Local features

Semester 2, 2025

Kris Ehinger

Outline

- Recognition from local features
- Feature detection
- Feature descriptors

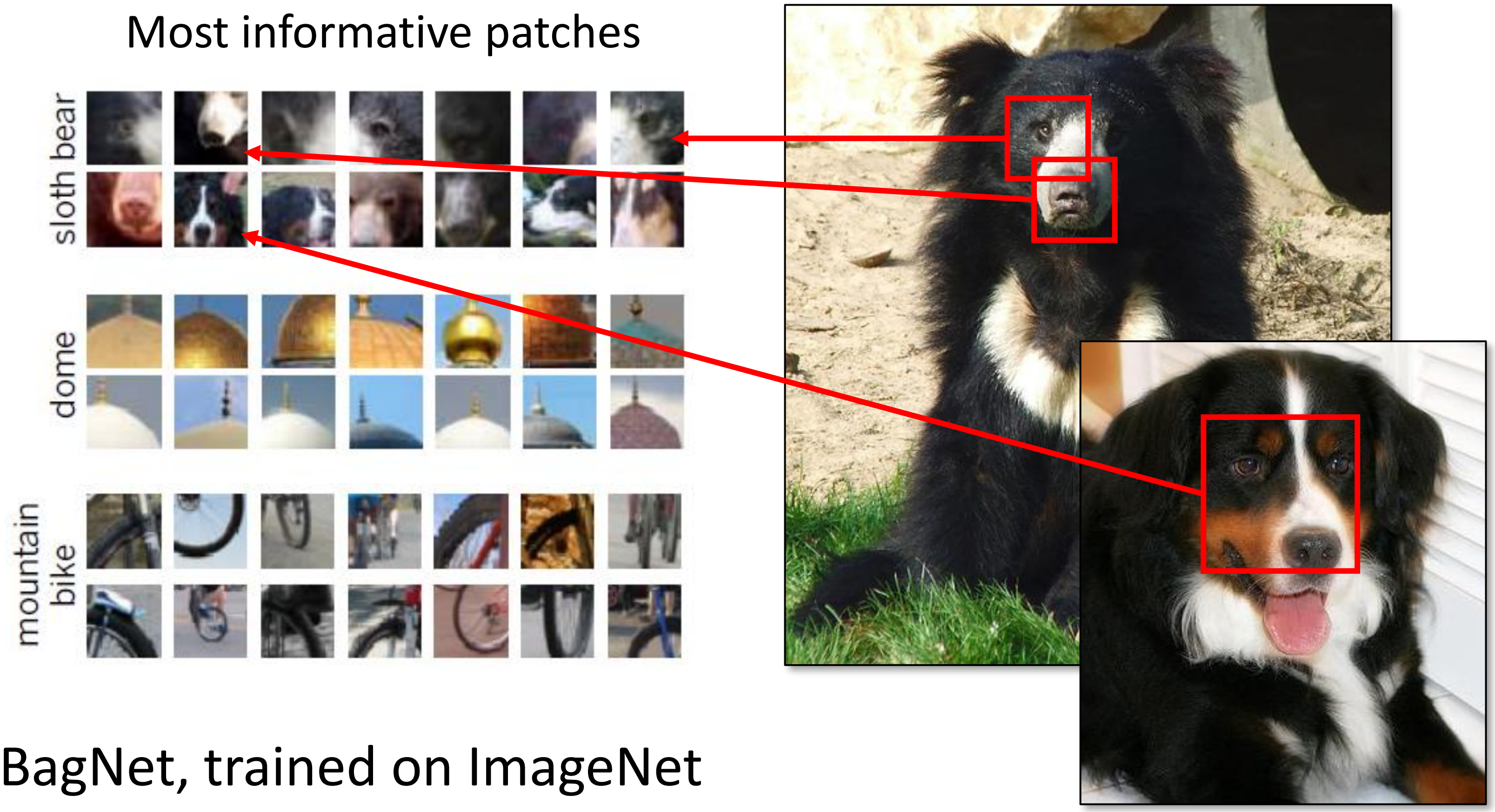
Learning outcomes

- Explain the differences between bag-of-features and feature-detection-based approaches and their applications
- Implement an algorithm for feature detection (Harris corners)
- Explain the desirable properties of feature descriptors

Approaches to recognition

CNN object recognition

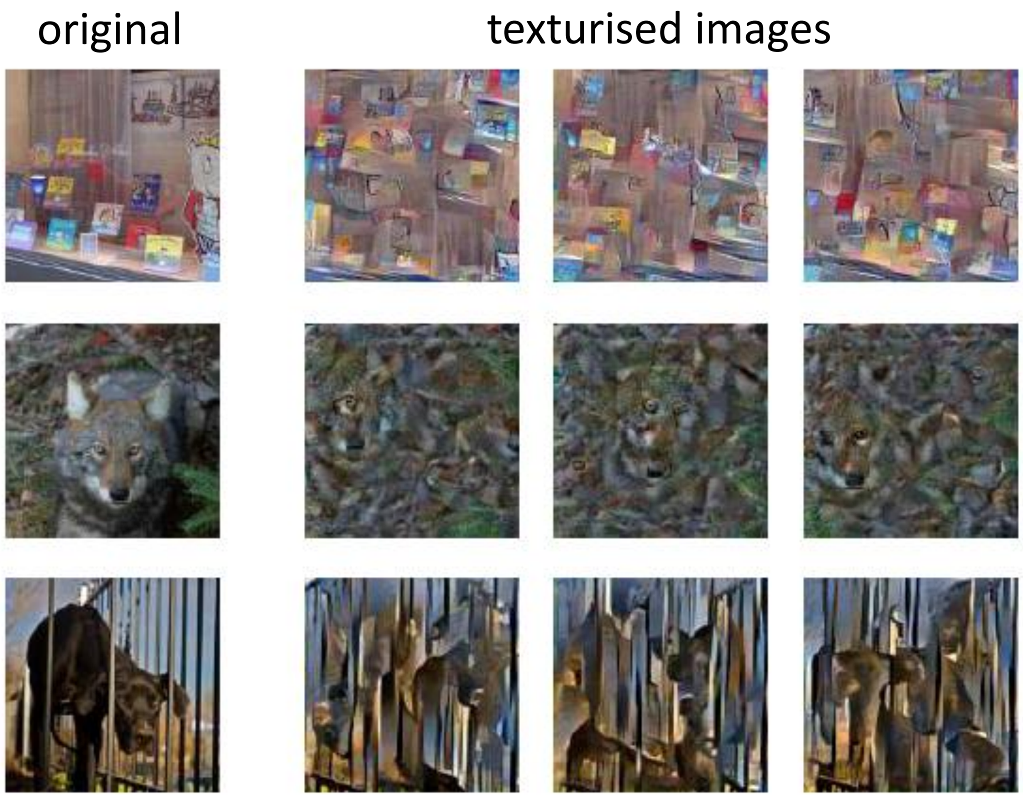
Most informative patches



BagNet, trained on ImageNet

Brendel & Bethge (2019)

CNN object recognition



VGG-16, trained on ImageNet

Performance drop: 90% → 79%

Image: Brendel & Bethge (2019)

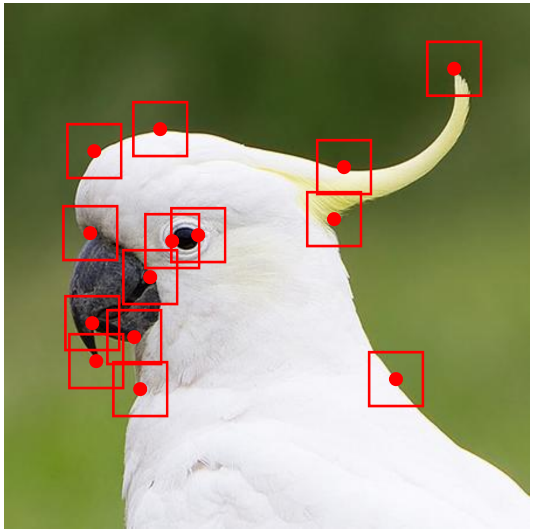
Approaches to recognition

- Detect local features, ignore spatial position
  - Example: bag of words / bag of features
- Local features + weak spatial relations
  - Spatial pyramid models
- Detect local features and model spatial relations between them
  - Deformable-parts models
  - Keypoint tracking / matching

Feature detection

Dense vs. sparse features

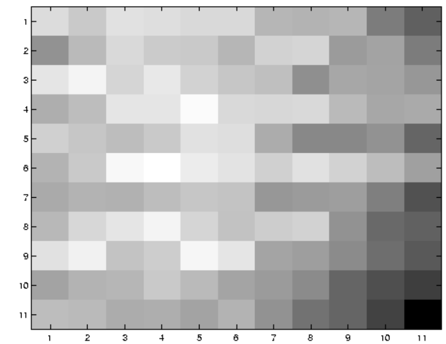
- Dense feature representation: compute local features everywhere
- Sparse feature representation : compute local features only at a few “important” points



Definitions

- **Feature detection:** finding "important" points (interest points or keypoints) in images
  - What's important?
  - Generally, points that can be detected reliably across image transformations
- **Feature descriptor:** a short code or set of numbers to represent a point in an image

What makes a good keypoint?



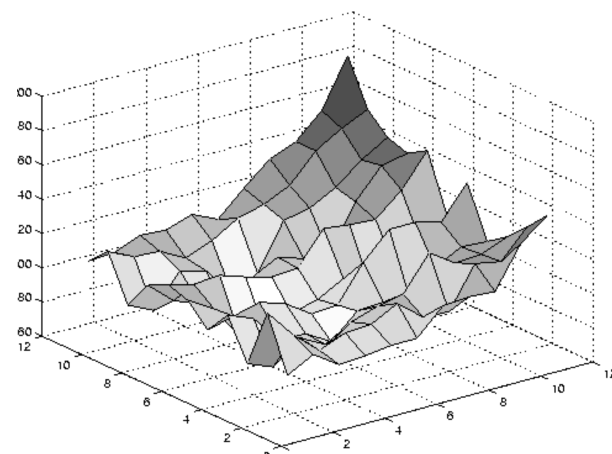


Figure: R. Szeliski

What makes a good keypoint?

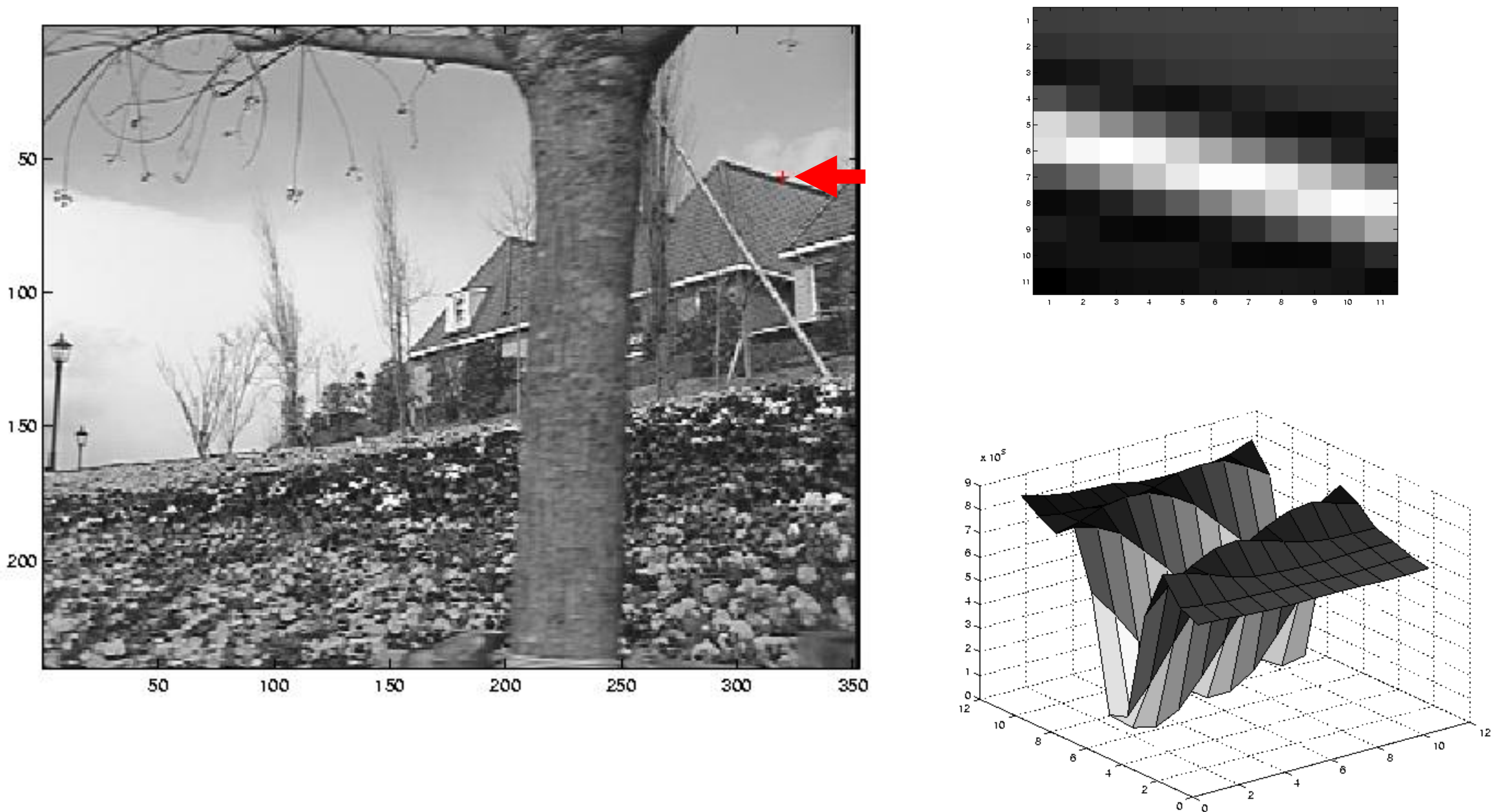
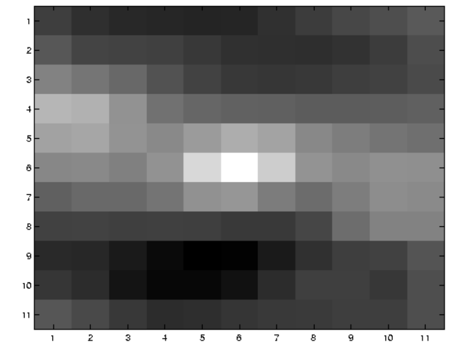
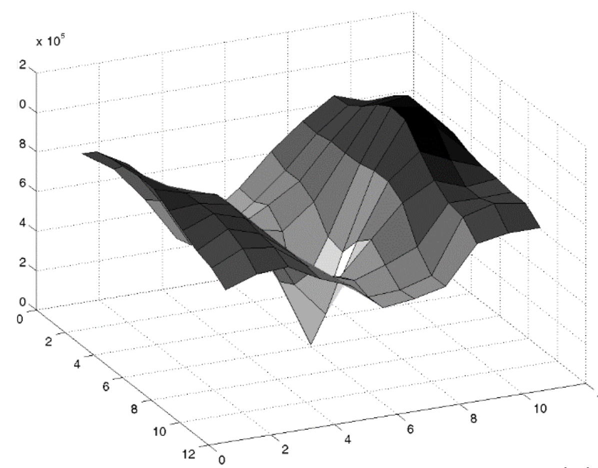


Figure: R. Szeliski

What makes a good keypoint?



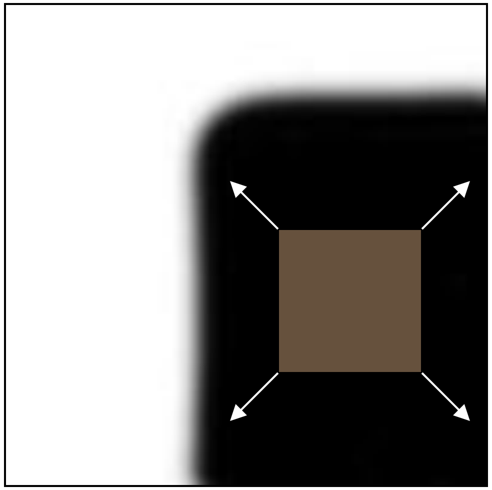


pollev.com/krisehinger432

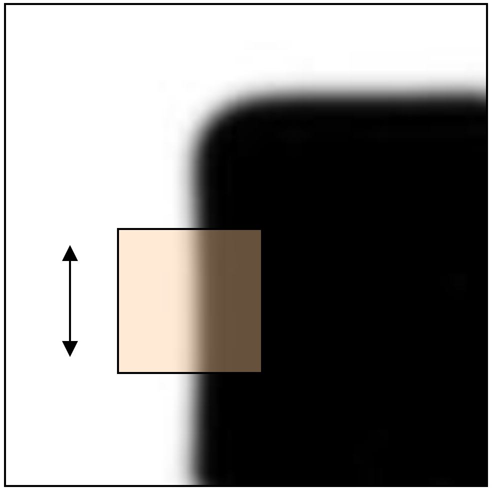
Figure: R. Szeliski

Selecting good keypoints

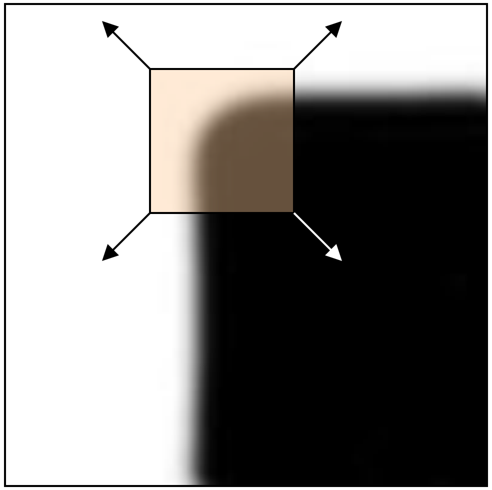
- Should be easy to recognize in a small window
- Shifting the window in any direction should produce a large change in intensity



Uniform = no change in any direction



Edge = no change along edge direction



Corner = change in all directions

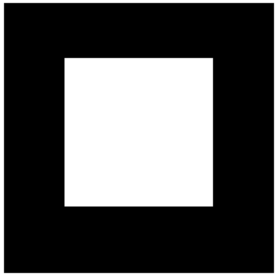
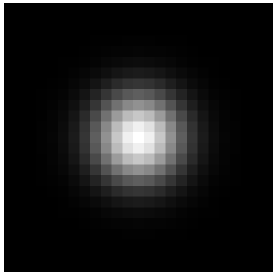
Slide: K. Grauman

Corner detection

- Change in appearance of window w(x,y) for the shift [u,v]:

$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{Window function}} \big[\underbrace{I(x+u, y+v)}_{\text{Shifted intensity}} - \underbrace{I(x,y)}_{\text{Intensity}} \big]^2$$

Common window functions: square, Gaussian

Corner detection

- Change in appearance of window w(x,y) for the shift [u,v]:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

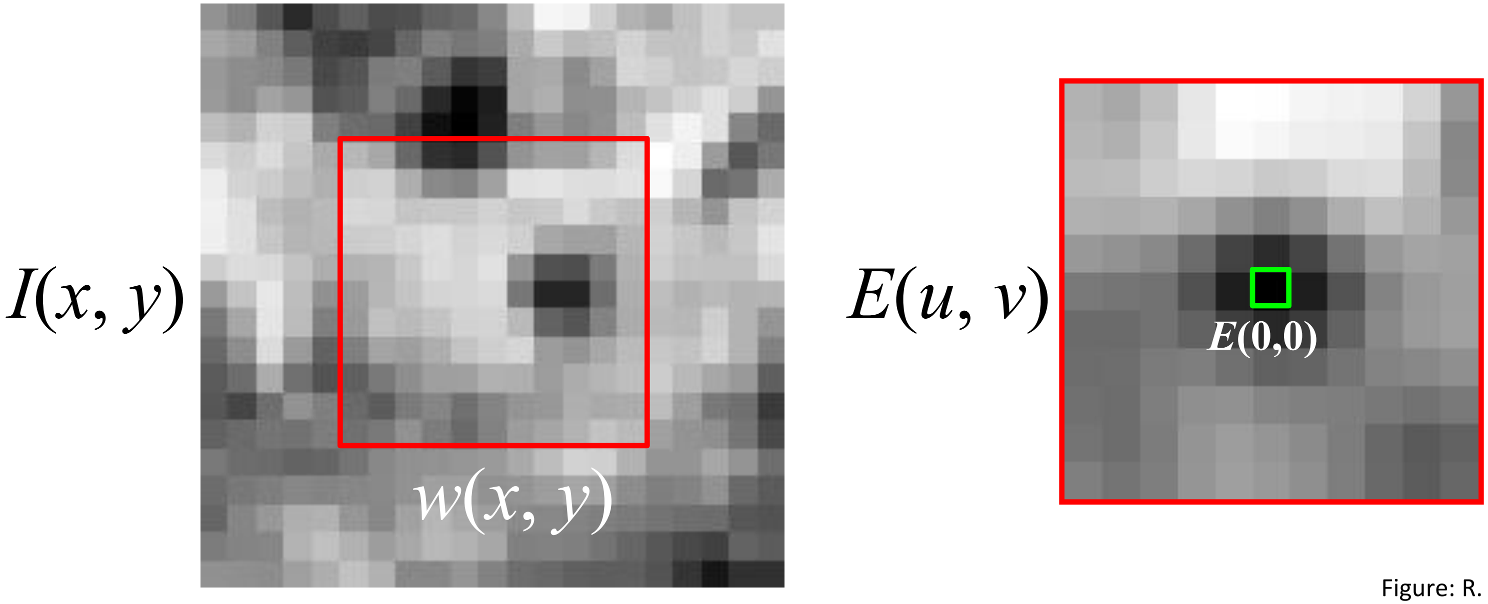
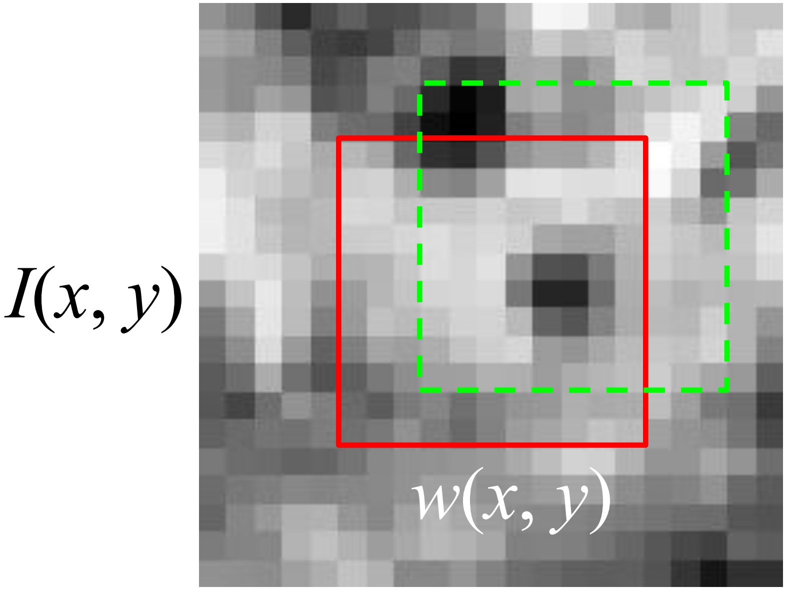


Figure: R. Szeliski

Corner detection

- Change in appearance of window w(x,y) for the shift [u,v]:

$$E(u,v)=\sum_{x,y} w(x,y)\left[I(x+u,y+v)-I(x,y)\right]^2$$



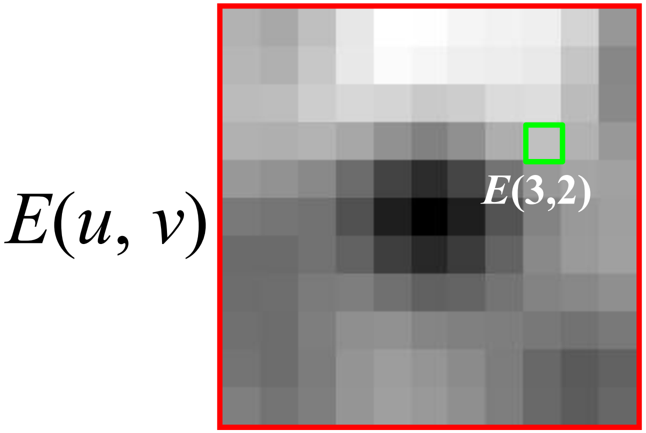
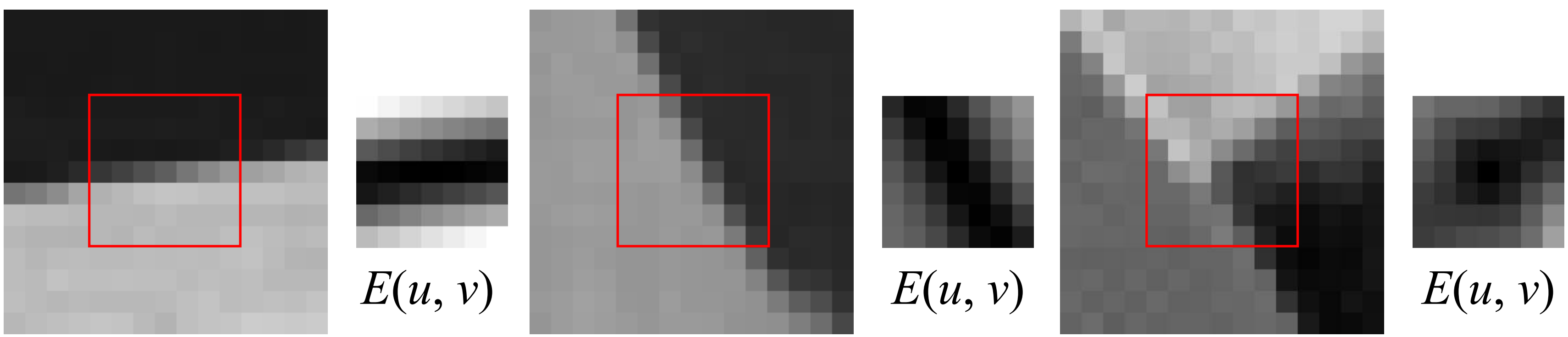


Figure: R. Szeliski

Corner detection

- Change in appearance of window w(x,y) for the shift [u,v]:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$



Corner detection mathematics

- Approximate shifted intensity using Taylor series:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

$$E(u,v) \approx \sum_{x,y} w(x,y)[I(x,y) + u\underbrace{I_x}_{\frac{\partial I}{\partial x}} + v\underbrace{I_y}_{\frac{\partial I}{\partial y}} - I(x,y)]^2$$

$$= \sum_{x,y} w(x,y)[uI_x + vI_y]^2$$

$$= \sum_{x,y} w(x,y)\begin{pmatrix} u & v \end{pmatrix}\begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}\begin{pmatrix} u \\ v \end{pmatrix}$$

Corner detection mathematics

- Change in appearance of window w(x,y) for the shift [u,v]:

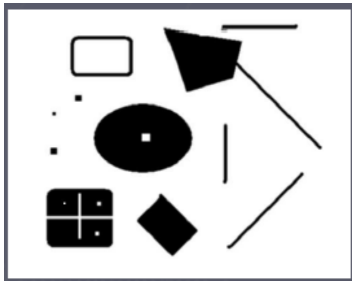
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

- Simplifies to: $E(u,v) \approx [u \ \ v] \ M \begin{bmatrix} u \\ v \end{bmatrix}$

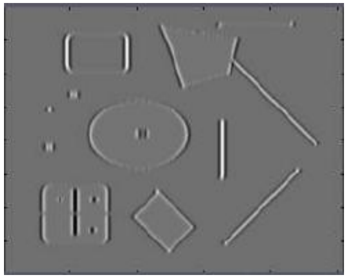
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Values of M

$$M = \sum w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

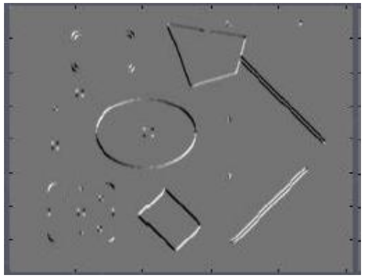


Image



$$I_x = \frac{\partial I}{\partial x}$$

$$I_y = \frac{\partial I}{\partial y}$$



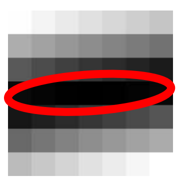
$$I_x I_y = \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$$

Slide: K. Grauman

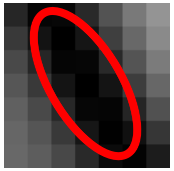
Corner response function

- Detect corners using eigenvalues λ_1, λ_2 of M

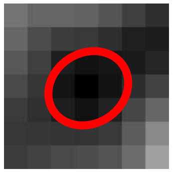
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



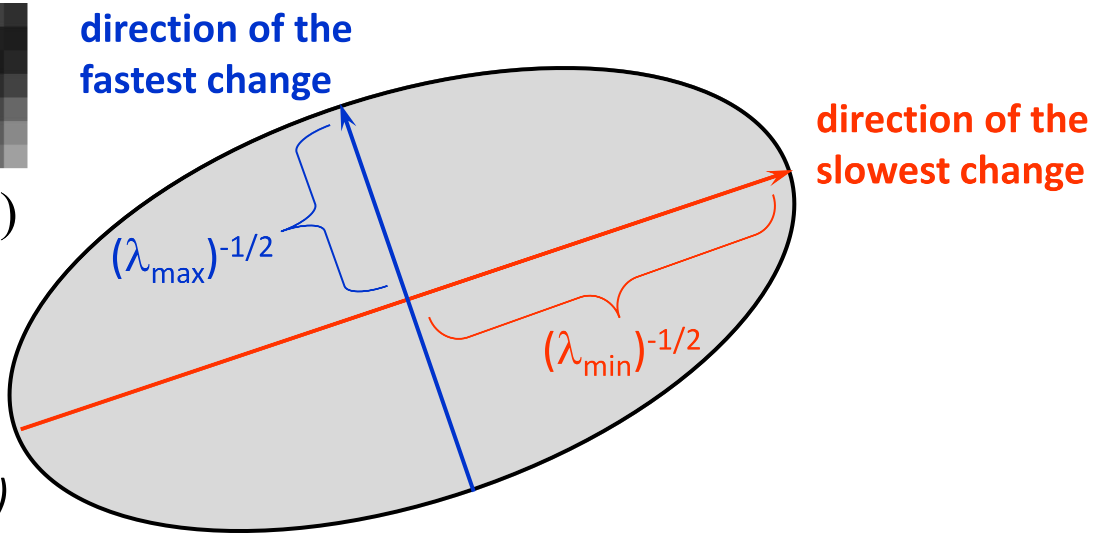
$E(u, v)$



$E(u, v)$



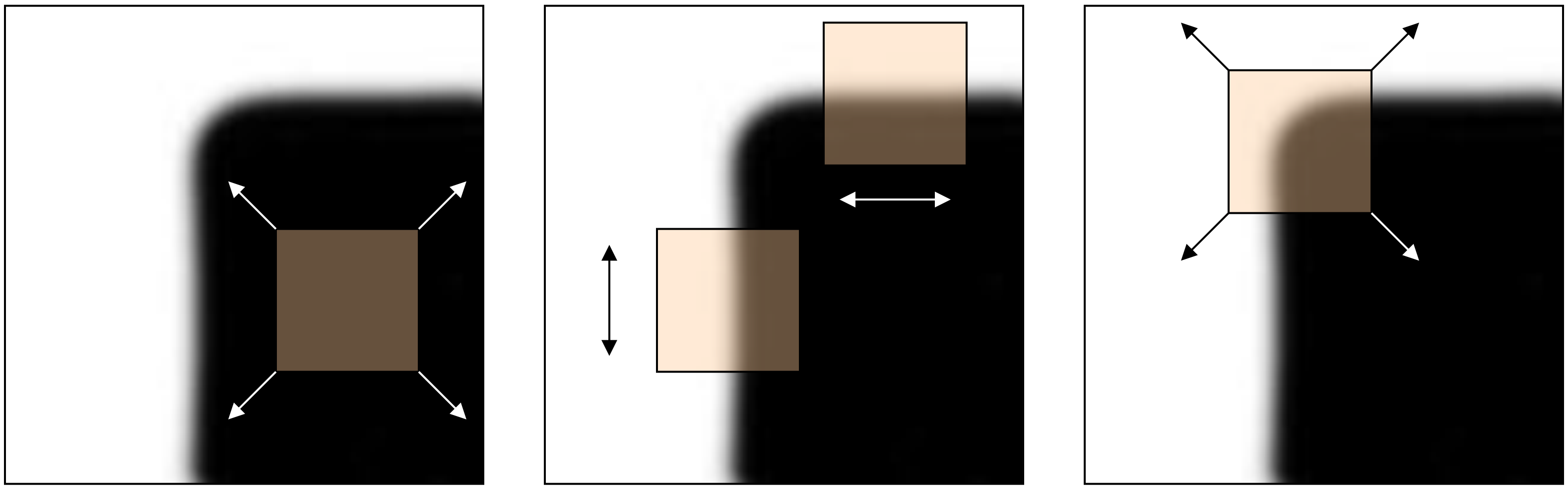
$E(u, v)$



Iso-intensity contour of *E(u,v)*

Figure: R. Szeliski

Corner response function



Uniform: λ_1 and λ_2 are small

Edge: $\lambda_1 >> \lambda_2$
$\lambda_2 >> \lambda_1$

Corner: λ_1 and λ_2 are large

To find corners, look for points where $\lambda_1\lambda_2$ is high, and $\lambda_1 + \lambda_2$ is low

Figure: K. Grauman

Harris corners

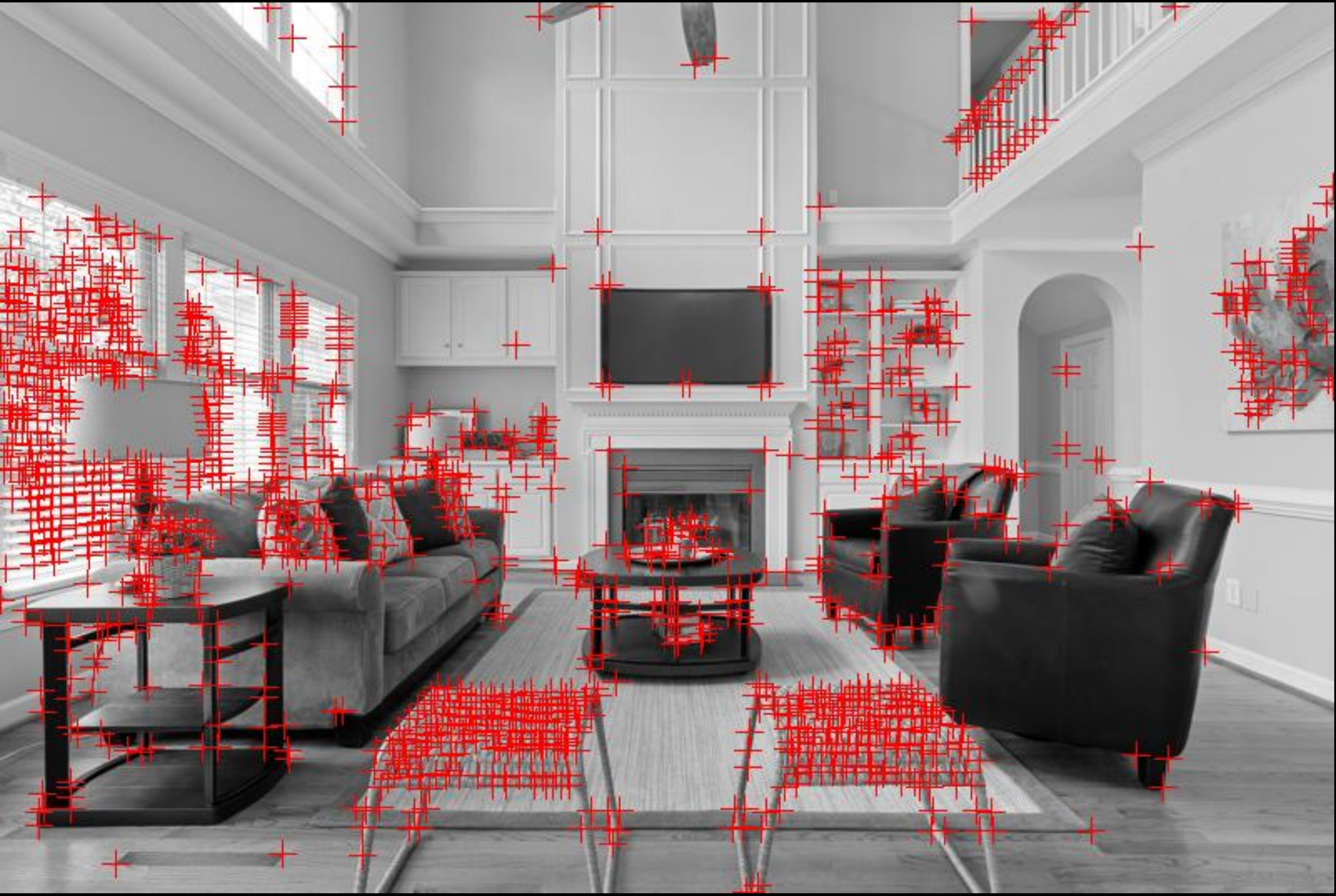
- $\lambda_1\lambda_2$ and $\lambda_1 + \lambda_2$ are the determinant and trace of matrix M:
  - det = np.linalg.det(m) $\quad \det(M) = \lambda_1\lambda_2$
  - trace = m.trace() $\quad \mathrm{tr}(M) = \lambda_1 + \lambda_2$

- Harris corner response: $R = \det(M) - k(\mathrm{tr}(M))^2$

k determined empirically, around 0.04-0.06

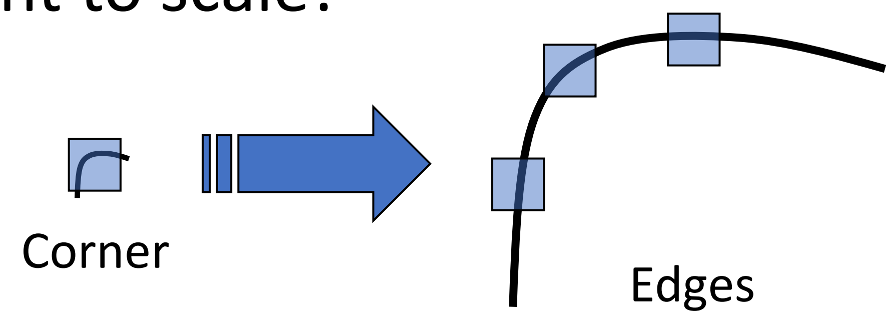
Harris corner response = $\det(M)-k(\mathrm{tr}(M))^2$

Harris corners



Invariance / tolerance

- Corner detection is based on the image gradient (edges), so it's
  - Invariant to translation
  - Tolerant to changes in lighting
- Because the corner response is based on eigenvalues, it is invariant to image-plane rotation
- Not invariant to scale!



Alternatives to Harris corners

- Alternative corner response functions:
  - Shi-Tomasi (1994): $\min(\lambda_1, \lambda_2)$
  - Brown, Szeliski, & Winder (2005): $\dfrac{\det M}{\operatorname{tr} M}$
- Alternatives to corner detectors:
  - Blob detectors
  - Machine-learning-based detectors

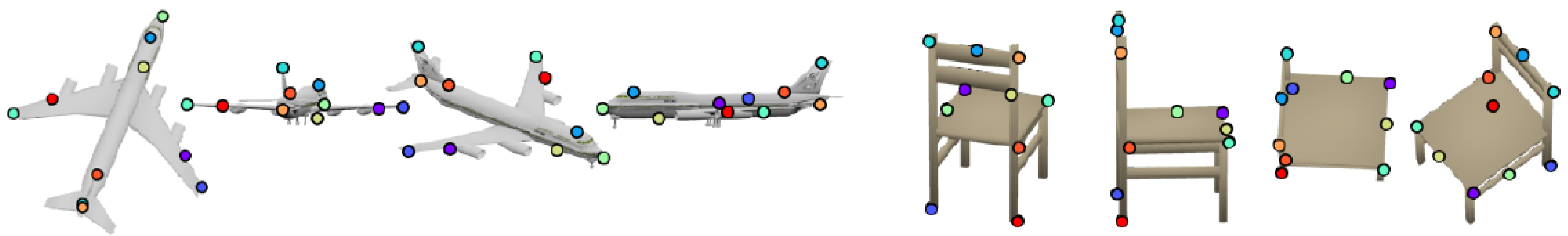


Image: Suwajanakorn, Snavely, Tompson, & Norouzi (2018)

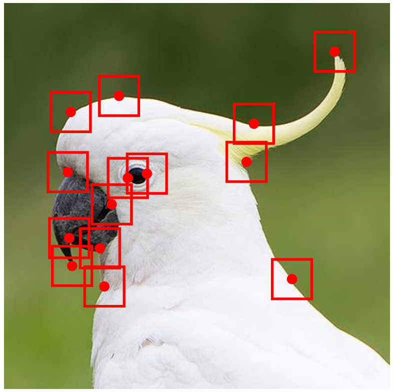
Summary

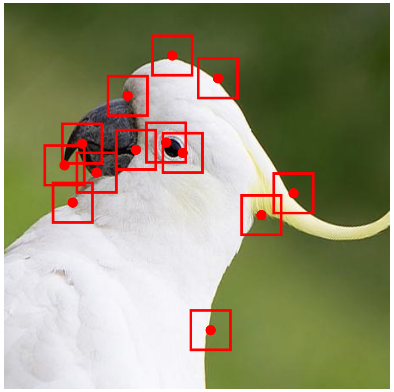
- Rather than detecting local features everywhere, feature detectors can be used to find “important” points (interest points or keypoints)
- Common type of interest point = corners
- Corners can be detected from local gradients

Feature descriptors

Feature descriptors

- Having found keypoints in an image, we need a way to represent them
- Options:
  - Image patch
  - Handcrafted descriptors
    - SIFT
    - GLOH
    - BRIEF
    - BRISK
    - ORB
  - Machine-learned descriptors
    - DeTone, Malisiewicz, & Rabinovich (2018)





Scale-Invariant Feature Transform (SIFT)

- Compute gradient, take histograms in a grid of pixels around interest point
- Weight gradient magnitudes based on distance from centre of patch
- Normalise histograms to sum to 1

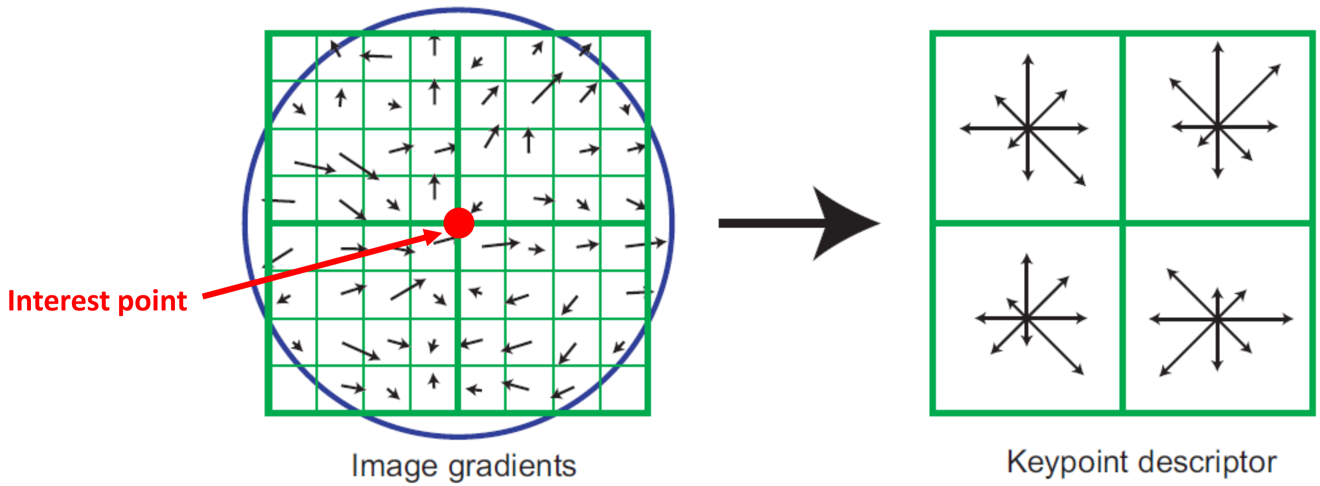


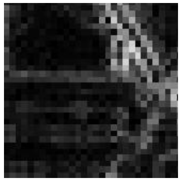
Image: D. Lowe

Scale-Invariant Feature Transform (SIFT)

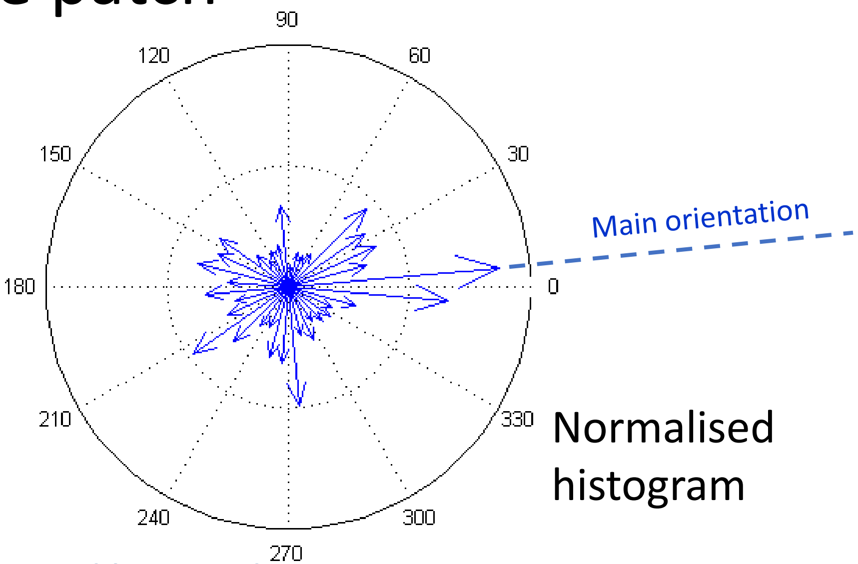
- SIFT implementation details:
  - Patch size = 16 x 16 pixels
  - Grid = 4 x 4 cells
  - Histogram bins = 8 orientations
  - Gaussian weighting from centre of patch
  - Two-step normalisation: normalise to sum to 1, truncate values to 0.2, normalise to sum to 1
- Descriptor length = 4 x 4 x 8 = 128

Scale-Invariant Feature Transform (SIFT)

- Interest points (blobs) are detected at multiple scales; descriptor is based on the scale with maximum response
- Histograms are encoded relative to the main orientation in the patch



Patch



Normalised histogram

Summary

- Feature descriptor = a code to represent a local patch or interest point in an image
- Many handcrafted feature descriptors, with different:
  - Encoding method
  - Speed
  - Descriptor size
  - Feature detection method
- Goal of feature descriptors is invariance, so points can be matched reliably across image transforms

Summary

- Most recognition approaches are based on local features, but differ in how they represent spatial relations between features:
  - Bag-of-features methods: no spatial information
  - Feature detection methods: precise spatial information
- Choice of approach depends on task
  - Spatial information is probably not needed for category-level recognition
  - Spatial information is useful for tasks that require matching structures across images (next lectures!)