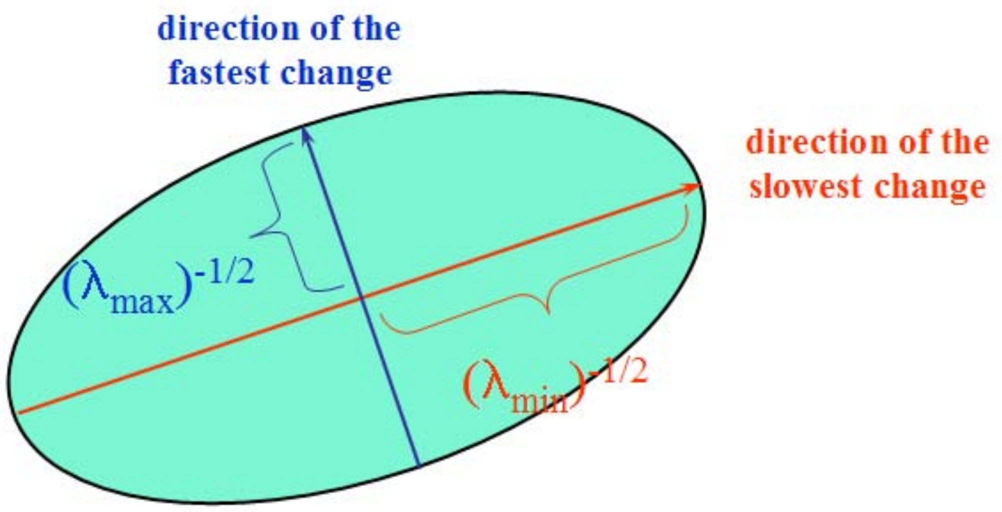
and

$$\boldsymbol{A} = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Eigenvalue Analysis

$$\boldsymbol{A} = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \longrightarrow \text{Eigenvalues } \lambda_{\text{min}}, \lambda_{\text{max}}$$

- This is the Hessian matrix of $I(x, y)$.
- It provides a quadratic approximation to the local shape of the deviation.
- The deviation changes most gradually in the direction of the smallest eigenvector.
- Thus when selecting features we should try to maximize the smallest eigenvalue.



Scalar Interest Measures

- A number of scalar interest measures based upon the eigenvalues of the Hessian have been proposed:

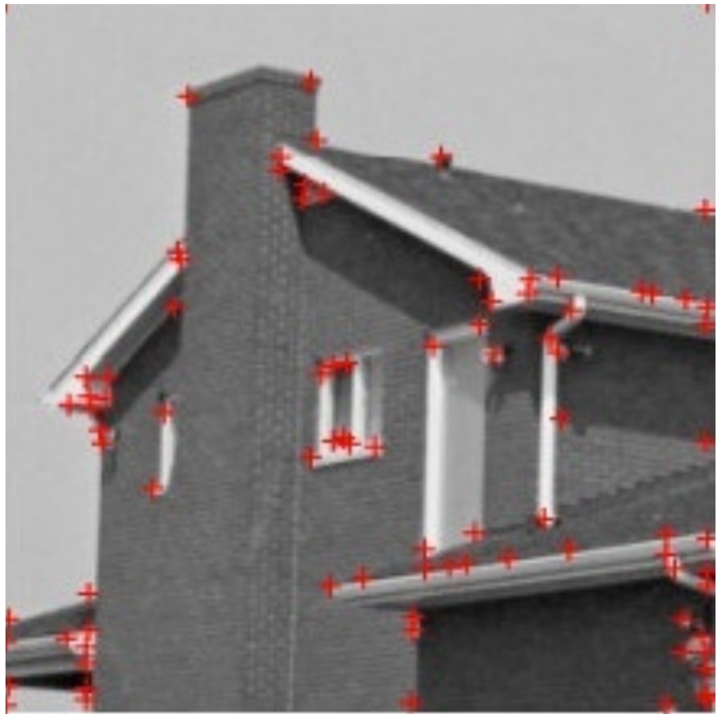
For $\lambda_0 < \lambda_1$:

λ_0	(Shi & Tomasi 1994)
$\lambda_0 \lambda_1 - \alpha \left(\lambda_0 + \lambda_1 \right)^2$	(Harris & Stephens 1988)
$\lambda_0 - \alpha \lambda_1$	(Triggs 2004)
$\dfrac{\lambda_0 \lambda_1}{\lambda_0 + \lambda_1}$	(Brown, Szeliski & Winder 2005)

Outline of Basic Feature Detection Algorithm

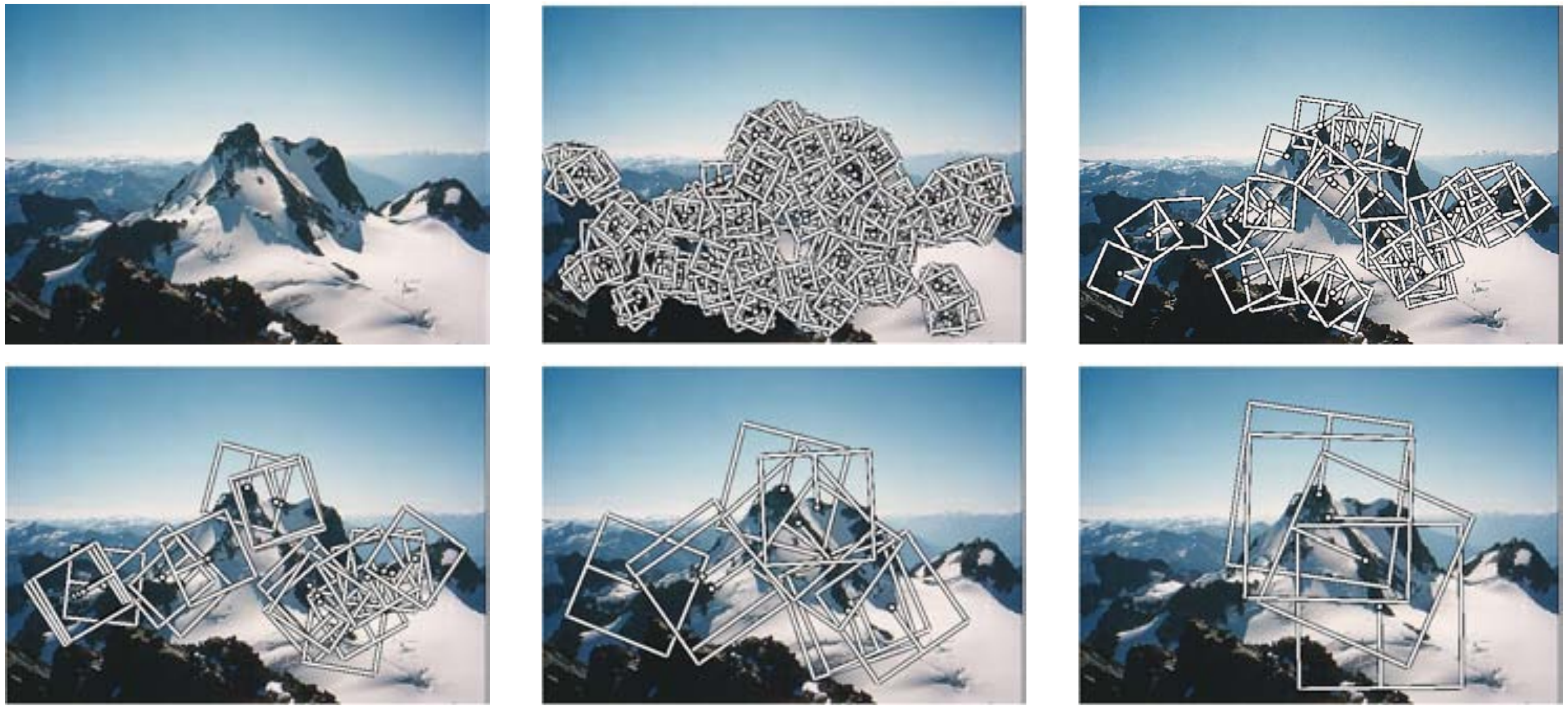
1. Compute the horizontal and vertical derivatives of the image I_x and I_y by convolving the original image with derivatives of Gaussians (Section 3.2.3).
2. Compute the three images corresponding to the outer products of these gradients. (The matrix $\boldsymbol{A}$ is symmetric, so only three entries are needed.)
3. Convolve each of these images with a larger Gaussian.
4. Compute a scalar interest measure using one of the formulas discussed above.
5. Find local maxima above a certain threshold and report them as detected feature point locations.

Output of Harris Detector



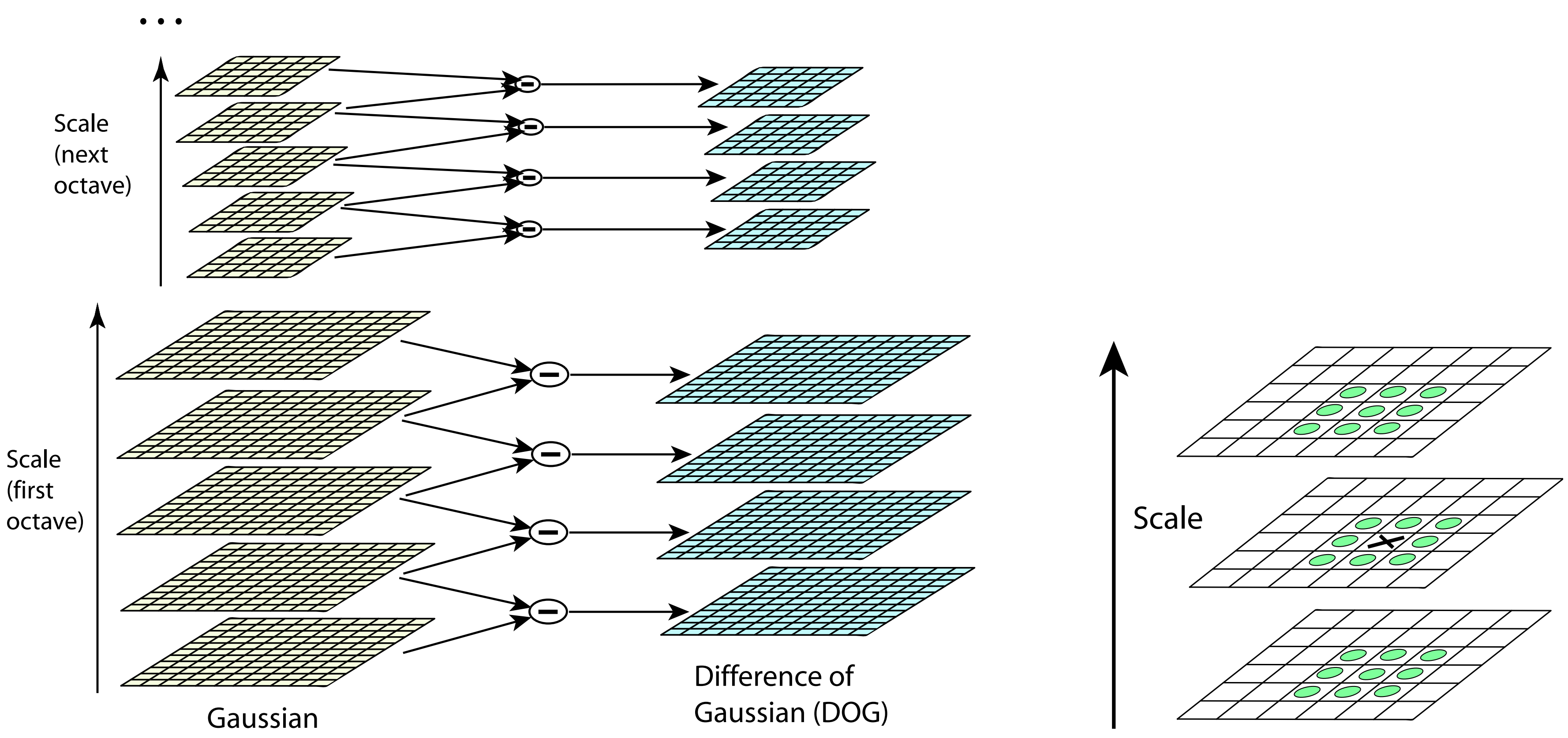
Multi-Scale Methods

- Features can exist at any scale
- Only using the finest-scale may not make sense (e.g., for images with no fine-scale structure)
- One option is to run the feature detector at many scales, in a pyramid design.
- Matching and tracking can then be done within each scale.
- This makes sense when the scale of a feature is not expected to change between frames
  - Aerial imagery
  - Panorama stitching



Scale-Invariant Methods

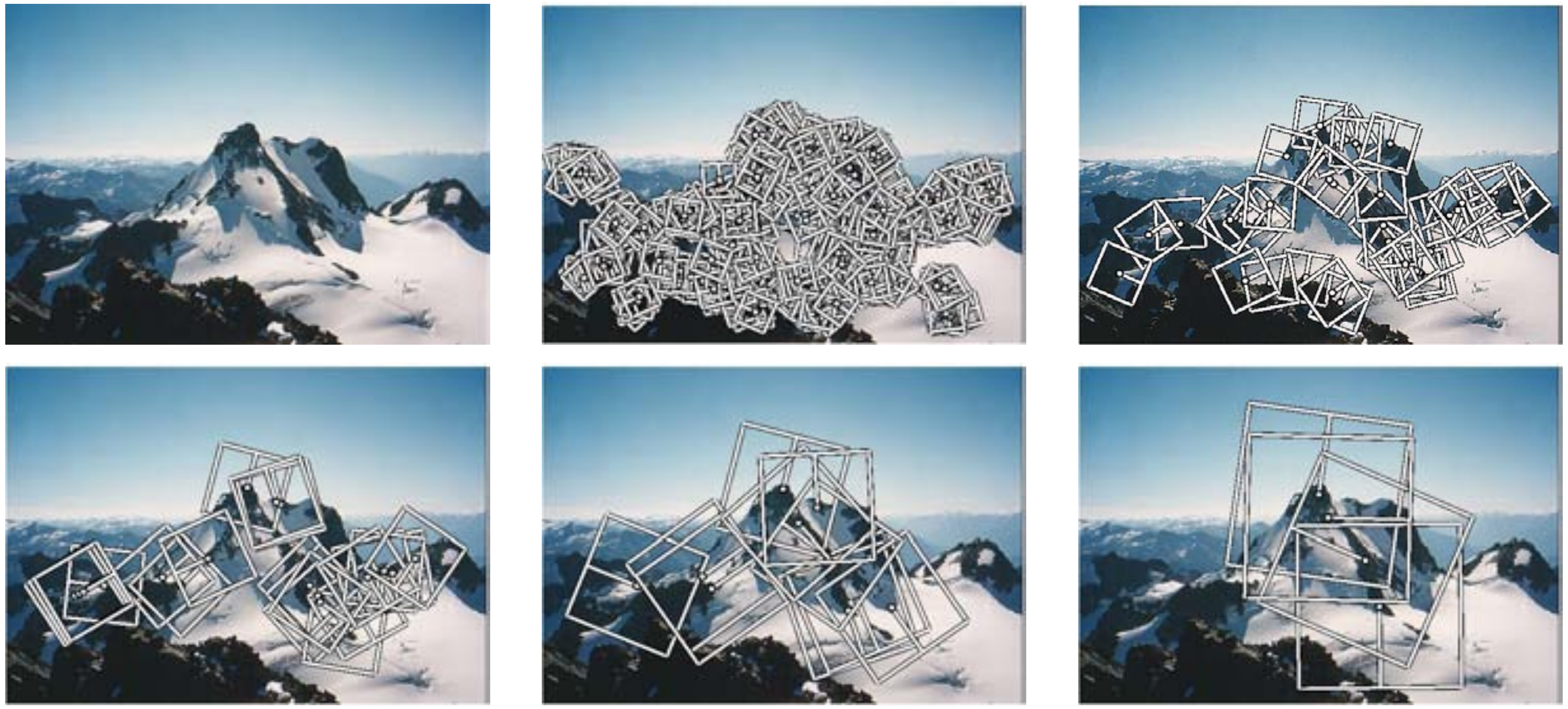
- It is often desirable to be able to detect and track a feature despite changes in scale due to, e.g.,
  - Changes in distance
  - Changes in focal length
- For this purpose, we seek a feature that is stable in both location *and* scale.
  - e.g., extrema (in both location and scale) of Laplacian of Gaussian (LoG) or Difference of Gaussian (DoG) response
    - Lindeberg 1993, Lowe 2004 (SIFT)



End of Lecture
Oct 22, 2018

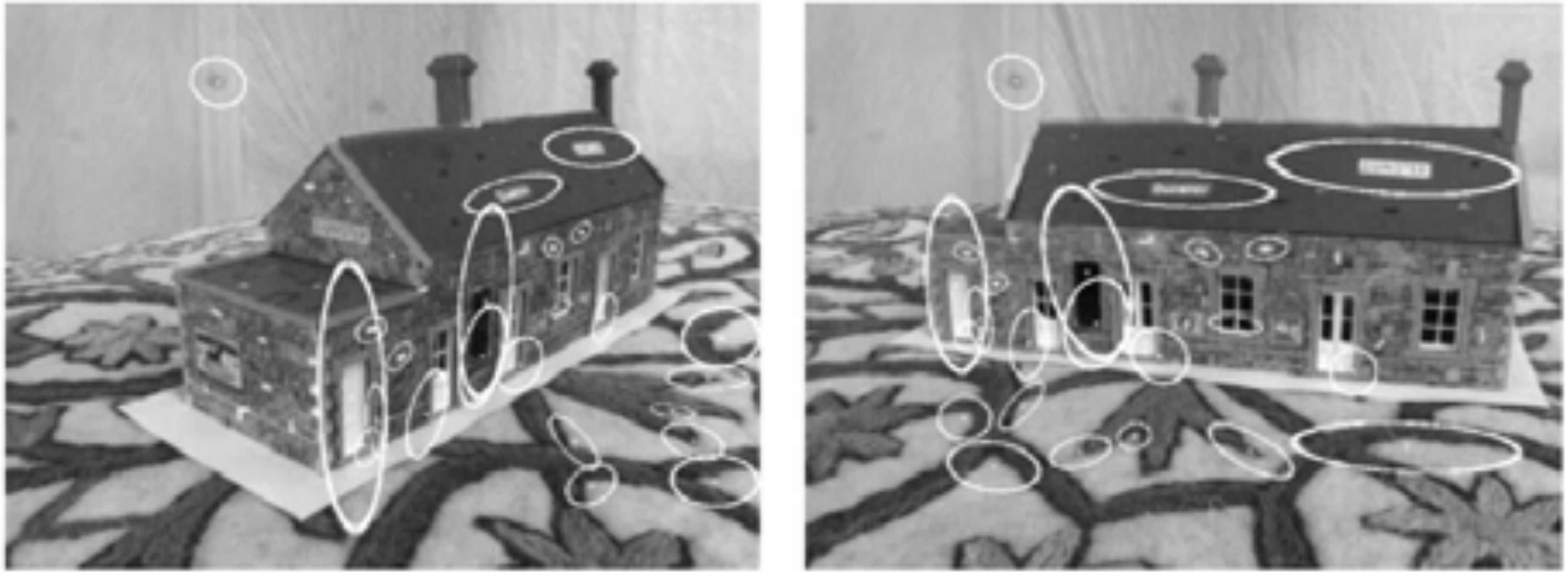
Invariance to In-Plane Rotations

- Objects may also change in orientation between frames.
- Solution 1: Use a rotationally invariant descriptor
  - Problem: such descriptors are not very discriminative - map very different image patches to similar descriptors
- Solution 2: Estimate locally dominant orientation
  - Estimate dominant orientation by averaging the Gaussian gradients within a local patch
  - Then align descriptor in both scale and orientation with detected key point



Affine Invariance

- In general, objects will undergo out-of-plane rotations between views.
- These transformations cannot be accounted for by scaling and rotation within the plane of the image
- However, small out-of-plane rotations can often be handled by building feature detectors that are *affine invariant*.



Feature Detection: State of the Art

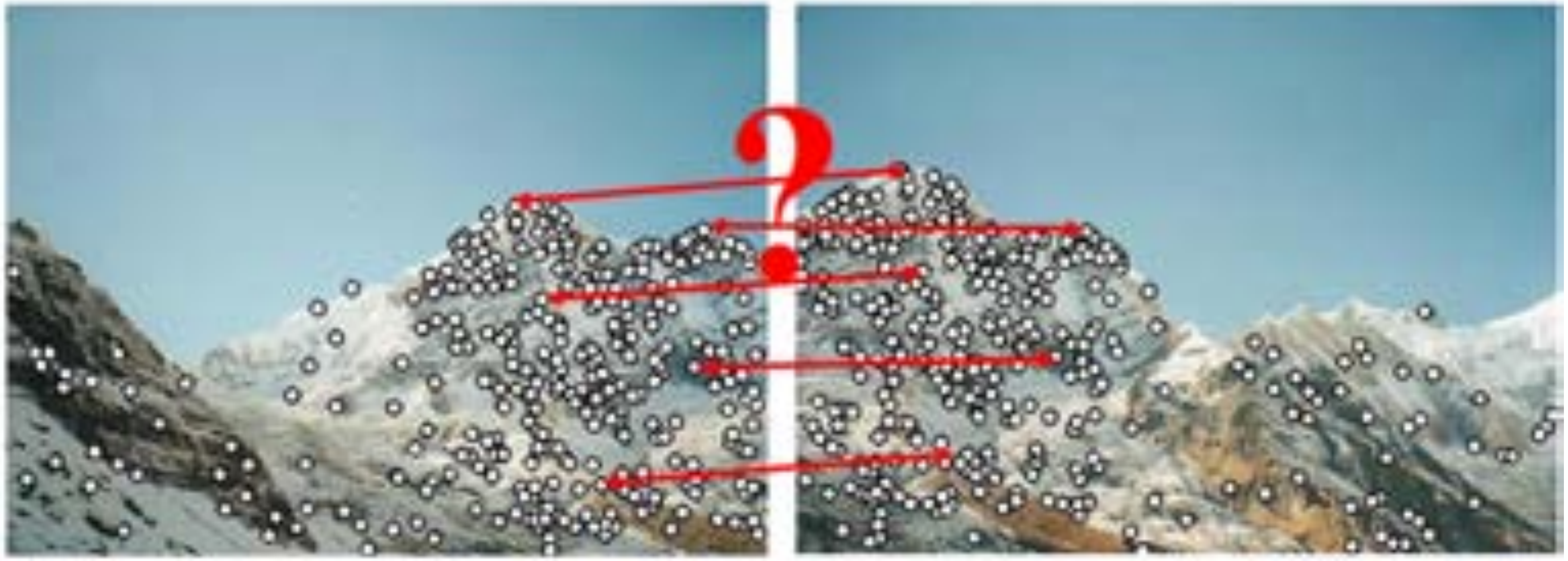
- Machine learning has become an important part of feature detection.
- State-of-the-art for object detection/recognition based on dense deep network features
  - Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in Neural Information Processing Systems, pages 91–99.
  - He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778.

Outline

- Feature detectors
- **Feature descriptors**
- Feature matching
- Feature tracking

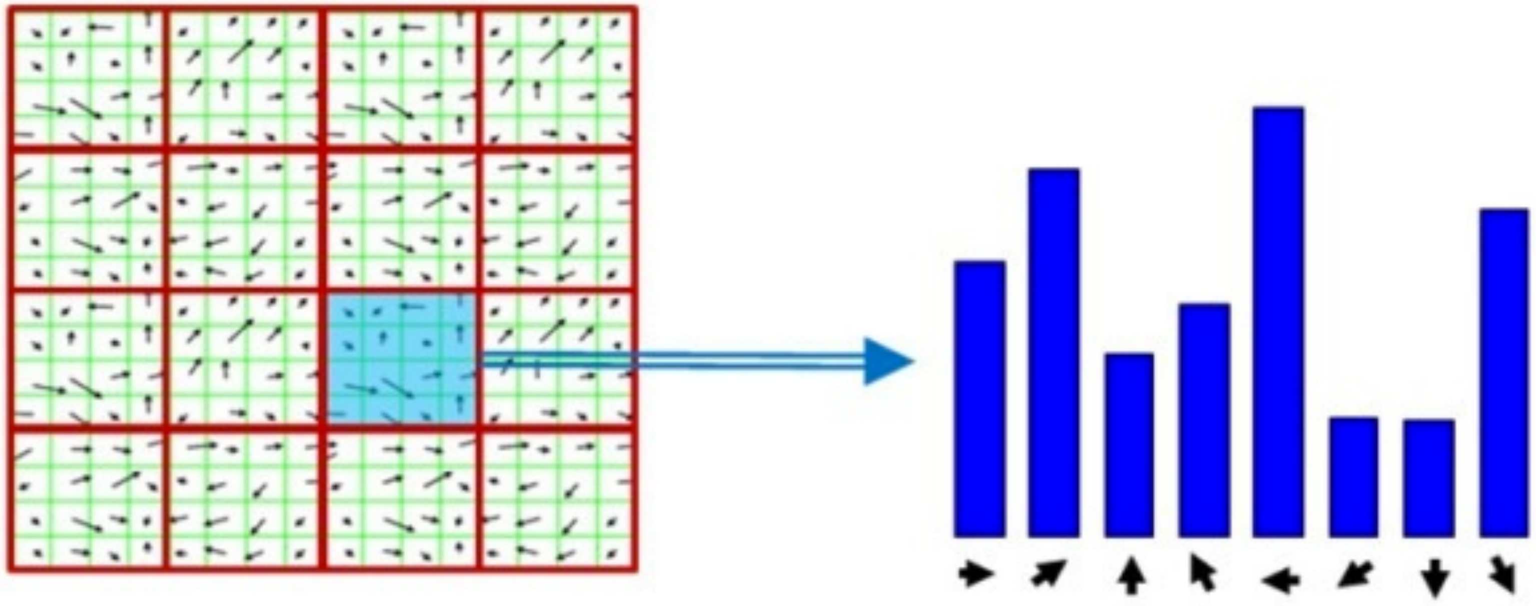
Feature Descriptors

- A 2D spatial pattern or 1D vector describing the appearance of the image patch centred at the keypoint.
- Used to match key points across images for tracking, structure from motion, stereo, object recognition, pose estimation.
- May estimate local scale, orientation and/or affine frame prior to computing descriptor to achieve invariance to these transformations.



Scale-Invariant Feature Transformation (SIFT)

- Keypoint detected at location (x, y) and scale σ in Gaussian pyramid
- Compute intensity gradient at each pixel within 16x16 pixel patch centred at keypoint at scale σ in Gaussian pyramid
- Weight gradients by Gaussian centred at keypoint
- Bin the 16 gradients within each of the 16 4x4 pixel blocks of the patch into an 8-orientation histogram, using gradient magnitude as weight and trilinear interpolation over (x, y, θ)
- Result is a 4 x 4 x 8 = 128-element feature vector.
- Normalize to unit length to increase invariance to photometric variations
- Also cap the maximum gradient magnitude to 0.2 to avoid errors due to camera saturation and larger illumination changes



Lowe, 2004

Outline

- Feature detectors
- Feature descriptors
- **Feature matching**
- Feature tracking

Feature Matching

- Given keypoint A in Image 1 and keypoint B in Image 2, we compute the Euclidean distance *d* between their feature vector. A small distance implies a likely match.

- Fixed threshold θ on distance:
  - $d < \theta \rightarrow$ match
  - $d > \theta \rightarrow$ no match

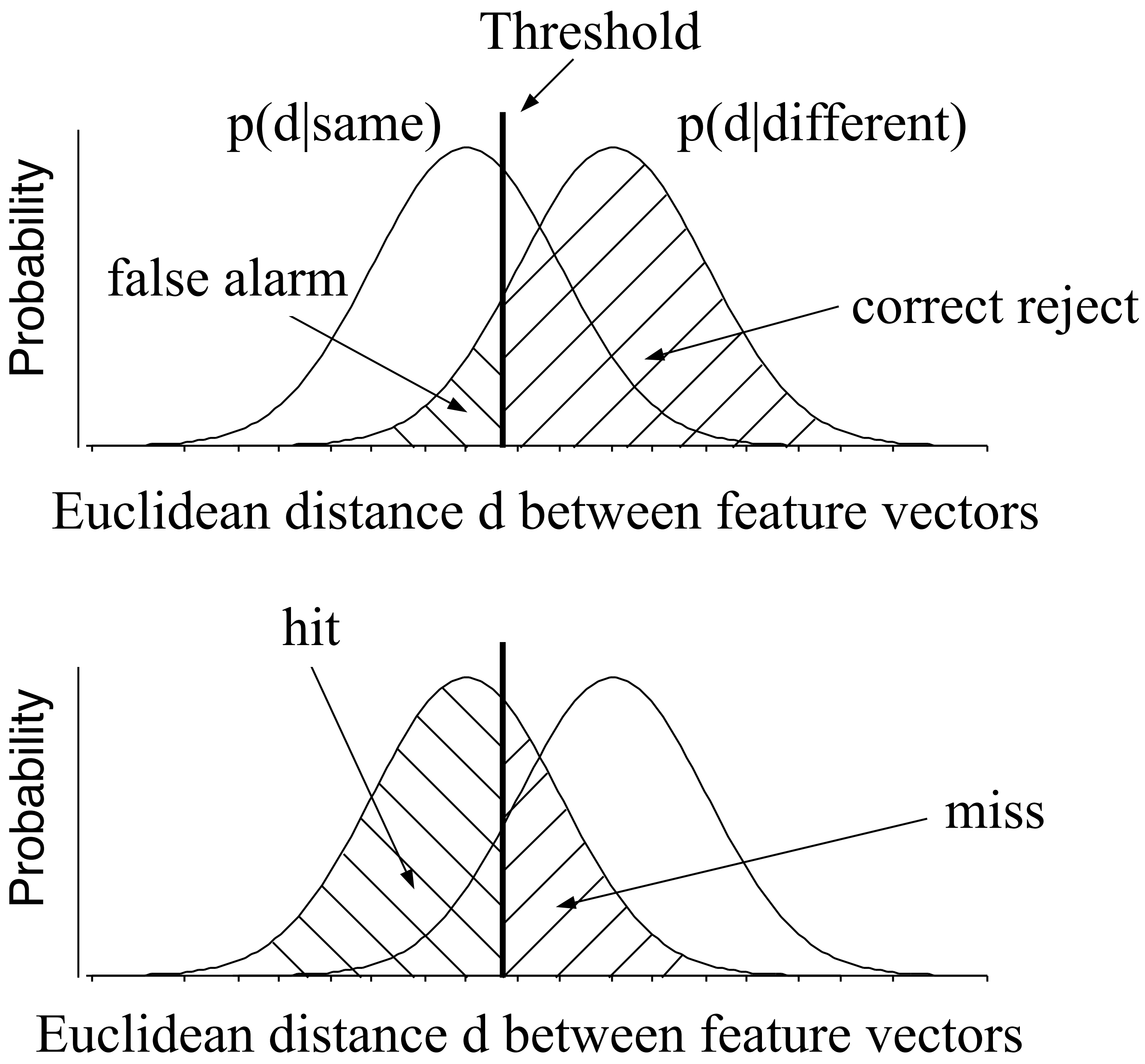
- There are 4 possible outcomes:

Ground Truth

Matching Algorithm

	Match	**Non-Match**
$d < \theta$	Hit	False Positive
$d > \theta$	Miss	Correct Reject

4 Possible Outcomes

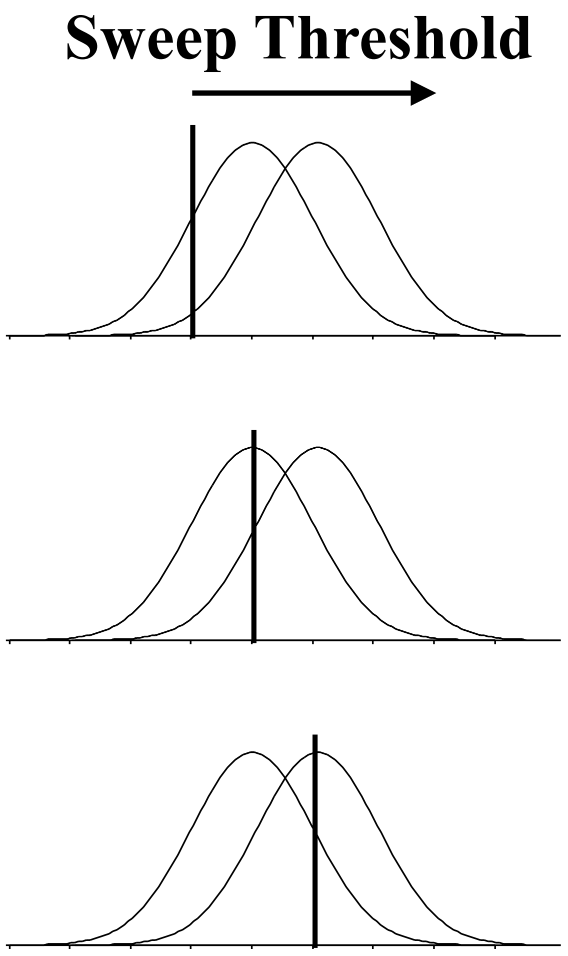


Performance Evaluation

- Let
  - $P = \#$ of ground truth matches
  - $N = \#$ of ground truth non-matches
- Then
  - $\text{Hit Rate} = \dfrac{|\text{Hits}|}{P}$
  - $\text{False Alarm Rate} = \dfrac{|\text{False Alarms}|}{N}$

ROC Plots

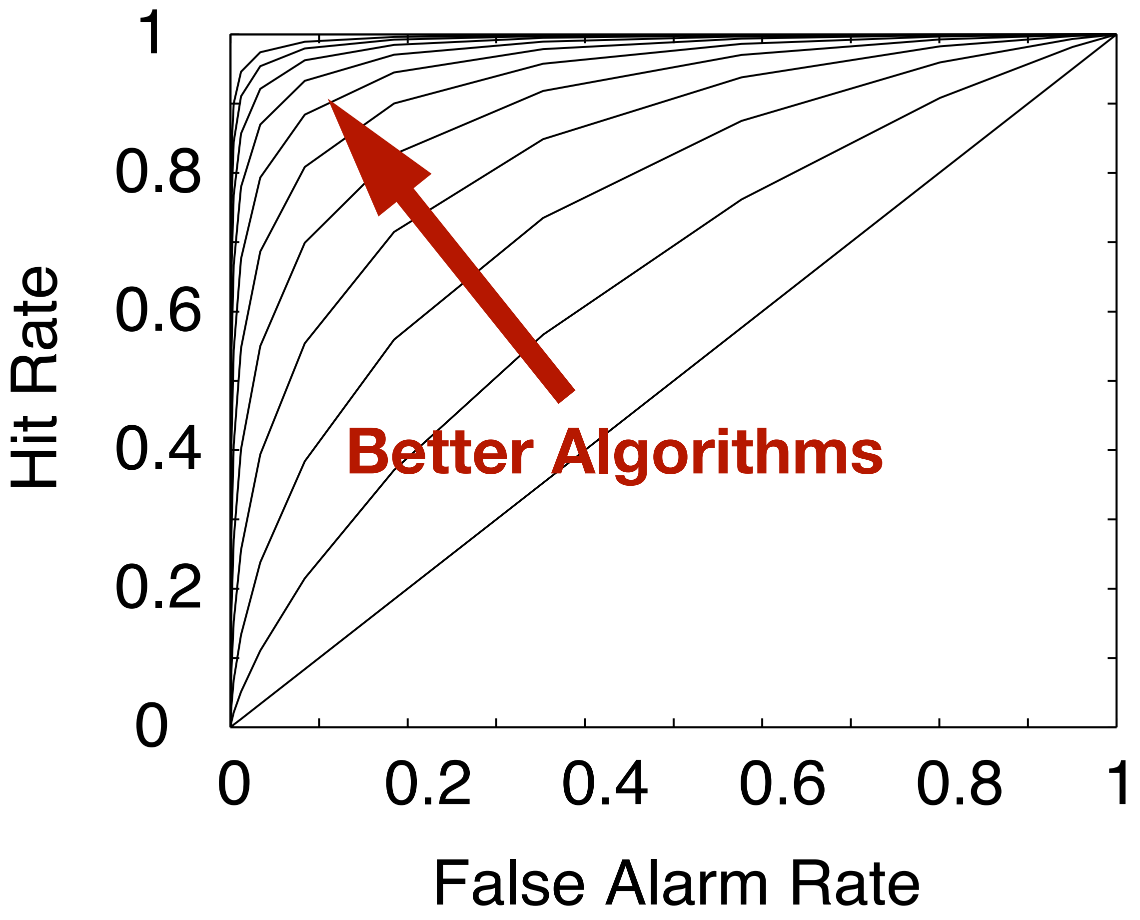
- Algorithms can be compared without committing to a specific threshold using a receiver-operator characteristic (ROC) plot
- Given ground truth data, the optimal threshold can be determined if we know the relative cost of misses and false alarms (decision theory).



Hits = 97.5%
False alarms = 84%

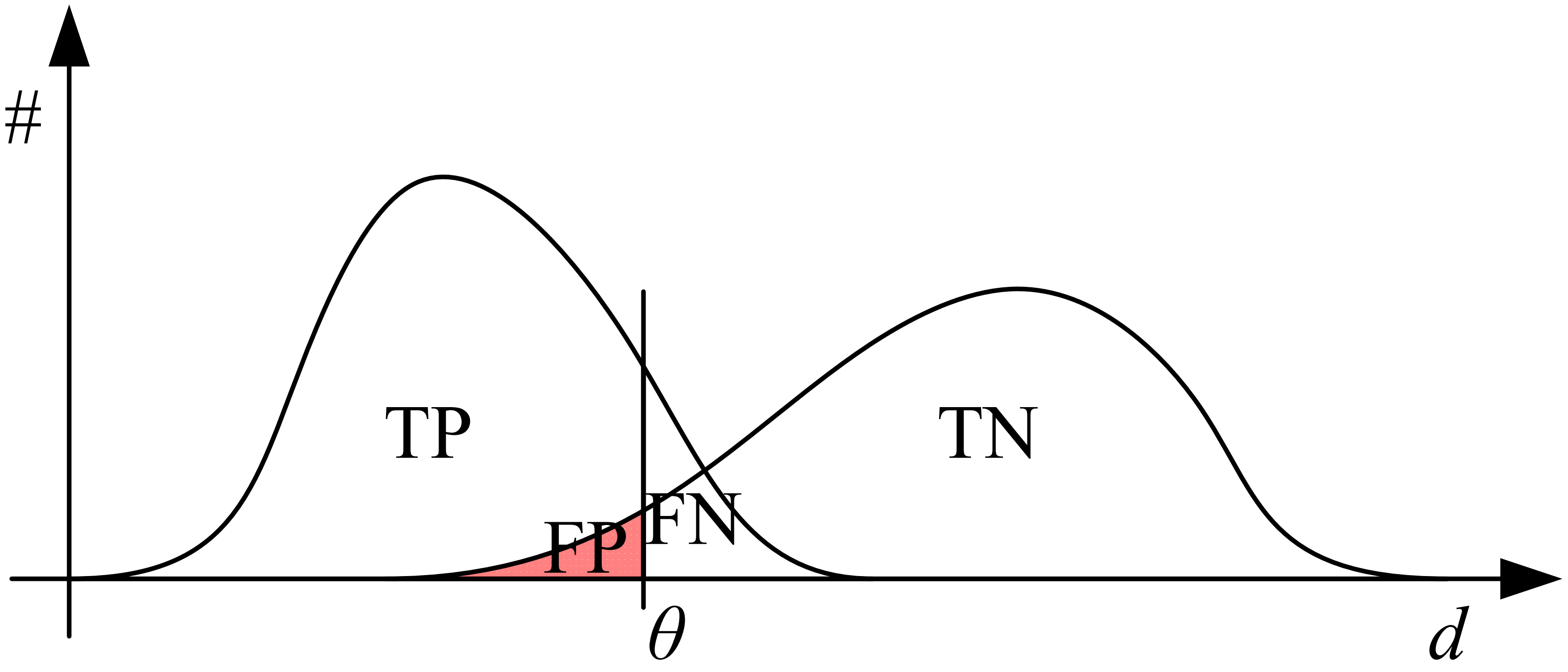
Hits = 84%
False alarms = 50%

Hits = 50%
False alarms = 16%



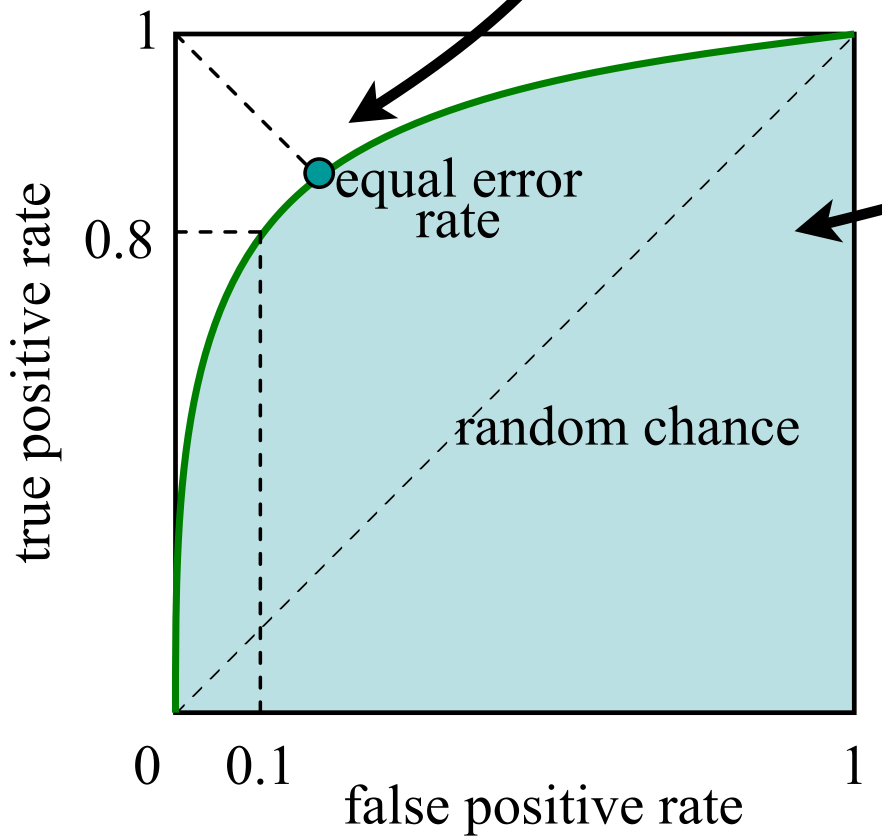
Alternative Terminologies

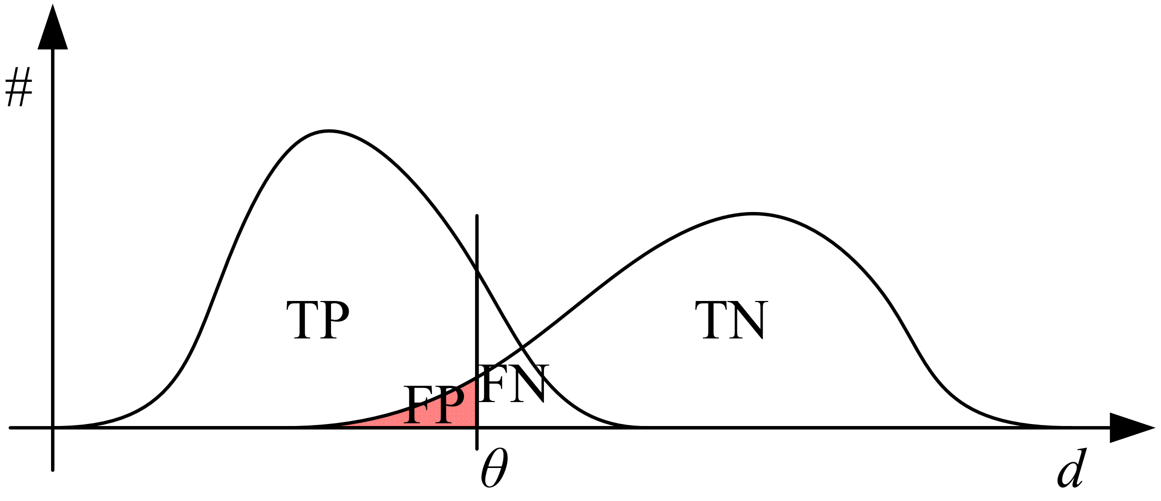
- Hit ≡ True Positive
- Miss ≡ False Negative
- False Alarm ≡ False Positive
- Correct Reject ≡ True Negative



Scalar Measures of Performance

- Area Under the ROC Curve (AUC)
- Equal Error Rate

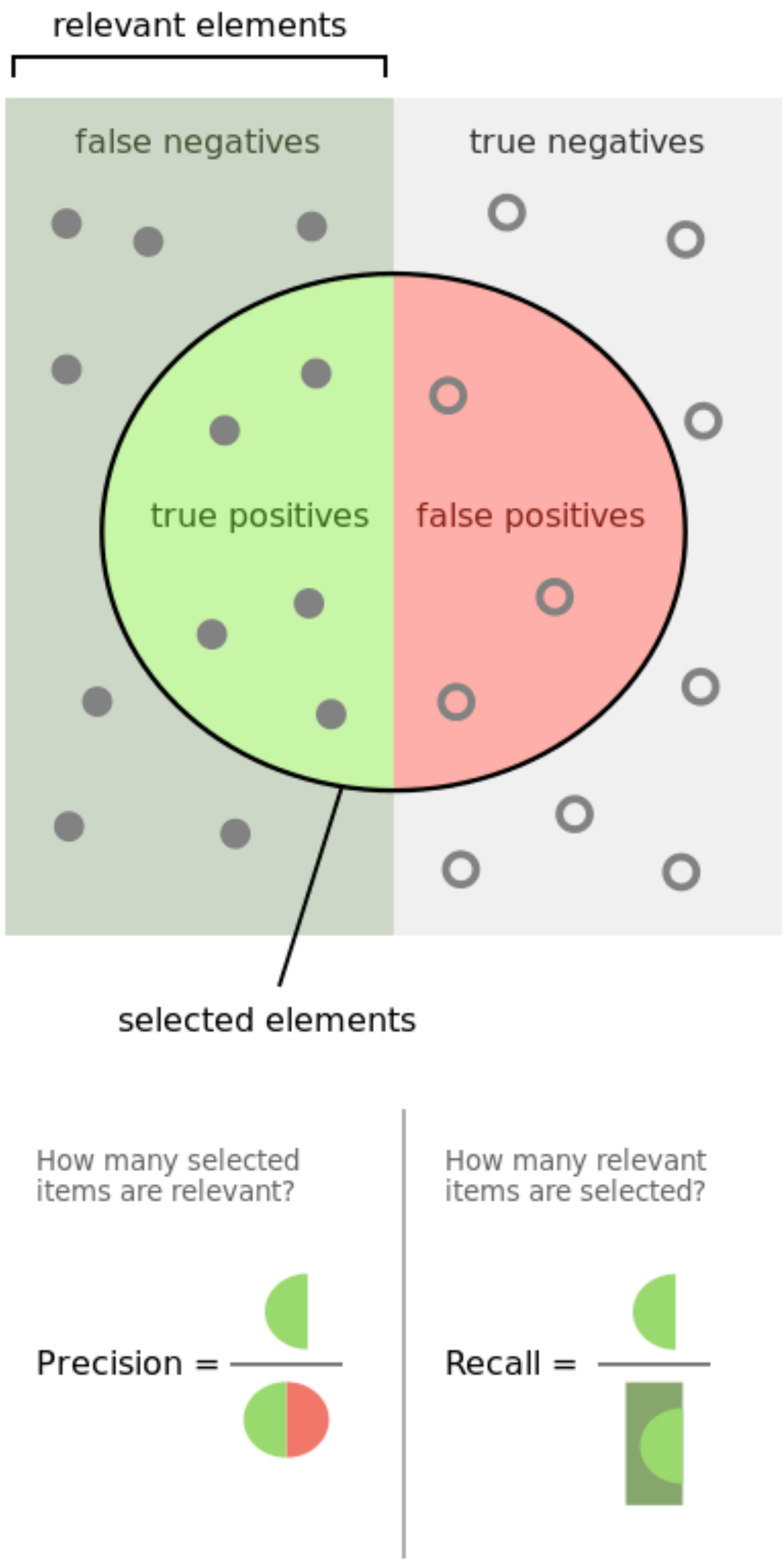




Alternative Terminology: Precision-Recall

- Let
  - p = # of algorithm matches
  - P = # of ground truth matches
- Then
  - $\text{Precision} = \frac{|\text{Hits}|}{p}$
  - $\text{Recall} = \frac{|\text{Hits}|}{P}$

Note: Recall $\equiv$ Hit Rate

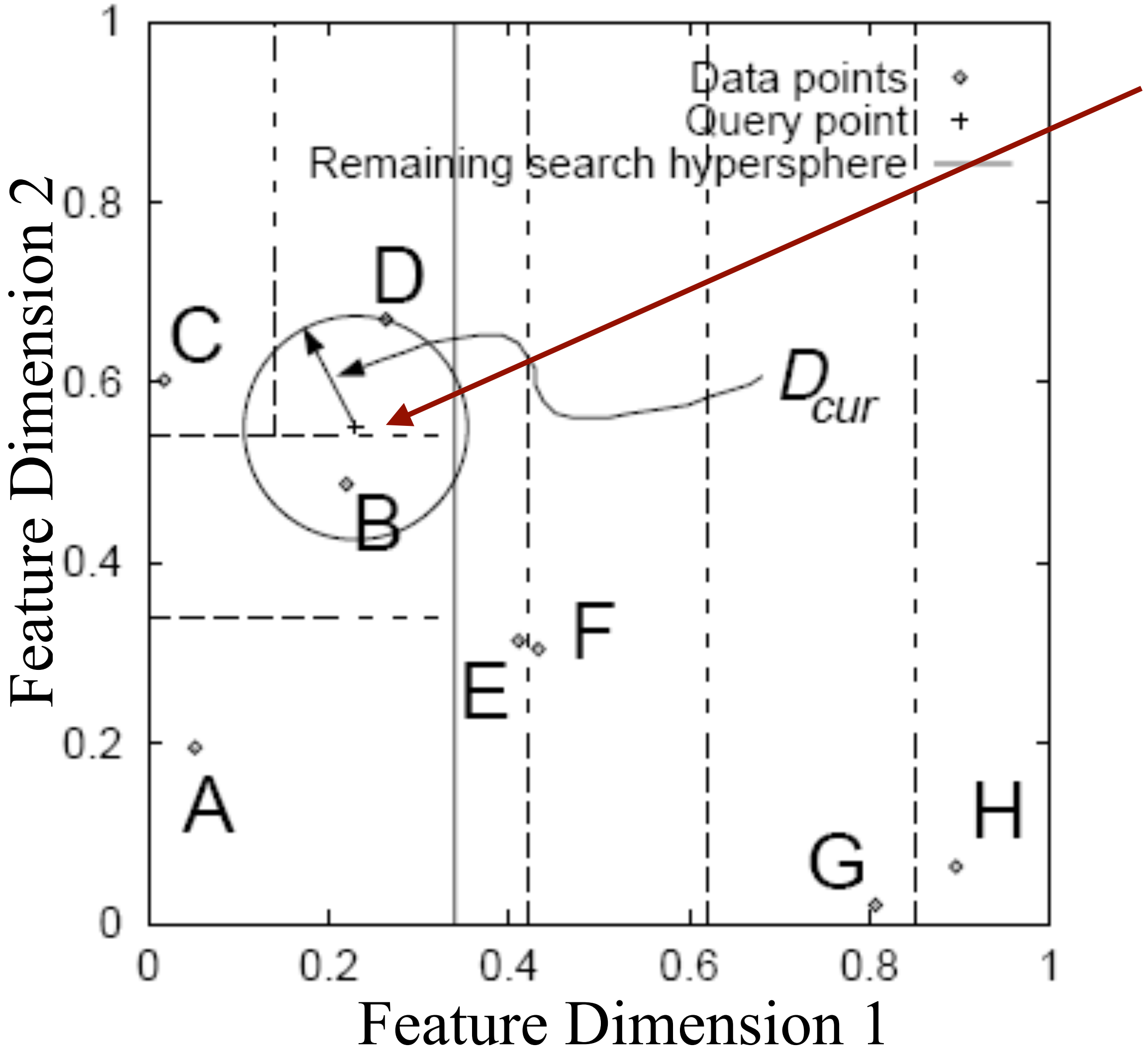


Efficient Matching

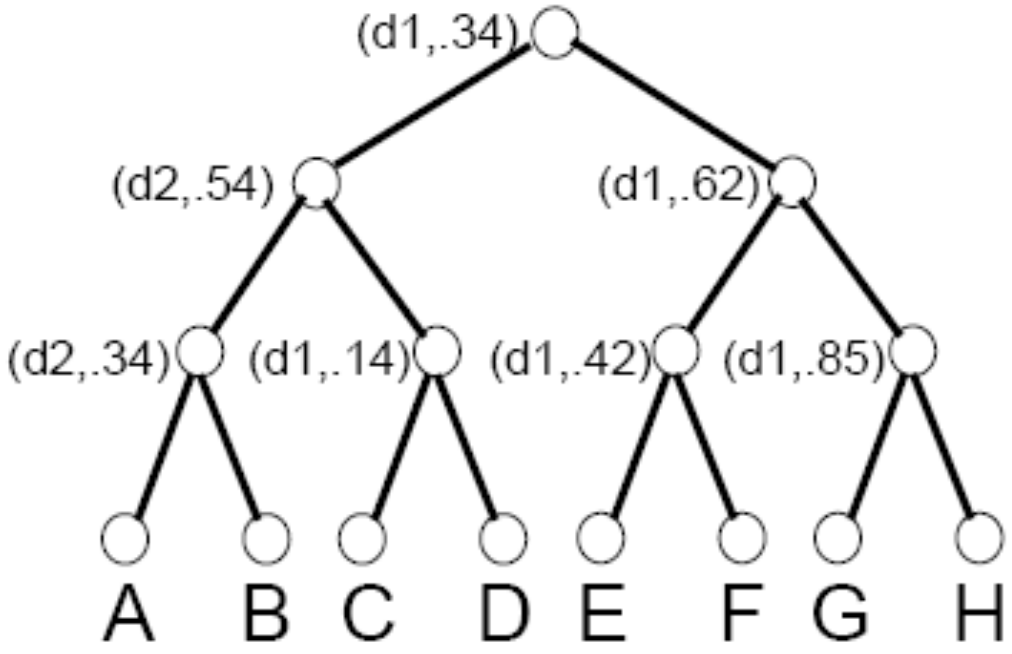
- Exhaustive: Compare all keypoints in Image A to all keypoints in Image B
  - Cost: Quadratic
- More efficient alternatives:
  - Hashing
  - Search trees

Example: k-d Trees

- Consider keypoints A-H.
- Recursively:
  - Select dimension with greatest variance
  - Partition at median
- Partitions can now be represented as binary tree with (dimension, threshold) stored at each node.
- Given query point, Best Bin First (BBF) strategy searches bins in order of proximity to query.



Query point



Beis & Lowe, 1999

Verification & Densification

- Once a set of hypothesized matches are identified, an optimal geometric alignment between the images can be computed.

- This alignment can then be used to prune outlier matches.

- This alternation of alignment and pruning can be iterated to convergence.

- An approximate alignment can also be used to conduct a more constrained search for additional feature matches

- These can be used to further refine the alignment.

Outline

- Feature detectors
- Feature descriptors
- Feature matching
- **Feature tracking**

Feature Tracking

- In some applications deviation between images is small
  - Object tracking in 30fps video
  - Optic flow at 30fps video
- In these scenarios, we may employ a detect-then-track strategy:
  - Detect features in Frame *t*
  - Search for corresponding features in Frame $t + 1$

Correlation Trackers

- Minimize squared deviation (maximize correlation)

$$E_{\mathrm{CC}}(\boldsymbol{u}) = \sum_i I_0(\boldsymbol{x}_i) I_1(\boldsymbol{x}_i + \boldsymbol{u})$$

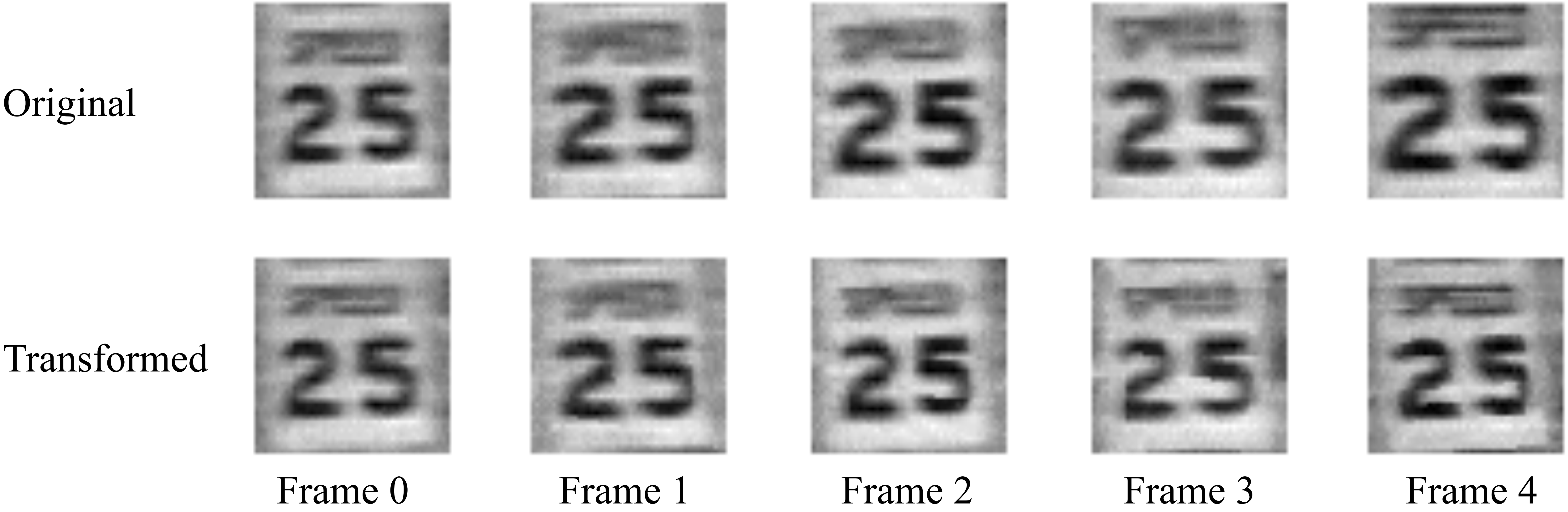
  - Sensitive to photometric changes caused by variation in camera parameters, illumination, specular reflections

- Normalized cross-correlation reduces these effects

$$E_{\mathrm{NCC}}(\boldsymbol{u}) = \frac{\sum_i [I_0(\boldsymbol{x}_i) - \overline{I_0}]\ [I_1(\boldsymbol{x}_i + \boldsymbol{u}) - \overline{I_1}]}{\sqrt{\sum_i [I_0(\boldsymbol{x}_i) - \overline{I_0}]^2}\sqrt{\sum_i [I_1(\boldsymbol{x}_i + \boldsymbol{u}) - \overline{I_1}]^2}}$$

Appearance Drift

- How should we track feature over multiple frames?
  - Match features in Frame 0 to features in all subsequent frames
    - Features may change substantially if object undergoes out-of-plane transformations
  - Re-sample features in each frame
    - Features may drift from original object to other objects
  - KLT Tracker: Use affine motion model to transform frames back to Frame 0 coordinates
    - Only re-sample when tracking fails



Shi & Tomasi 1994

Feature Tracking State of the Art: Learning

- Rather than hardwiring the feature descriptor, one can train a classifier to discriminate a patch on the object to be tracked from background patches, then use this classifier to track.

- This has now led to the application of fast deep networks for tracking, e.g.,

  - H Li, Y Li, F Porikli. Deeptrack: Learning discriminative feature representations online for robust visual tracking, IEEE Transactions on Image Processing, 25(4), 1834-1848, 2016.

Outline

- Feature detectors
- Feature descriptors
- Feature matching
- Feature tracking