Robert Collins
CSE586

Feature Extraction
Summary/Overview

Reading: Chapter 13.1-13.3, Prince book

Quote p.323, Prince Book

We define preprocessing to be any transformation of the pixel data prior to building the model that relates the data to the world. Such transformations are often ad-hoc heuristics: their parameters are not learned from training data, but they are chosen based on experience of what works well.

In a sense, the need for preprocessing represents a failure; we are admitting that we cannot directly model the relationship between the RGB values and the world state.

- **Comment 1: Ouch!**
- **Comment 2: See "deep learning" if you want to learn the features too.**

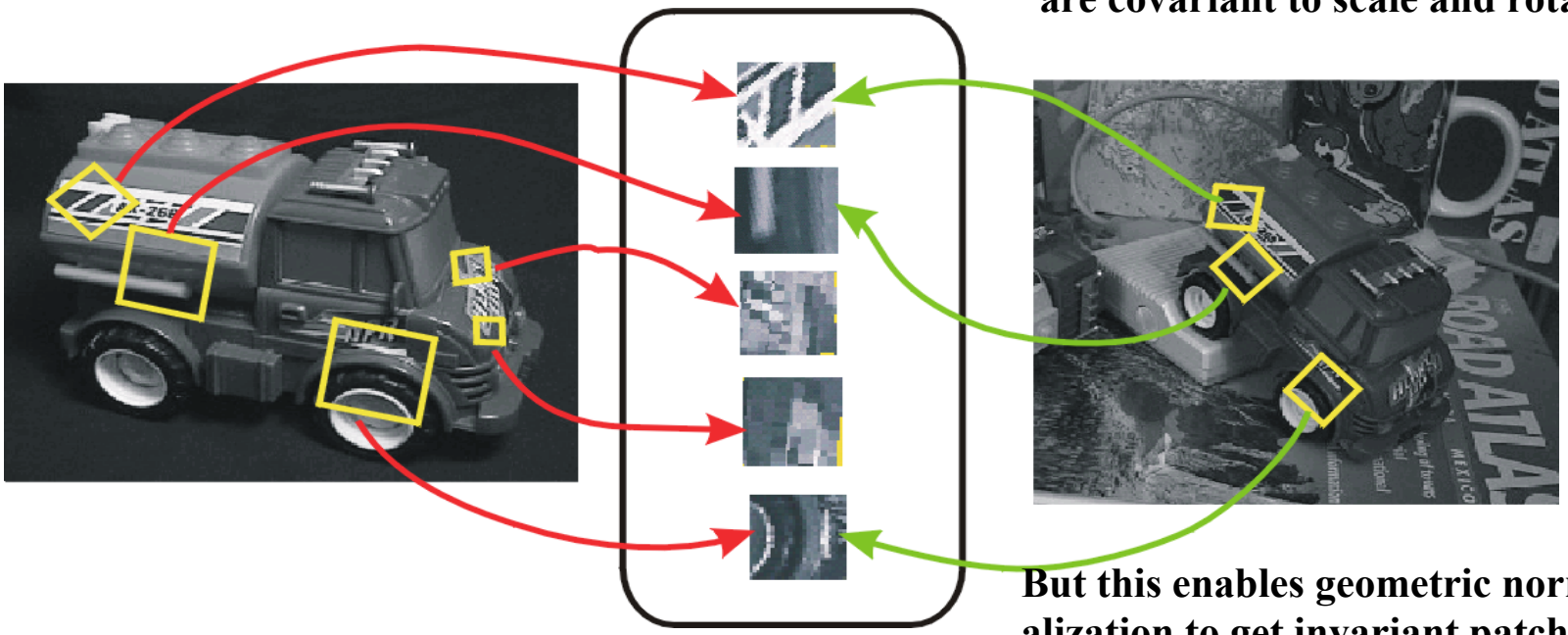
Concepts to Keep in Mind

- Dense vs sparse
  - dense: compute descriptor at each pixel (or on a grid)
    (example: convolution with a linear operator)
  - sparse: detect "good" places to compute descriptors
    (example: corner detection)
- Invariant vs covariant
  - Invariant: value does not change after transformation
    (example: normalized correlation is invariant to gain and offset)
  - Covariant: value changes reliably with some parameter of the transformation
    (example: area of a square scales by 4 when image is magnified by 2)

Invariance vs. covariance

- **Invariance:**
  - features(transform(image)) = features(image)
- **Covariance:**

e.g. SIFT detections (x,y,theta,scale) are covariant to scale and rotation



But this enables geometric normalization to get invariant patches.

Covariant detection => invariant description

S.Lazebnik, UNC

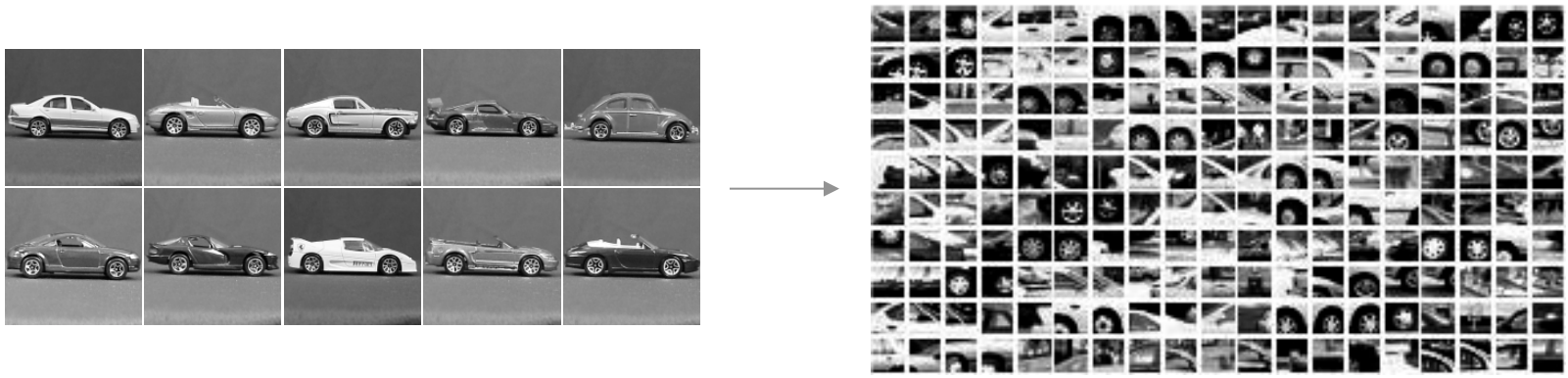
Invariance, Covariance

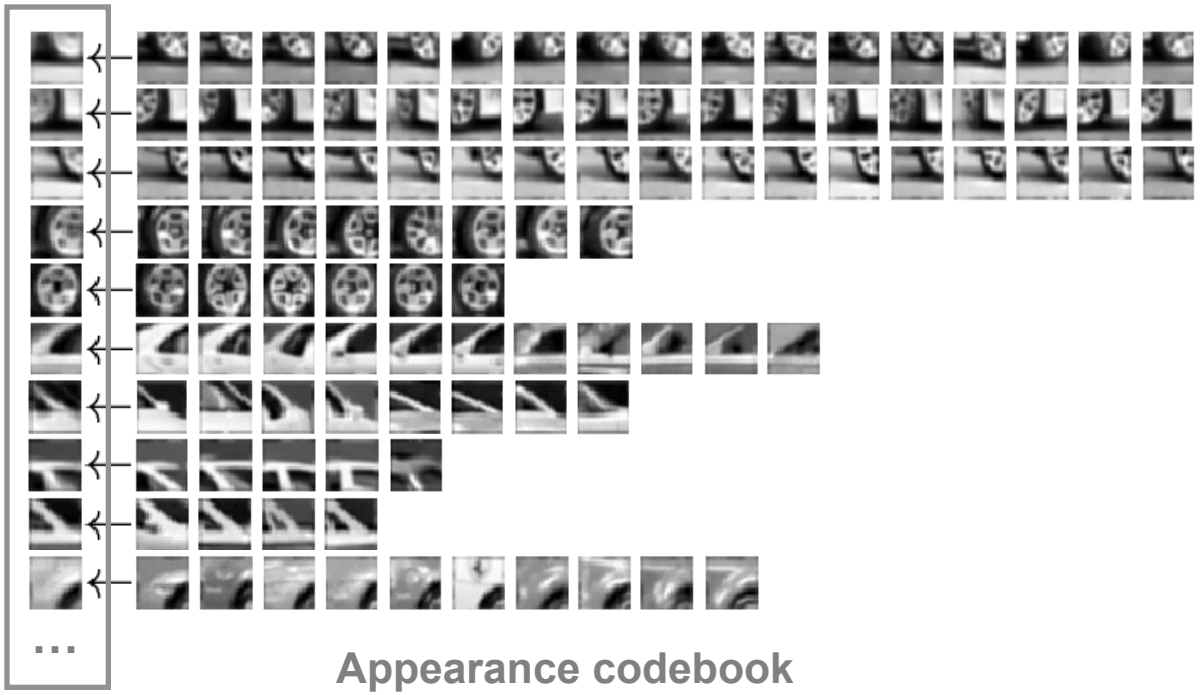
- Rules of thumb
  - want to reduce unwanted variation in image due to lighting, scale, deformation etc.
  - want detector response values/scores to be invariant to changes in lighting and viewpoint
  - want feature descriptor vectors to be invariant so we can match them easily
  - want feature location/orientation/scale to be covariant with geometric transformations
    - e.g. if object shifts in the image we want it's detected location to shift along with it

Outline

- Per-pixel processing
- Edges, corners, blobs/regions
- Feature descriptors

Motivation: Using Image Patches as Descriptors



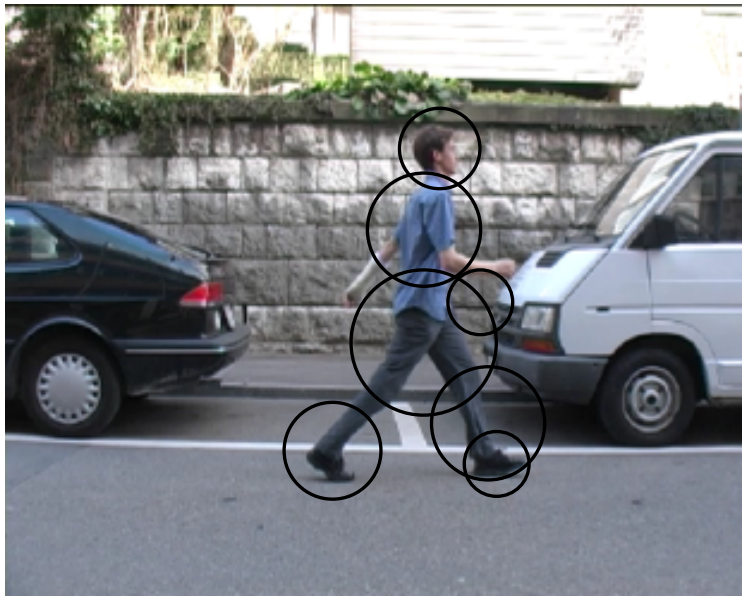


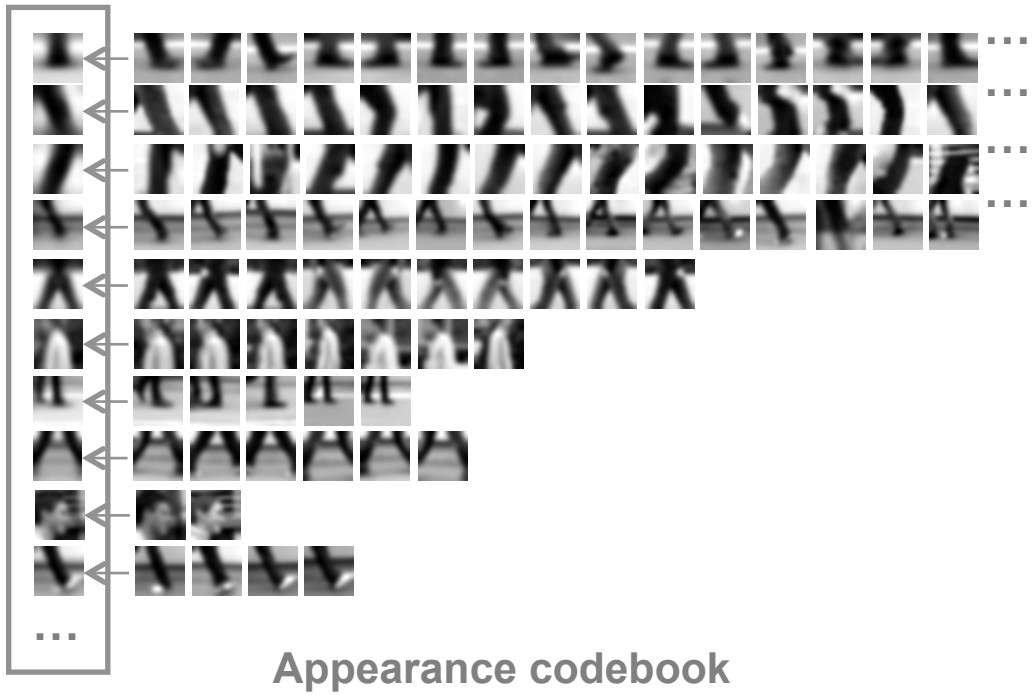
Source: B. Leibe

Robert Collins
CSE586

Motivation: Using Image Patches as Descriptors

Note: each patch is same size, say RxC pixels. It can therefore be represented by a point vector in R*C-dimensional space.





Appearance codebook

Source: B. Leibe

Intensity Normalization

- We would like to normalize patches to not be sensitive to changes in illumination
- Fix first and second moments to standard values
  e.g. mean=0, std=1
- Removes contrast and constant additive luminance variations

Before

After

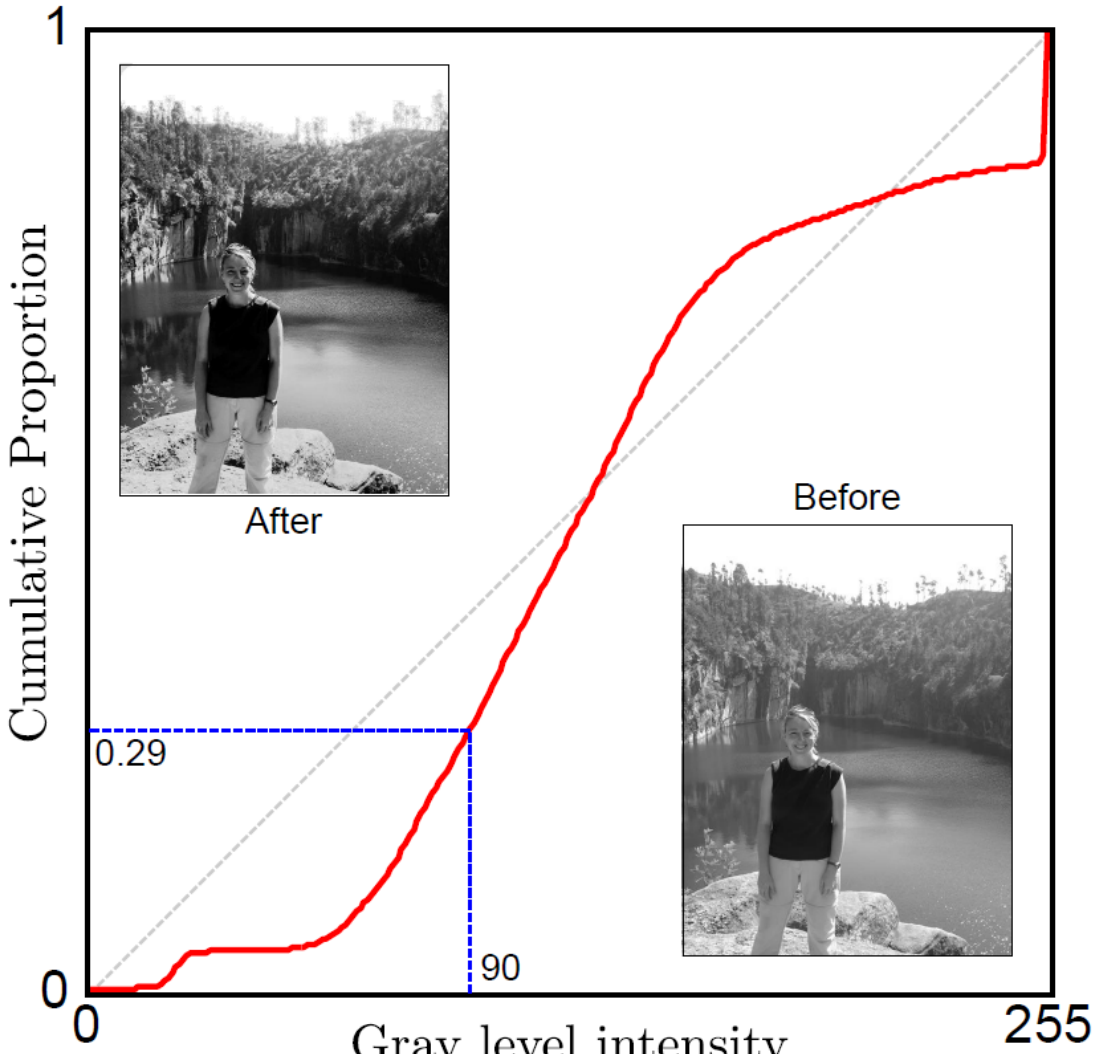
Histogram Equalization

Make all of the moments the same by forcing the histogram of intensities to be the same

Before/ normalized/ Histogram Equalized

Histogram Equalization

recall that using the cumulative distribution function as an intensity transformation s=T(r) yields a new image with a more uniform intensity distribution.



Filter Responses

Convolution **P*F** of patch **P** and filter **F**

$$x_{ij} = \sum_{m=-M}^{M} \sum_{n=-N}^{N} p_{i-m,j-n} f_{m,n}$$

- Computes weighted sum of pixel values with weights specified by filter coefficients $f_{m,n}$
- Convolution is a linear operator
- Filter response is shift invariant
- Location of filter response is shift covariant

Blurring (convolve with Gaussian)

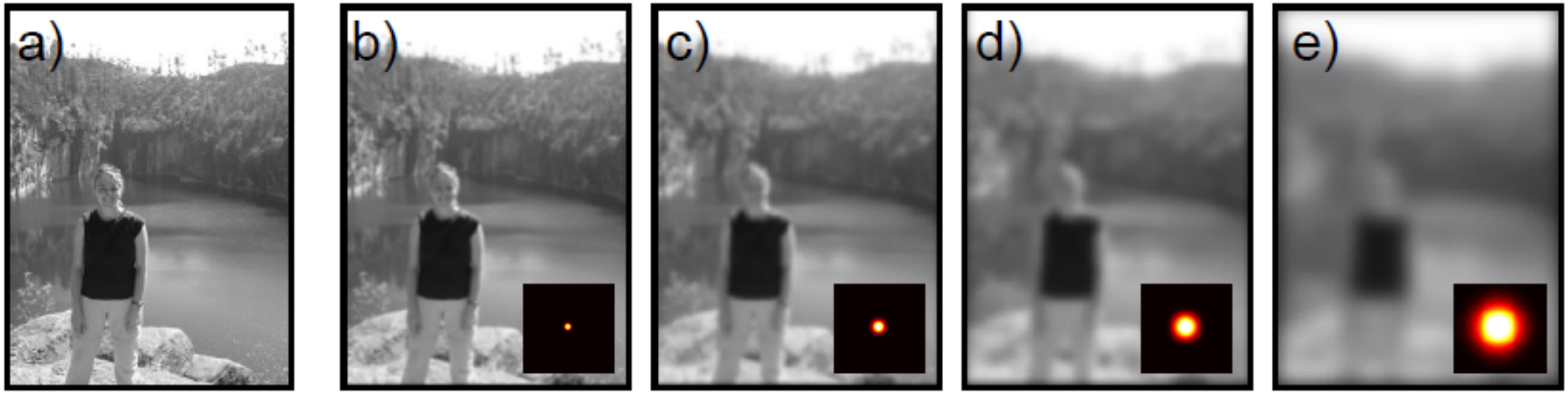
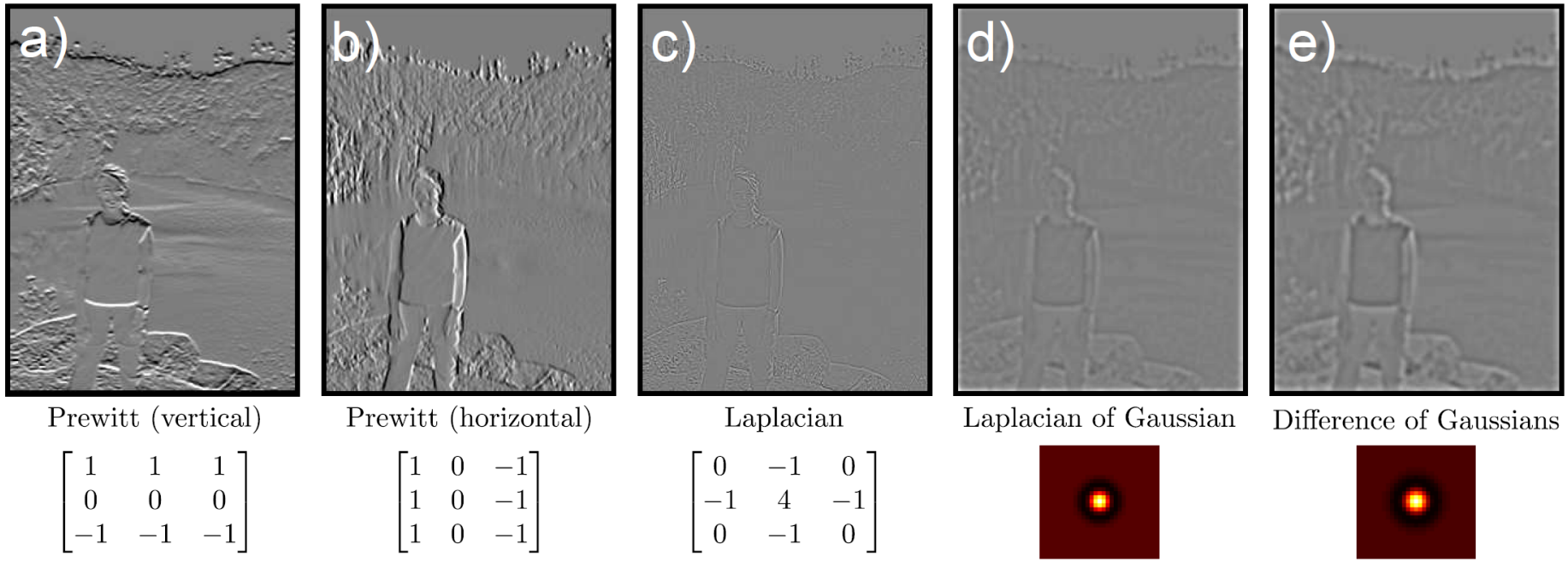


Figure B.3 Image blurring. a) Original image. b) Result of convolving with a Gaussian filter (filter shown in bottom right of image). The image is slightly blurred. c-e) Convolving with a filter of increasing standard deviation causes the resulting image to be increasingly blurred.

Zero-Mean Filters

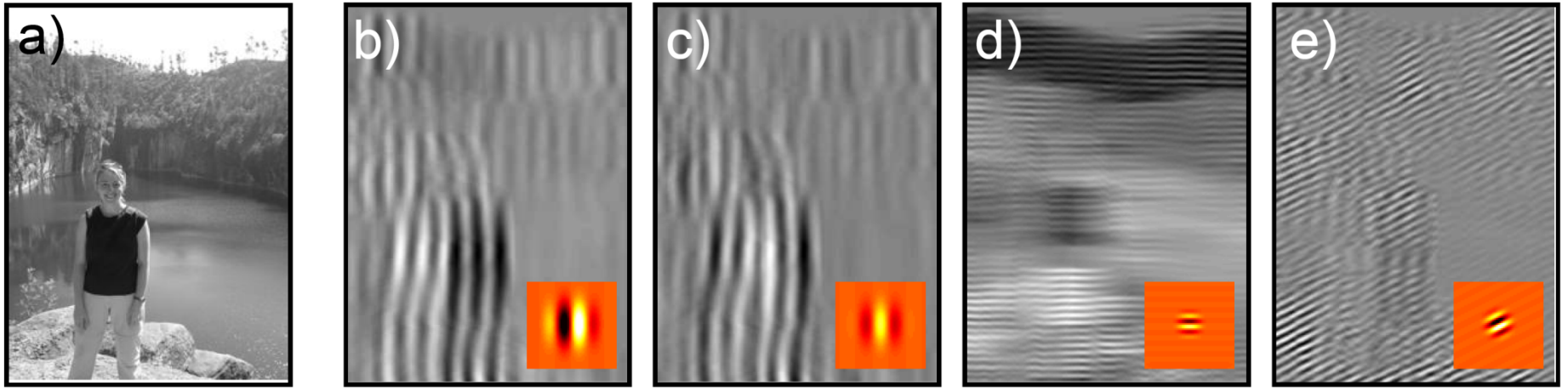
- Zero-mean filters (sum of filter coefficients is 0) are invariant to additive brightness change
  - F*(I+b) = F*I + b F*(ones(size(I))) = F*I + 0 = F*I
- Zero-mean filters are covariant to contrast changes
  - F*(aI) = a F*I
- Examples: derivative filters; Laplacian

Gradient Filters



- Rule of thumb: big response when image matches filter

Gabor Filters

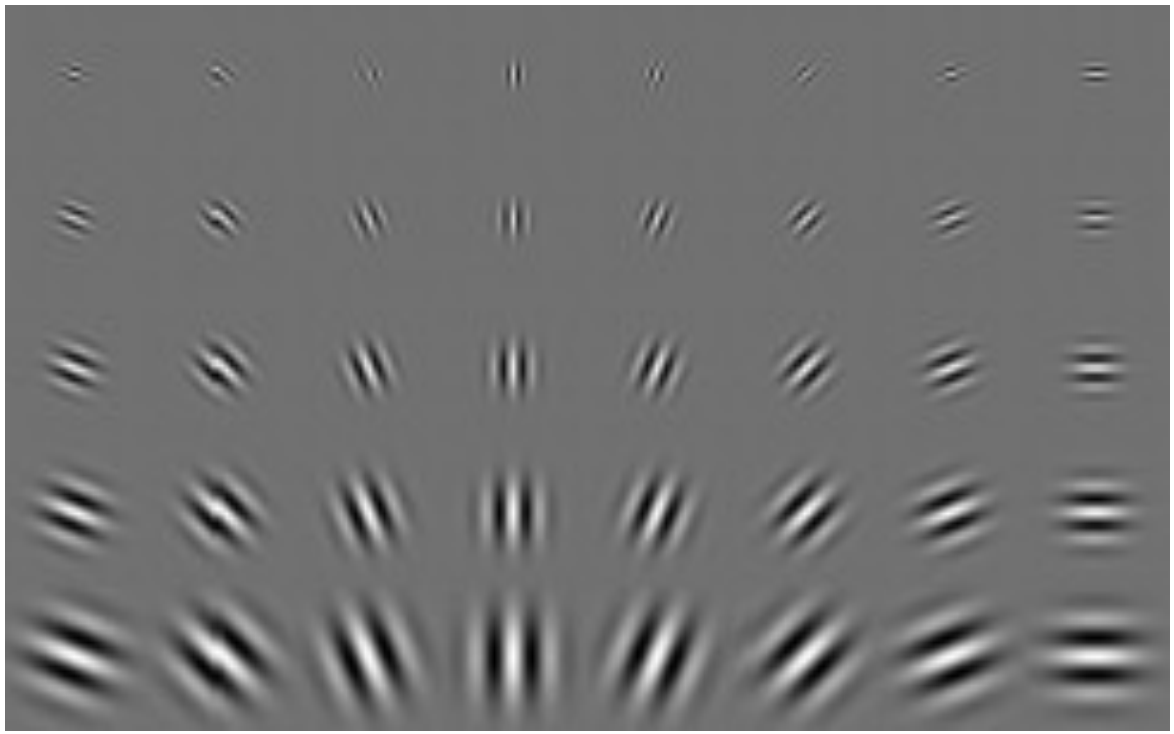


$$f_{mn} = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{m^2+n^2}{2\sigma^2}\right] \sin\left[\frac{2\pi(\cos[\omega]m + \sin[\omega]n)}{\lambda} + \phi\right]$$

Wavelets at different orientations and scales

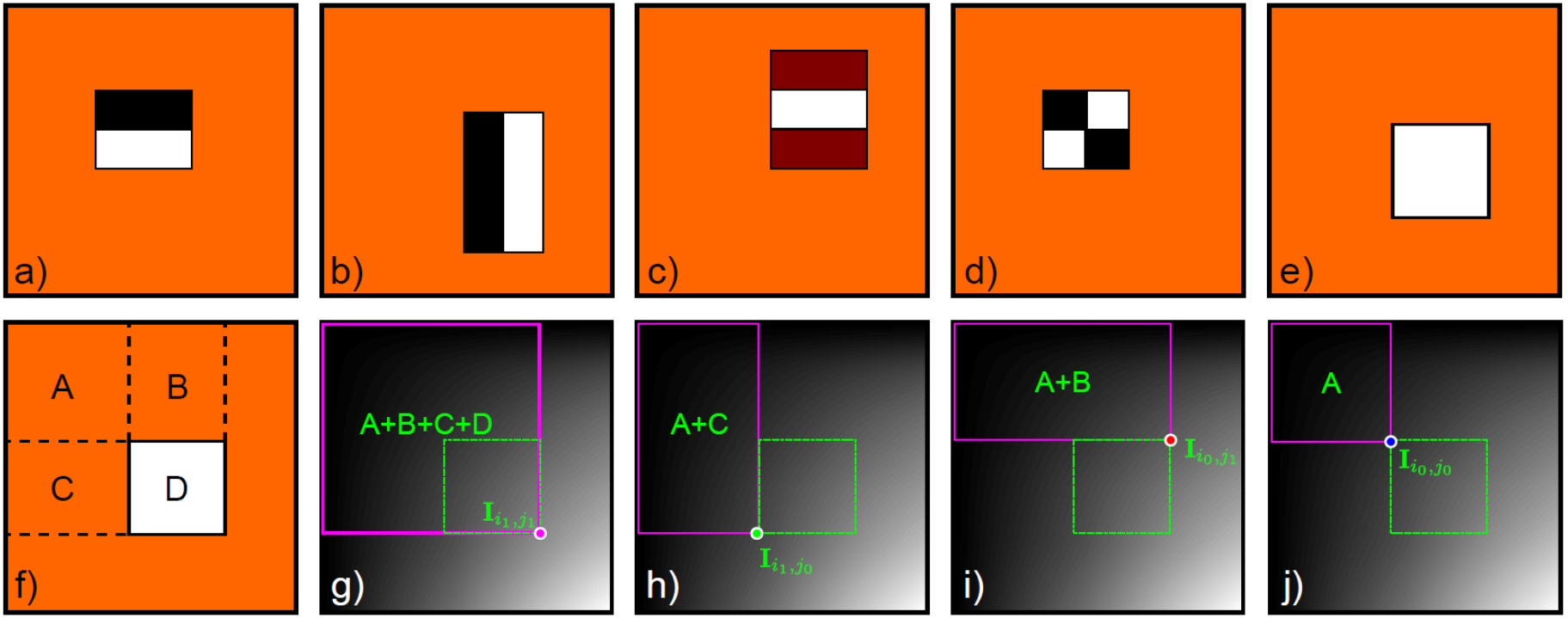
Gabor Jets (Descriptor)

- Concatenate Gabor filter responses at several scales and orientations in a feature vector (jet)



This is computing a vector of responses at each pixel.

Haar Filters

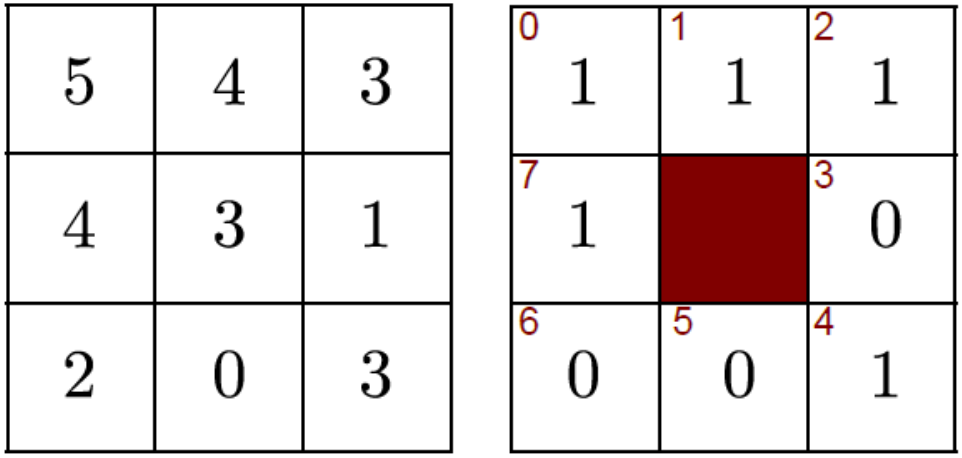


- Another kind of wavelet
- Can be computed efficiently using "integral images" that quickly computing sums over rectangular image patches

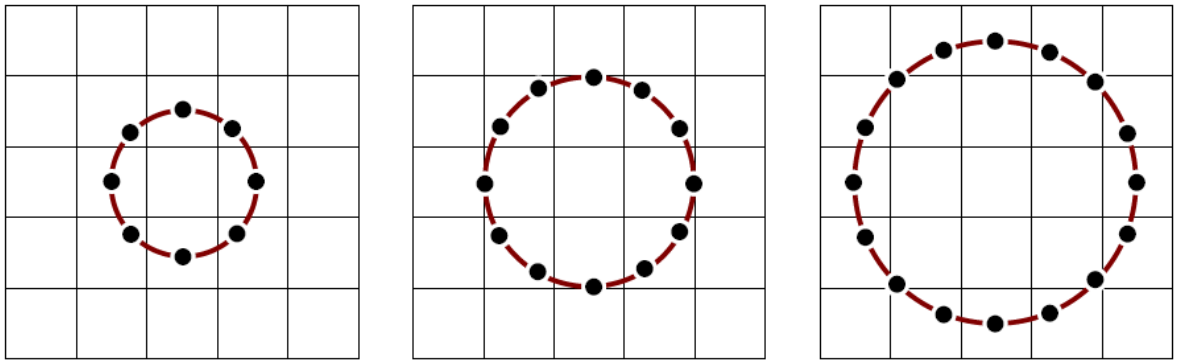
Local binary patterns

Nonlinear (cannot be computed by convolution)

insensitive to illumination changes



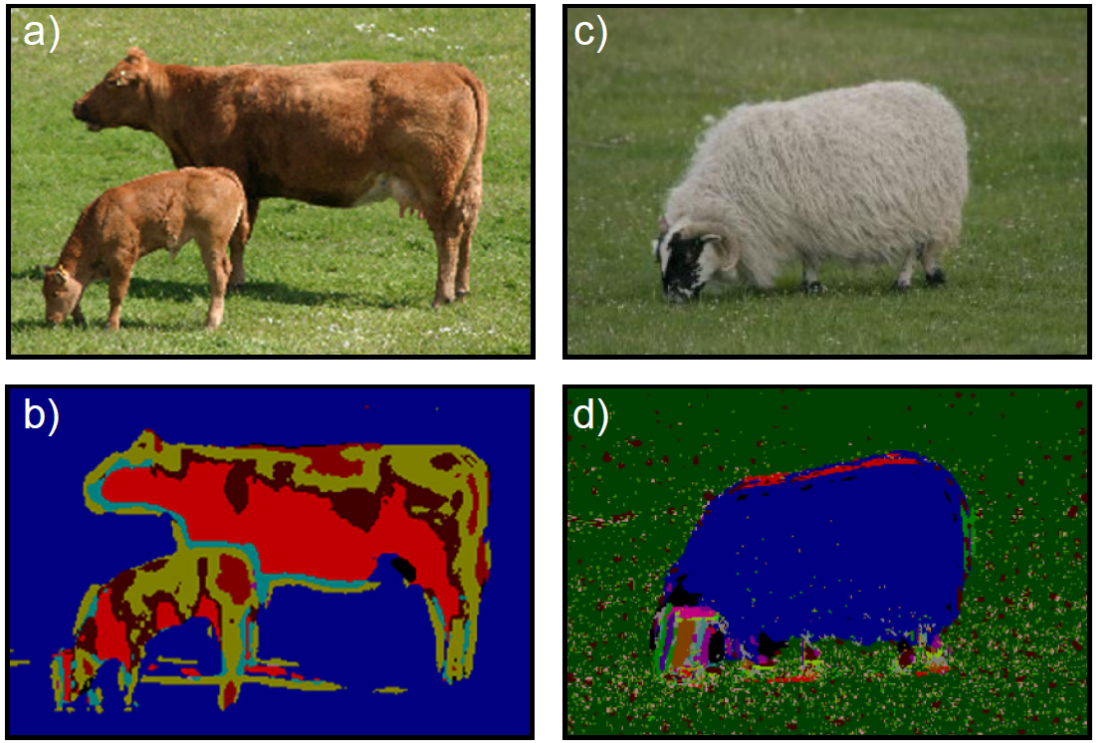
$$\text{LBP} = 10010111 = 151$$



$(P = 8, R = 1.0)$ $(P = 12, R = 1.5)$ $(P = 16, R = 2.0)$

Textons

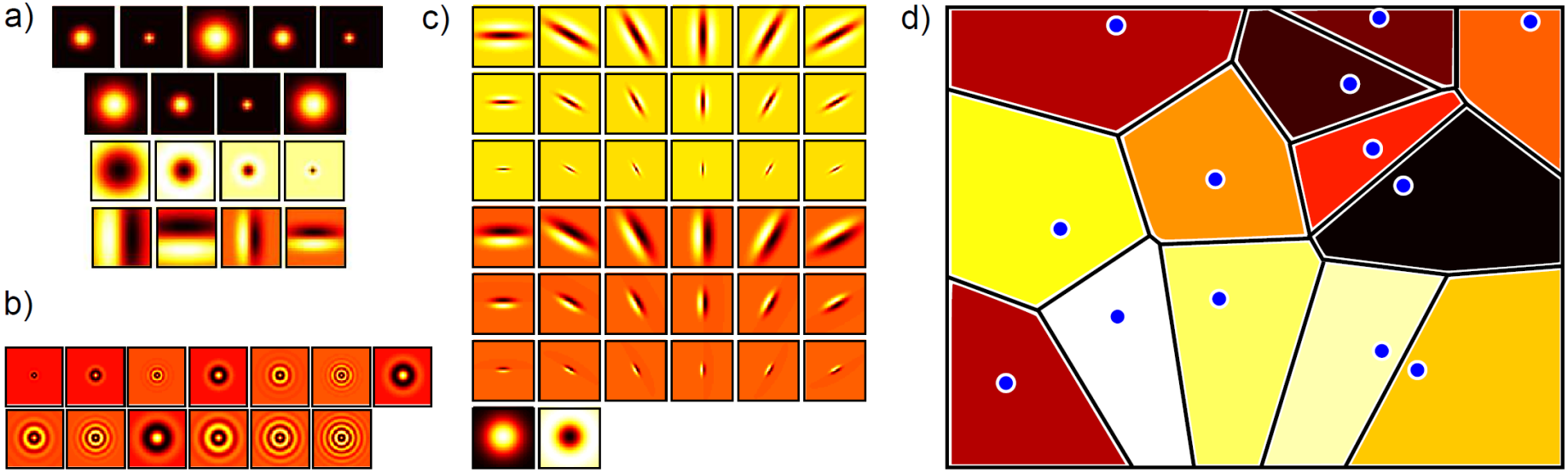
- An attempt to characterize texture
- Replace each pixel with integer representing the texture "type"
- Categorical data (suitable for modeling with categorical distribution)



Computing Textons

Take a bank of filters and apply to lots of images

Cluster in filter space



For new pixel, filter surrounding region with same filter bank, and assign to nearest cluster

Outline

- Per-pixel processing
- Edges, corners, blobs/regions
- Feature descriptors

Edges

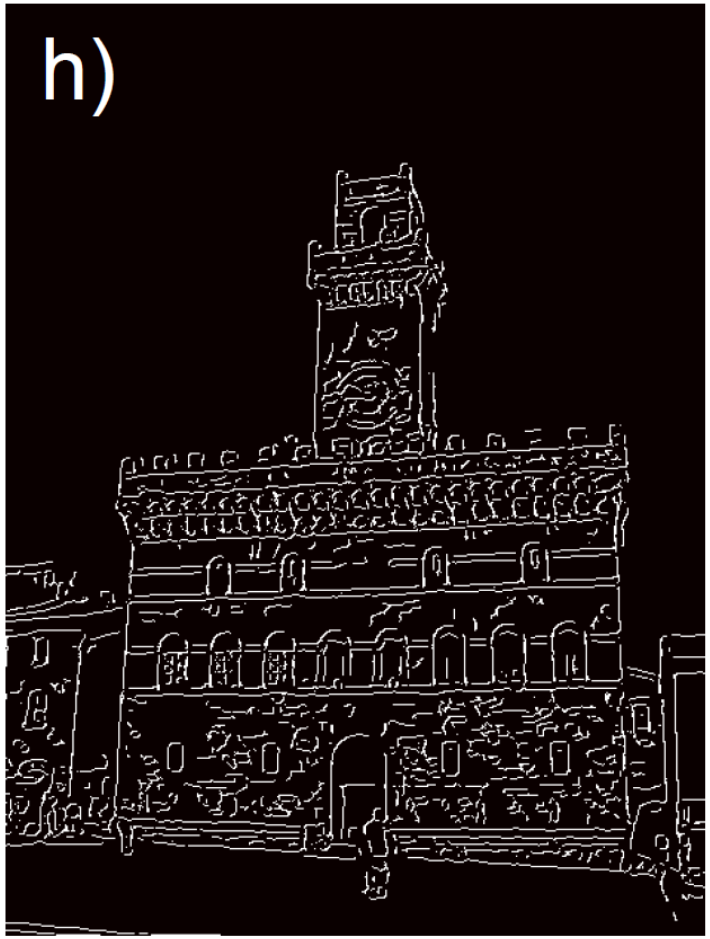
Original image

Reconstructed from edges and contrast info

(from Elder and Goldberg 2000)

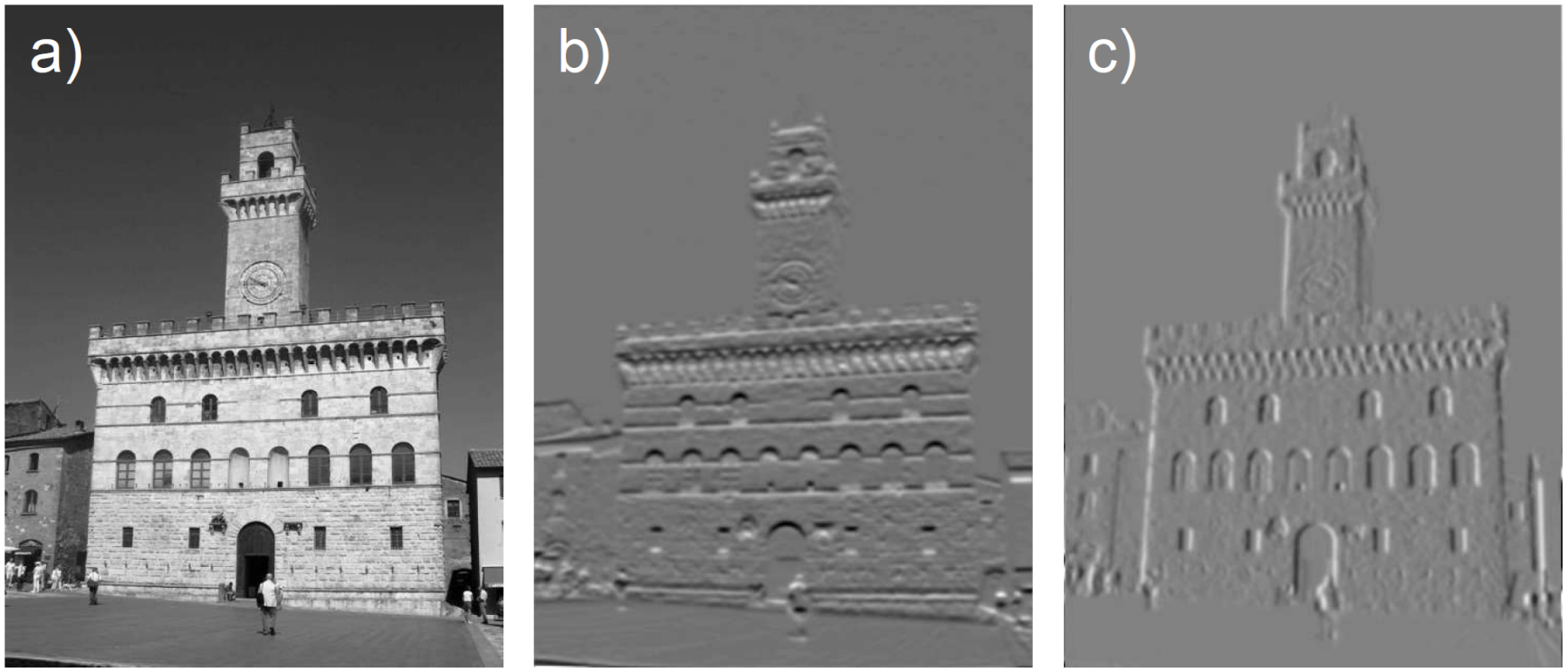
Canny Edge Detector

J. Canny *A Computational Approach to Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 8, No.6, Nov 1986



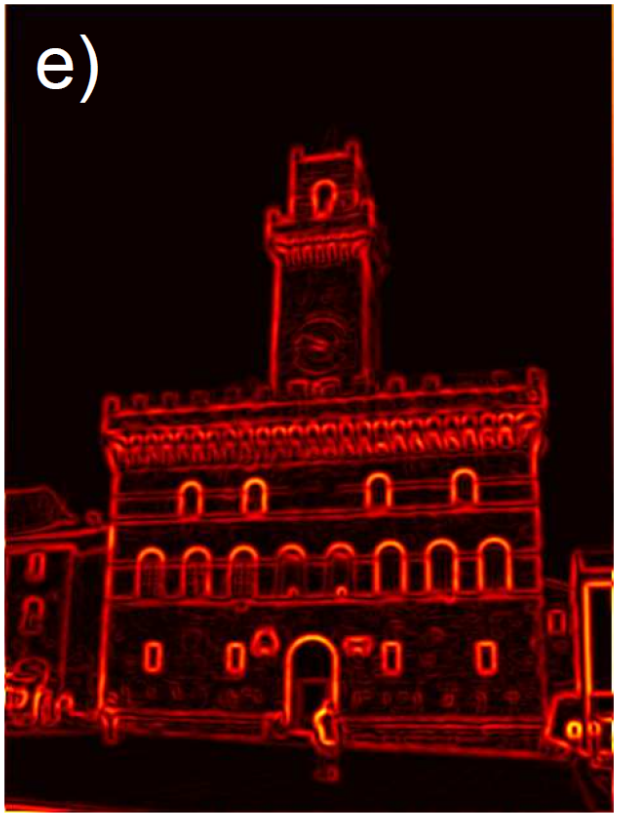
Canny Edge Detector

Compute horizontal and vertical gradient images **h** and **v**



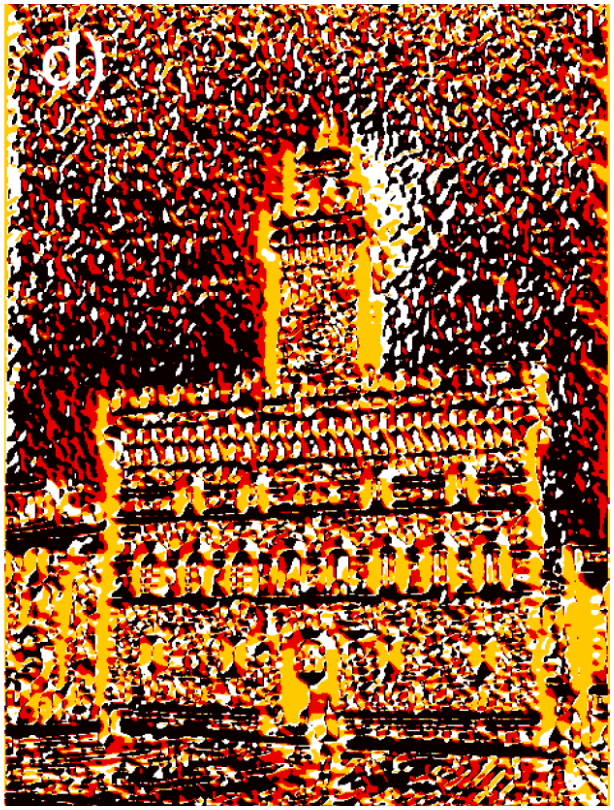
Canny Edge Detector

magnitude



$$a_{ij} = \sqrt{h_{ij}^2 + v_{ij}^2}$$

orientation

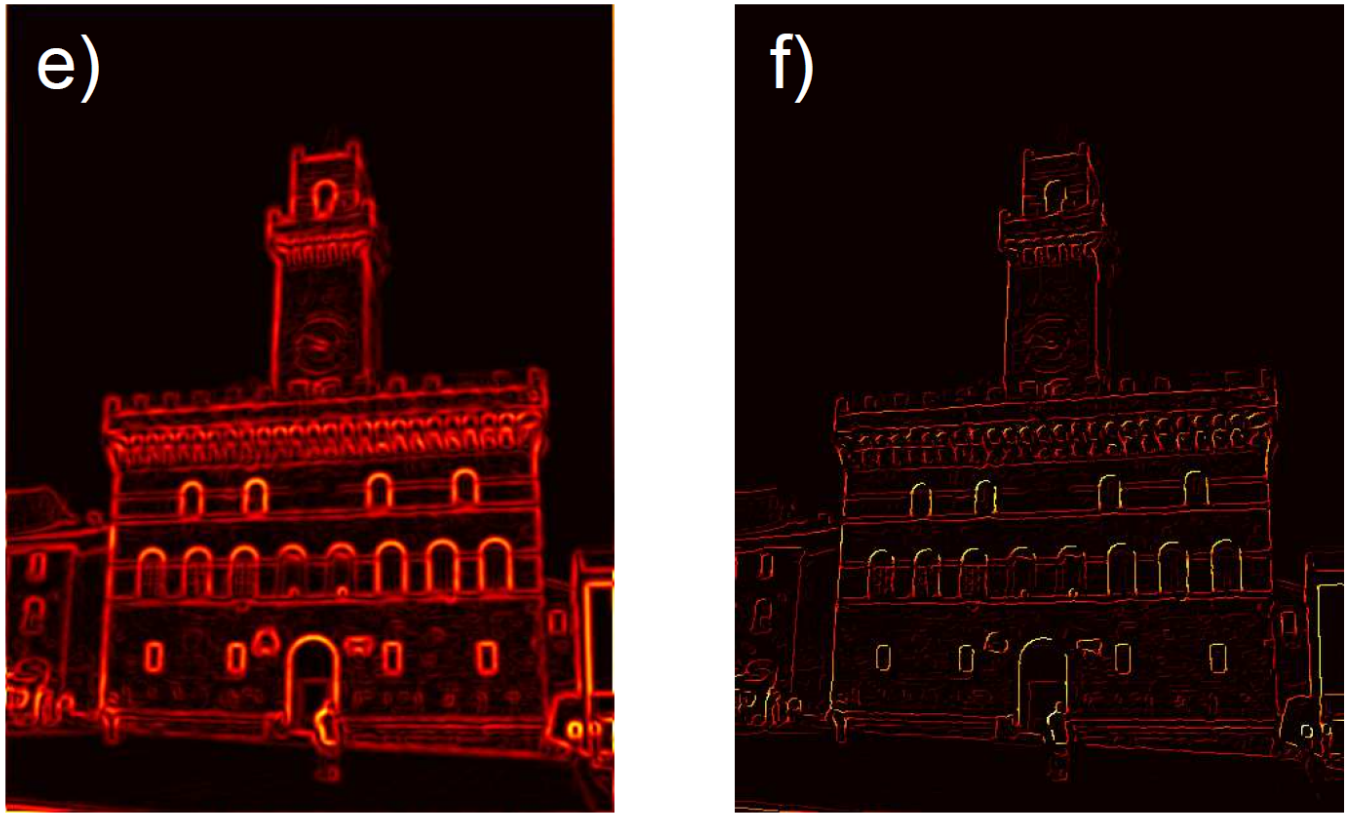


$$\boldsymbol{\theta}_{ij} = \arctan[v_{ij}/h_{ij}]$$

Quantize to 4 directions

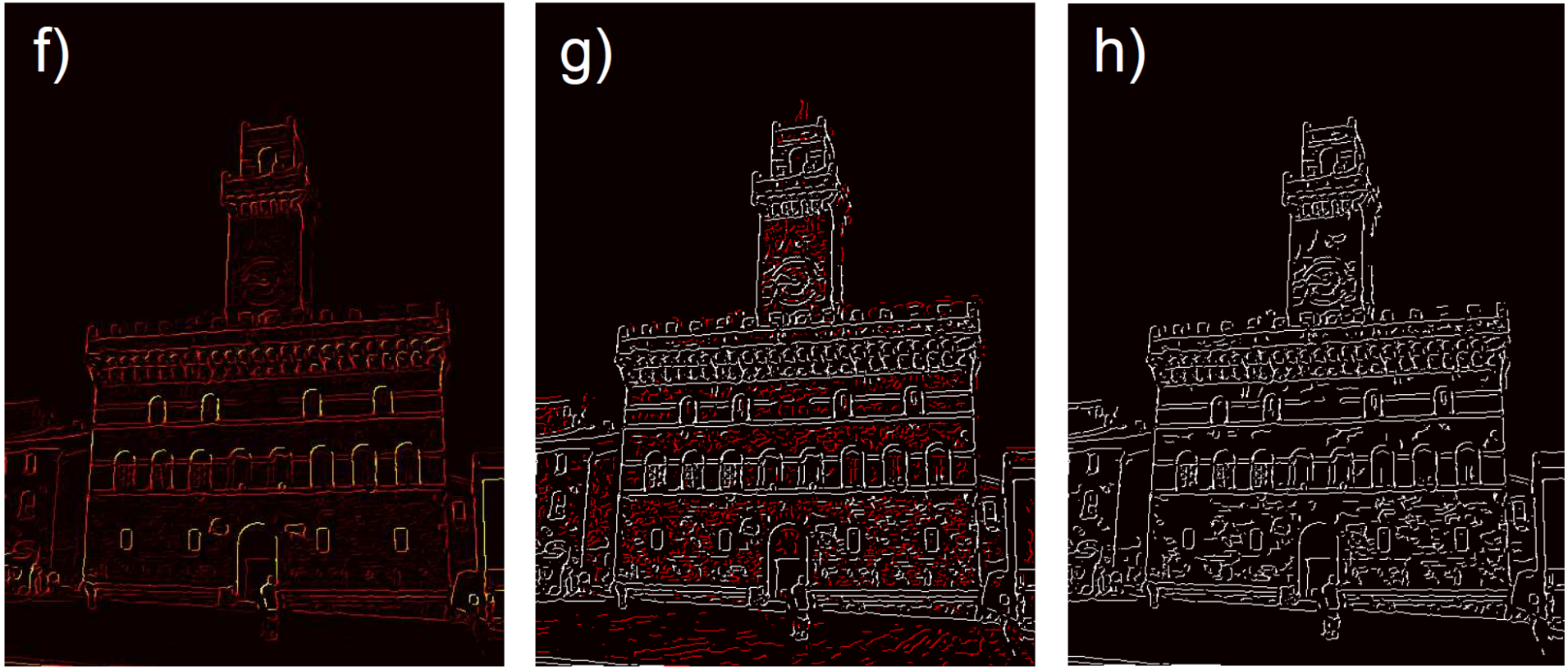
Canny Edge Detector

Non-maximal suppression

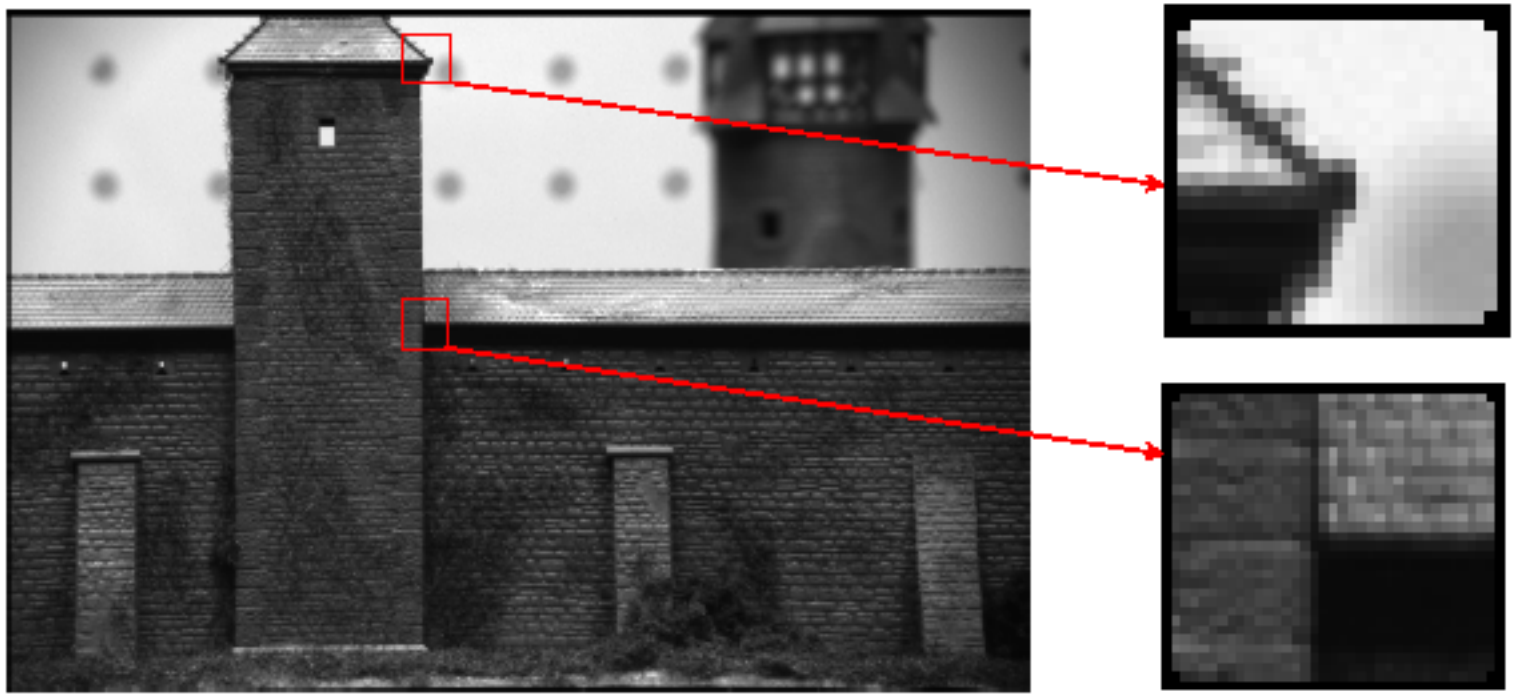


Canny Edge Detector

Edge Linking with Hysteresis Thresholding



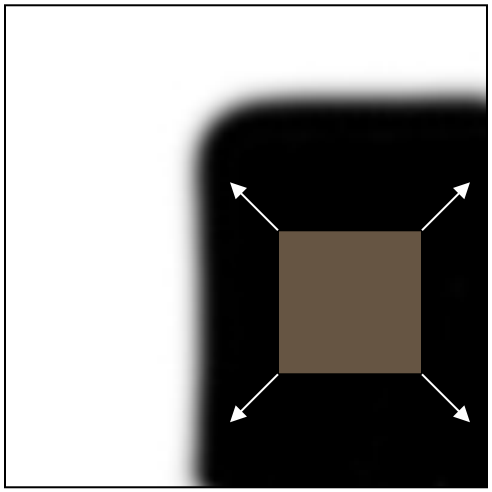
Corners



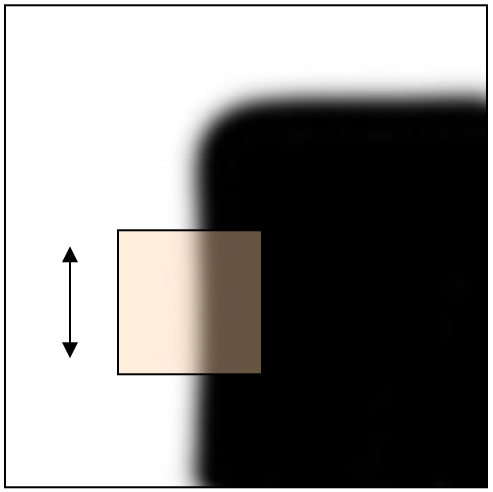
- Intuitively, junctions of contours.
- Generally more stable features over changes of viewpoint
- Intuitively, large variations in the neighborhood of the point in all directions
- They are good features to match!

Corner Detection: Basic Idea

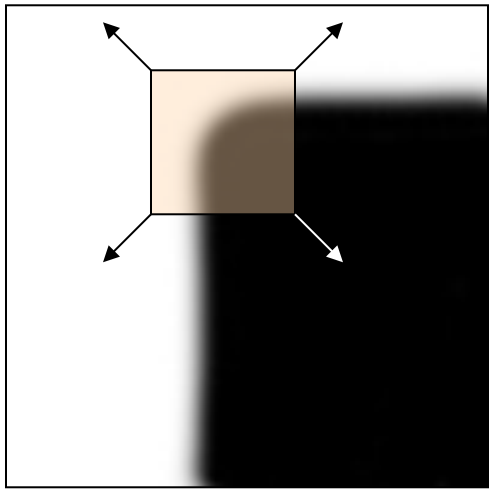
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in all directions



"edge":
no change along the edge direction



"corner":
significant change in all directions

Source: A. Efros

Harris Corner Matrix

$$\left(\sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right)$$

A real, symmetric matrix

=> has real eigenvalues

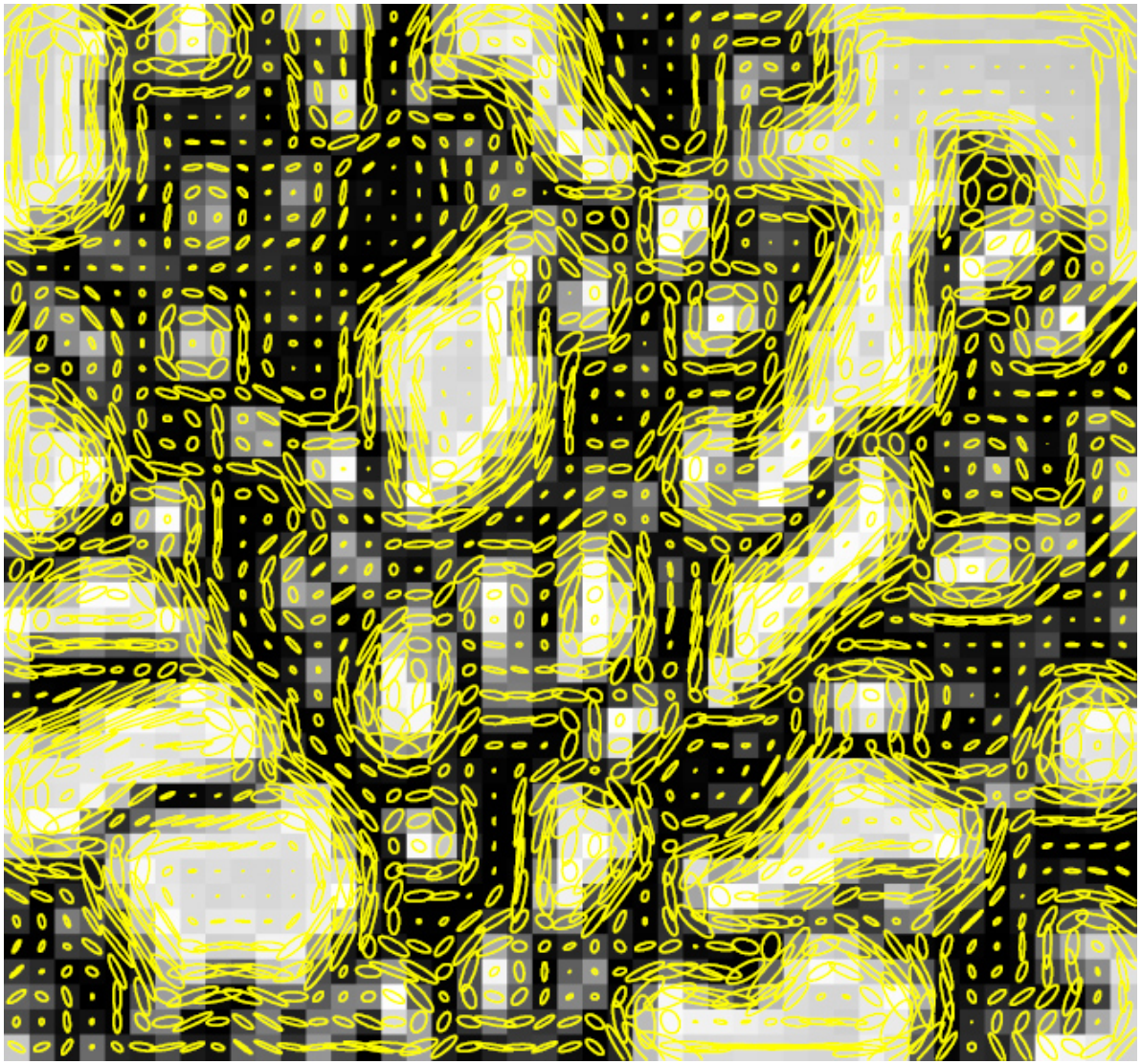
=> can be decomposed as R D R^t

=> describes the shape of an ellipse!

Visualization of second moment matrices

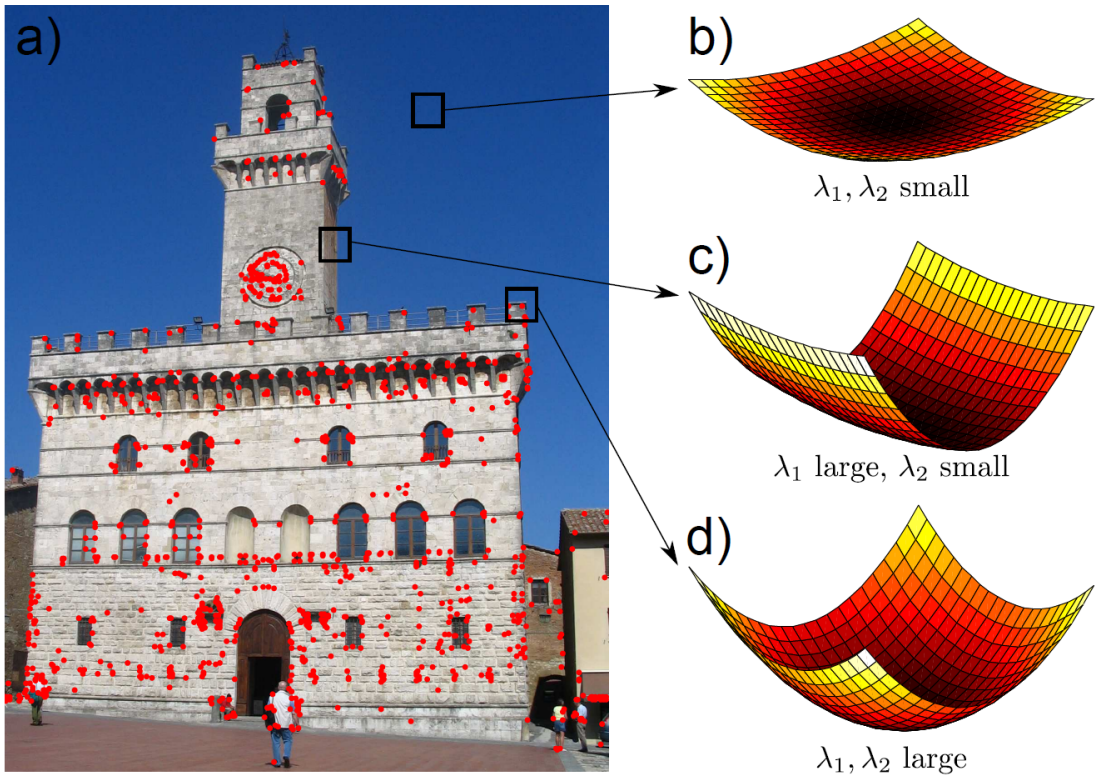
S.Lazebnik, UNC

Visualization of second moment matrices



S.Lazebnik, UNC

Harris Corner Detector



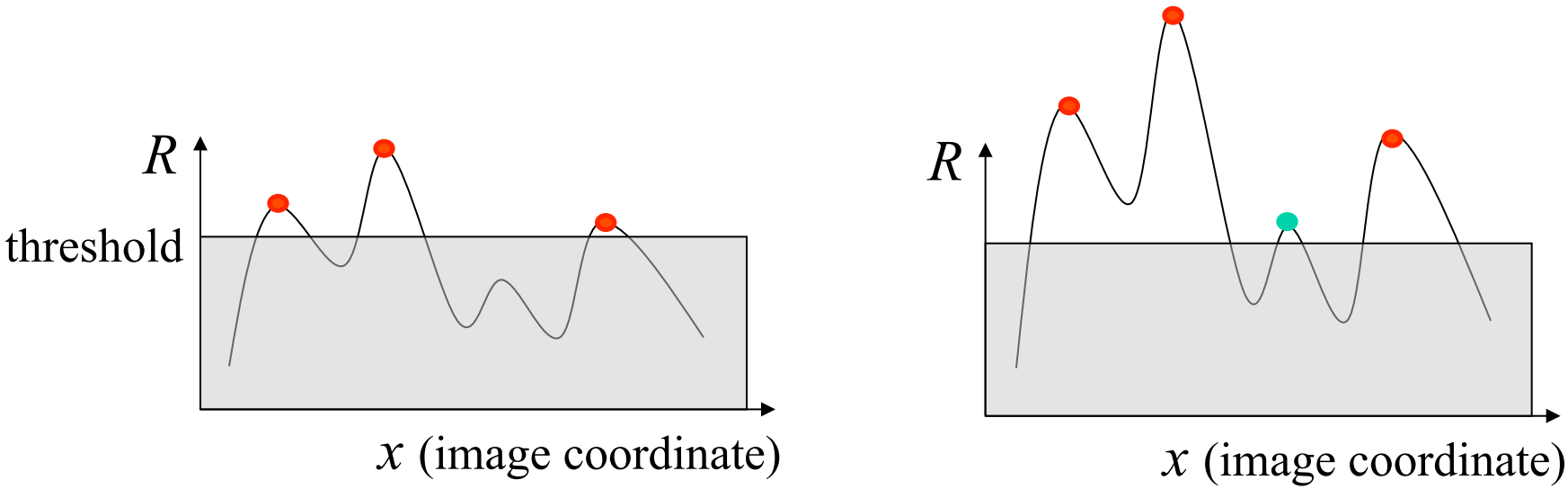
Make decision based on image structure tensor

$$\mathbf{S}_{ij} = \sum_{m=i-D}^{i+D} \sum_{n=j-D}^{i+D} w_{mn} \begin{bmatrix} h_{ij}^2 & h_{ij}v_{ij} \\ h_{ij}v_{ij} & v_{ij}^2 \end{bmatrix}$$

Invariance to intensity change?

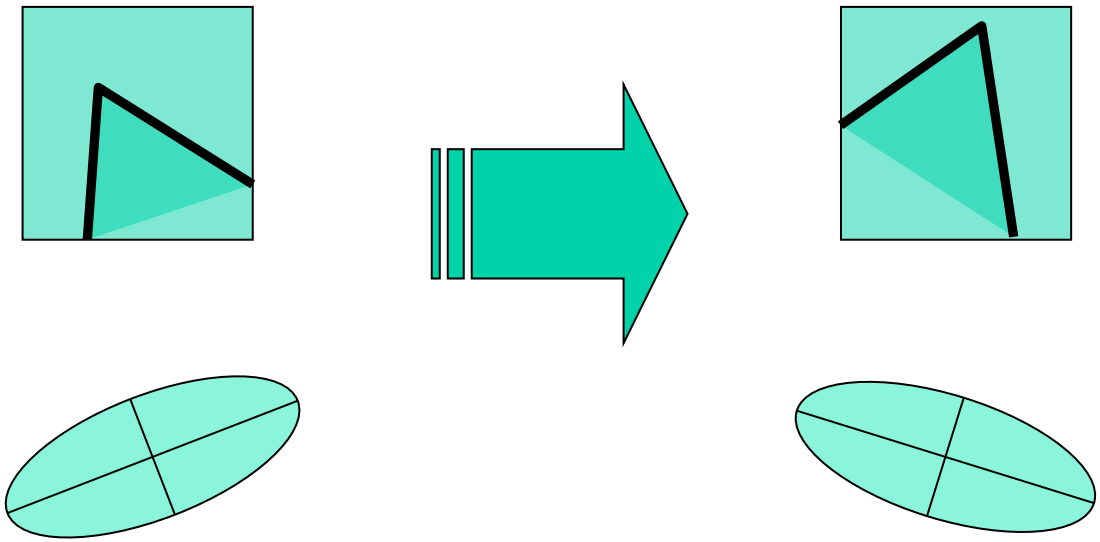
✓ Only derivatives used => invariant to intensity shift $I \rightarrow I + b$

-- However, what about change in contrast: $I \rightarrow a\, I$?



Partially invariant to affine intensity change

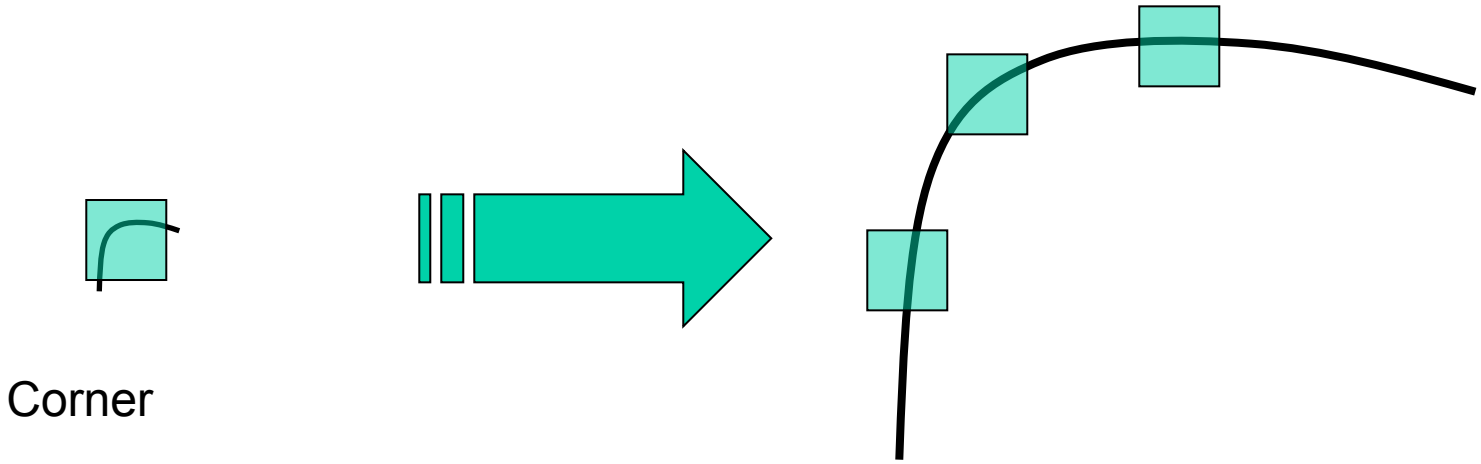
Invariance to Image rotation?



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

Corner response R is invariant w.r.t. rotation and *corner location and orientation* are covariant

Invariance to Scaling?



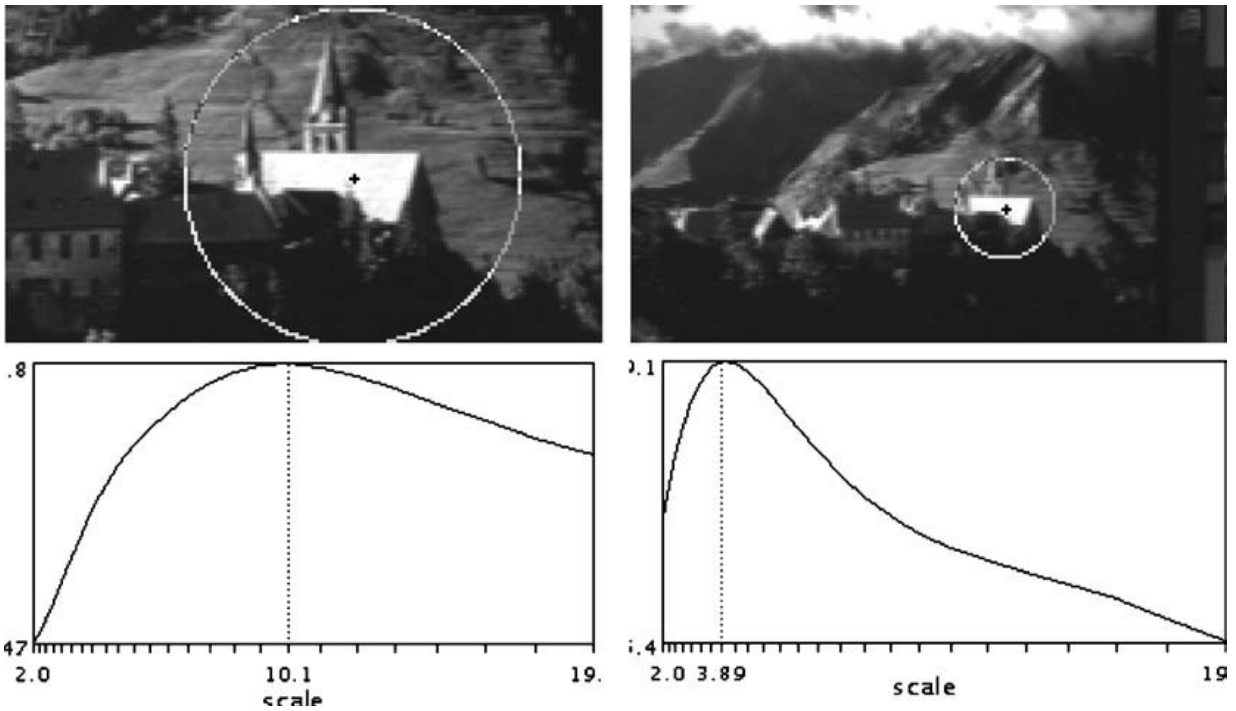
Corner

All points will be classified as edges

Not invariant to scaling

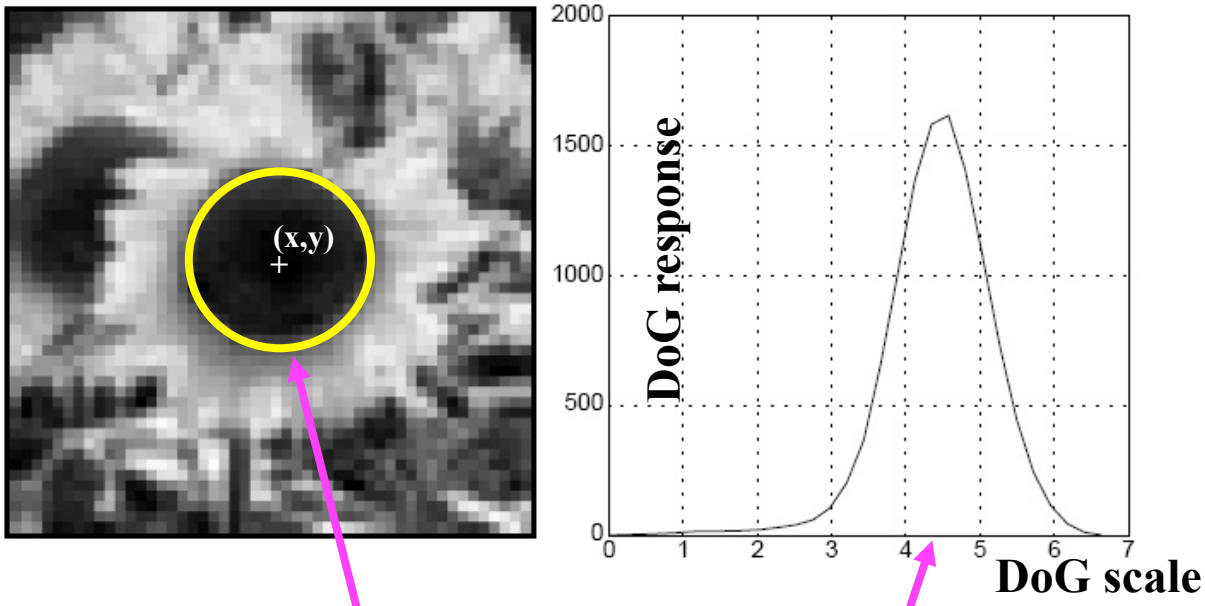
Achieving scale covariance

- Goal: independently detect corresponding regions in scaled versions of the same image
- Need *scale selection* mechanism for finding characteristic region size that is *covariant* with the image transformation



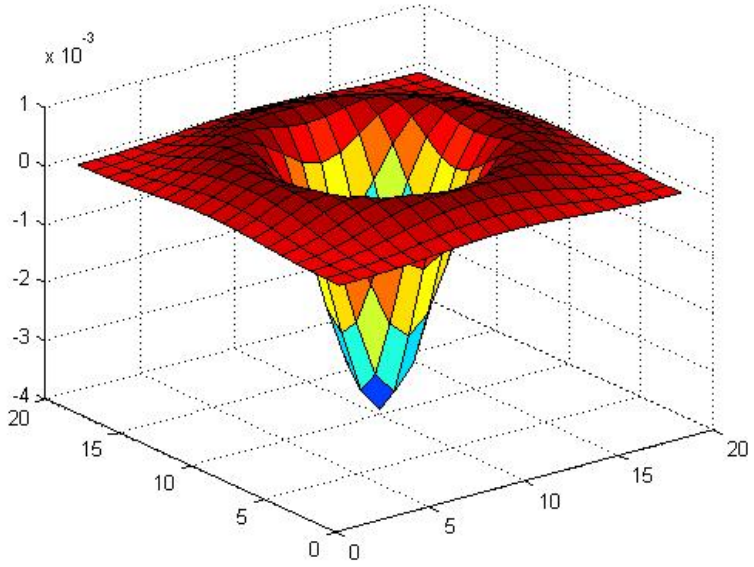
Characteristic Scale of a Blob

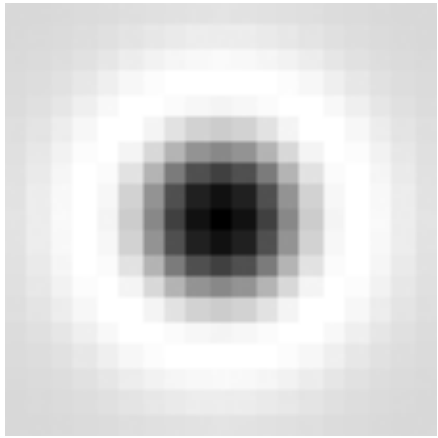
- *Characteristic scale* at a point location is the scale that produces a peak in the Laplacian (or DoG) response.



Blob detection in 2D

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





$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

Efficient implementation

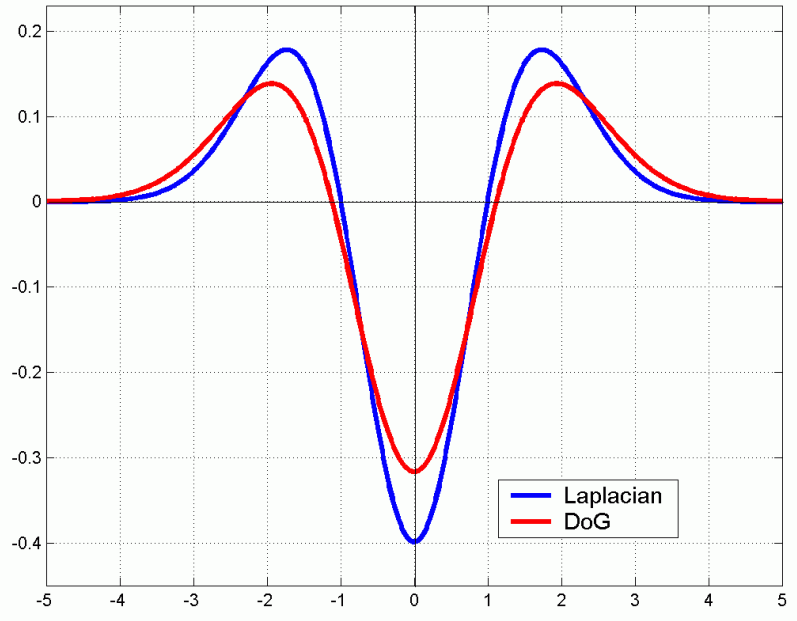
- Approximating the Laplacian with a difference of Gaussians:

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

(Laplacian)

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

(Difference of Gaussians)



Invariance and covariance properties

- Laplacian (blob) response is *invariant* w.r.t. rotation and scaling
- Blob location is *covariant* w.r.t. rotation and scaling

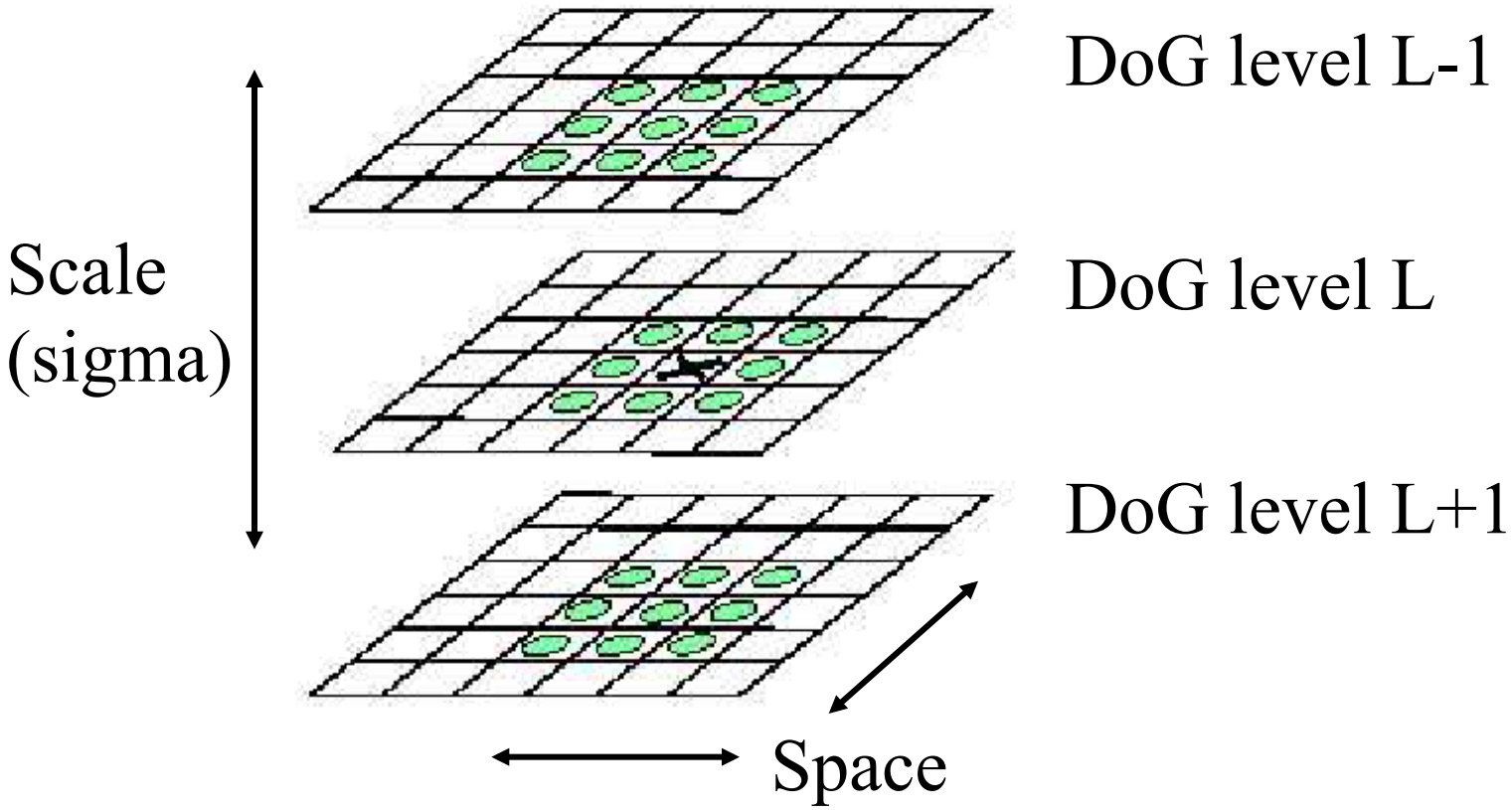
SIFT Keypoints

David G. Lowe, **"Distinctive image features from scale-invariant keypoints,"** *International Journal of Computer Vision,* 60, 2 (2004), pp. 91-110.

Keypoint = (x,y,scale,theta)

Provides an interest point that can be reliably located in a manner covariant to shift, scale, and 2D rotation of image

SIFT Key Points

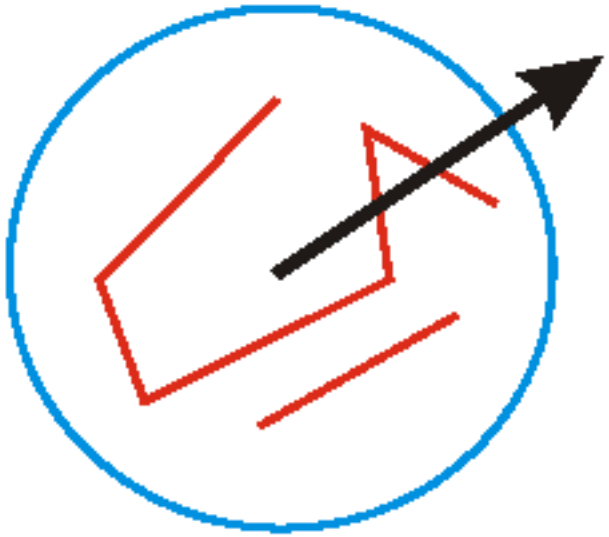


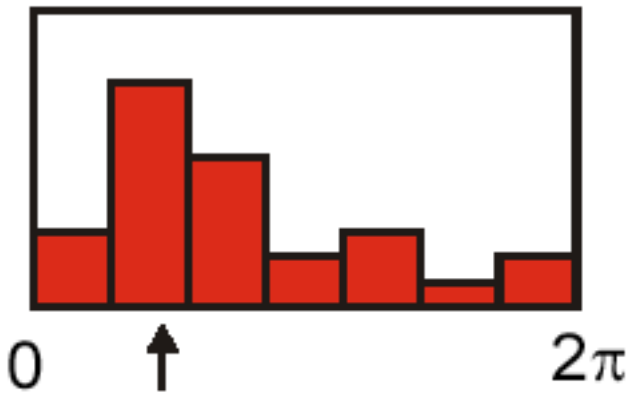
Find maxima and minima in an LoG (or DoG) scale space. The 2D location of the extremal point tells (x,y) **location** of the key point. The scale level of the extremal point tells the canonical **scale** of the image patch surrounding the key point.

Sift Keypoints

Select canonical orientation

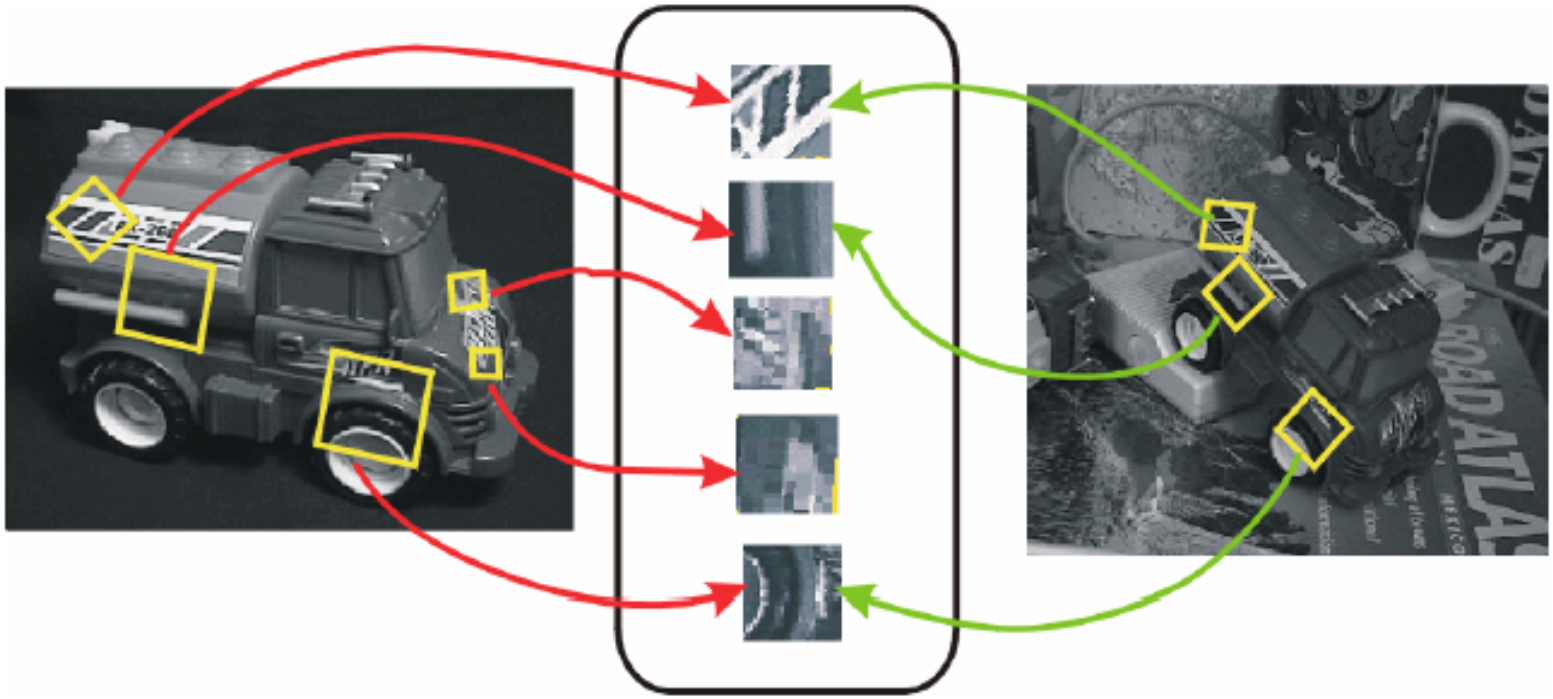
- Create histogram of local gradient directions computed at selected scale
- Assign canonical orientation at peak of smoothed histogram
- Each key specifies stable 2D coordinates (x, y, scale, orientation)





SIFT Keypoints

Intensity patches around detected keypoints can now be normalized to be invariant to translation, scale and rotation (a similarity transformation)



Normalized patches

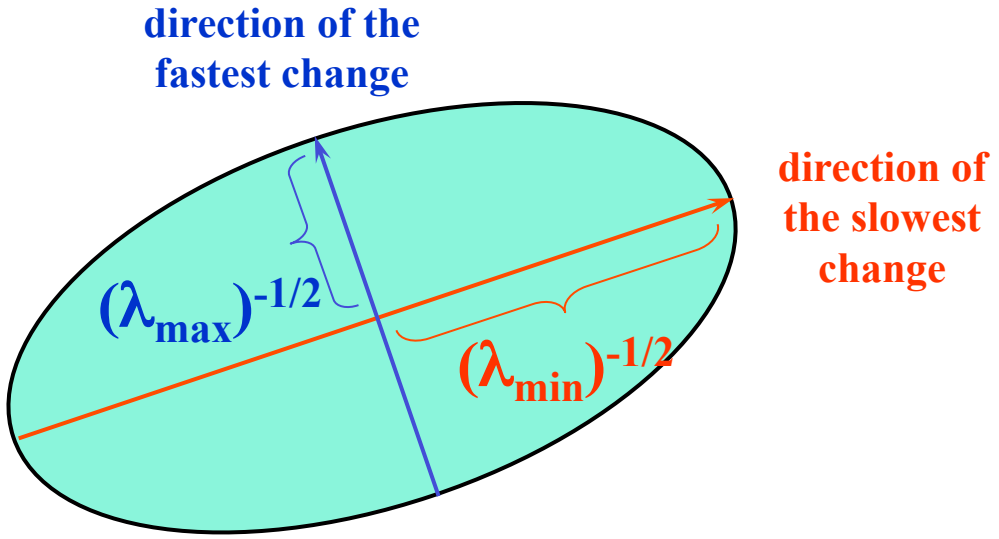
Achieving affine covariance

Consider the second moment matrix of the window containing the blob (note similarity to Harris corner matrix)

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

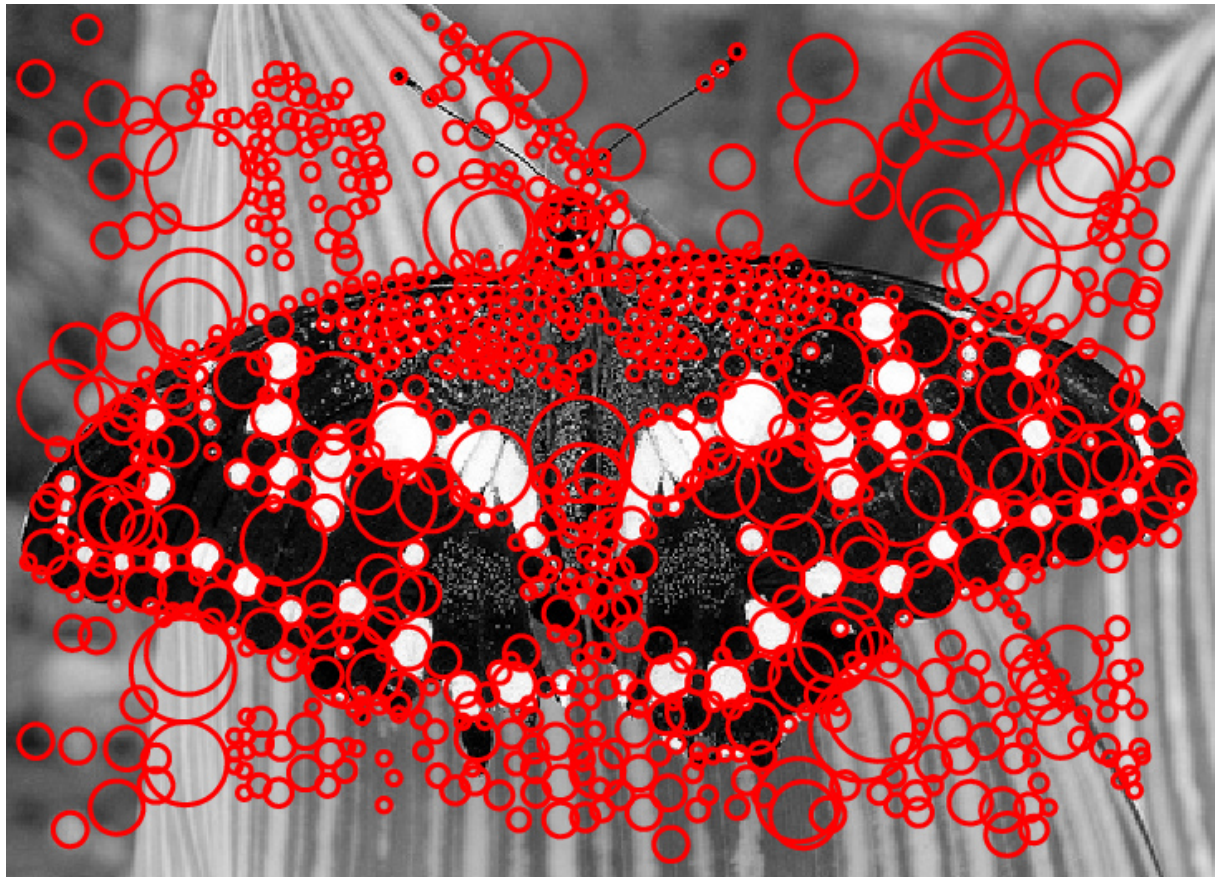
Recall:

$$[u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$



This ellipse visualizes the "characteristic shape" of the window

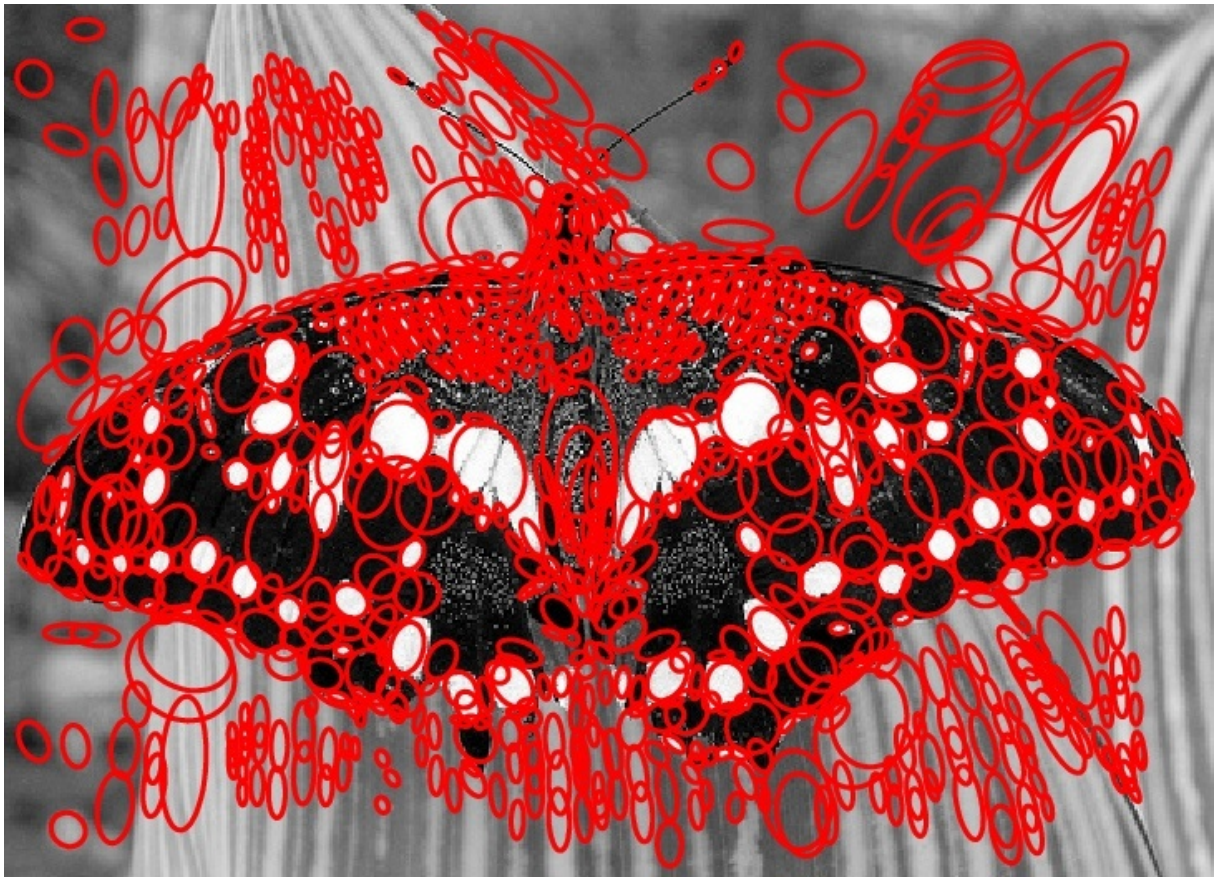
Affine adaptation example



Scale-invariant regions (blobs)

S.Lazebnik, UNC

Affine adaptation example



Affine-adapted blobs

S.Lazebnik, UNC

Affine Covariant Regions

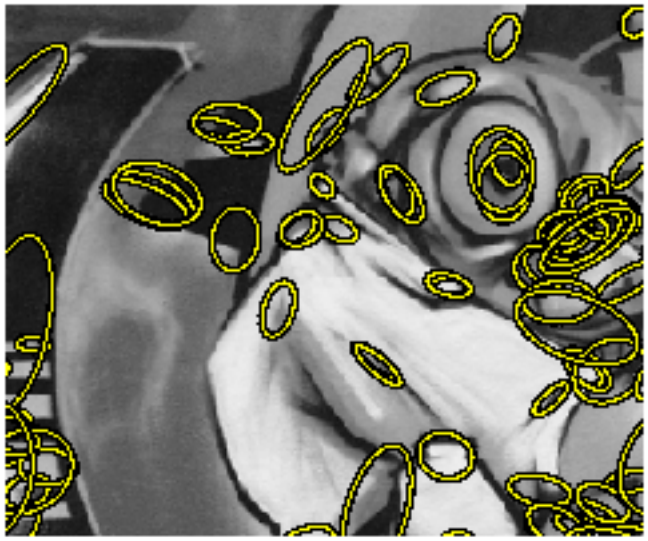


Image 1

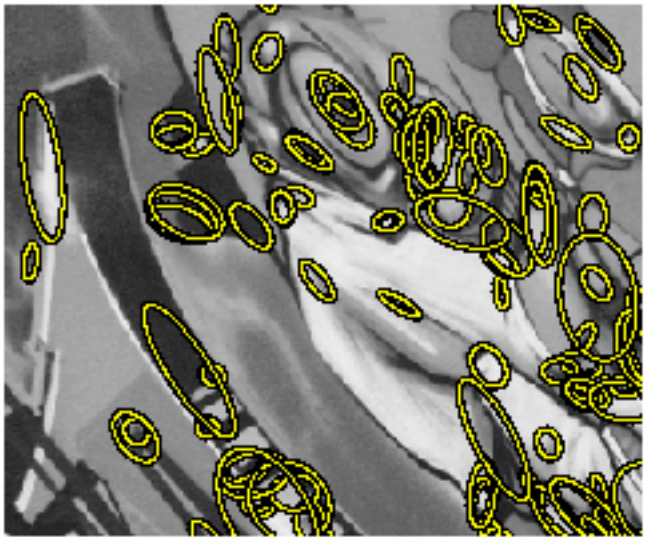


Image 2

Note some elliptical regions of the image are now being detected reliably across a large view deformation.

Affine Covariant Regions

- Affinely transformed versions of the same neighborhood will give rise to ellipses that are related by the same transformation
- What to do if we want to compare these image regions?
- *Affine normalization*: transform these regions into same-size circles

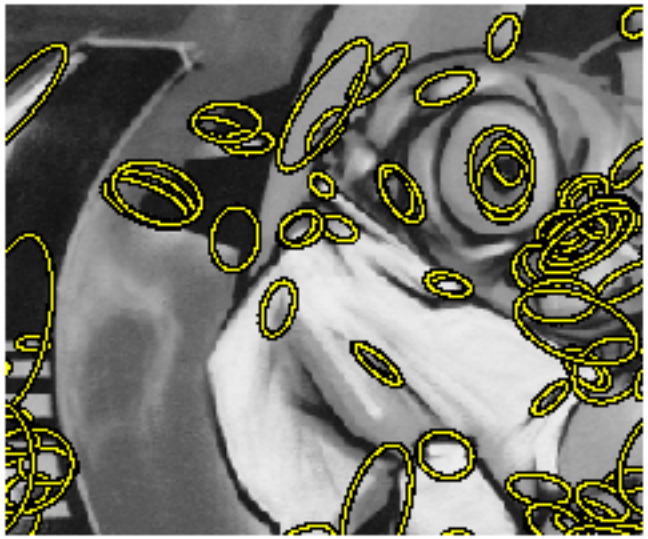


Image 1

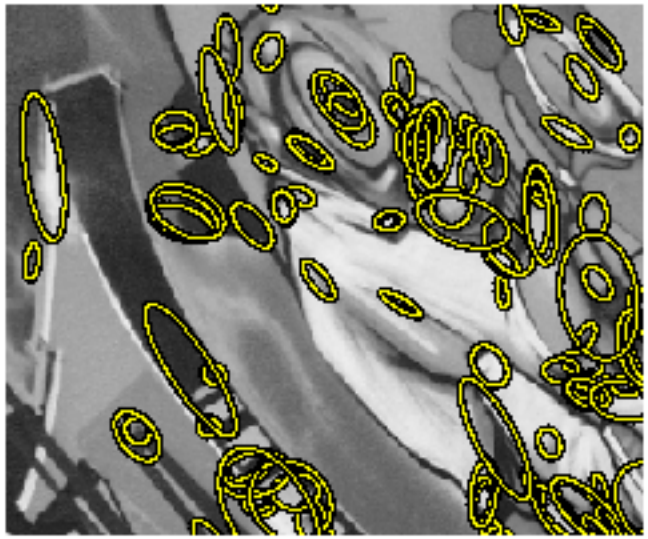
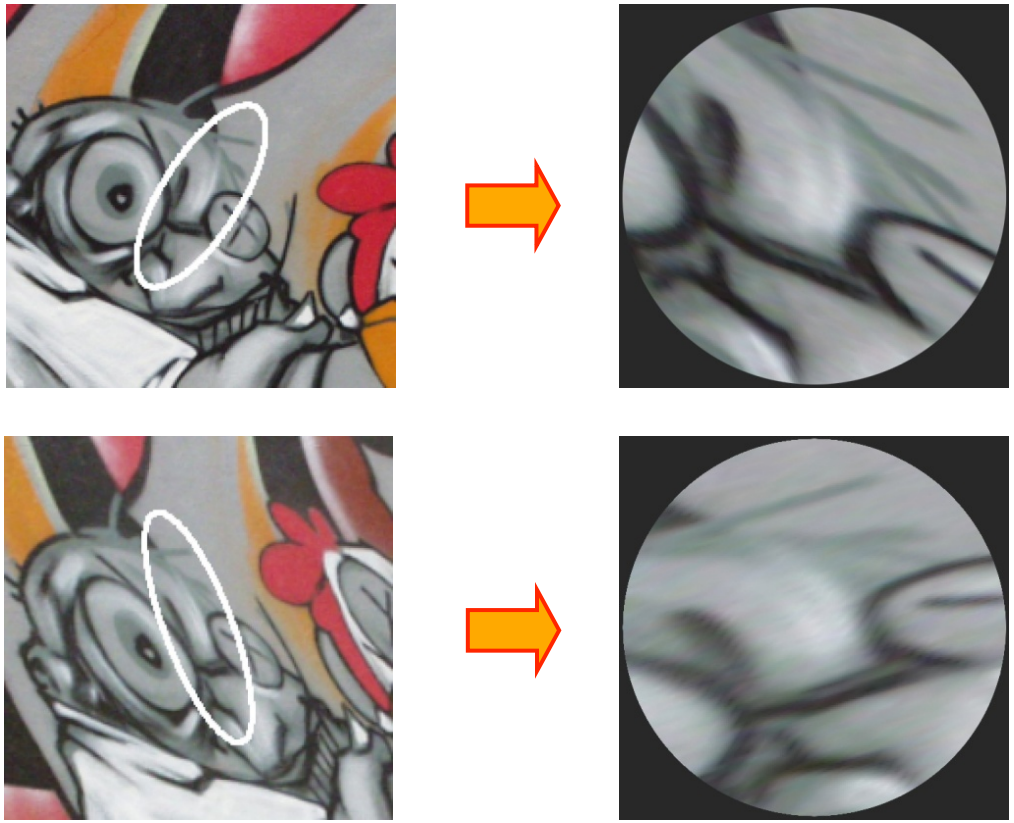


Image 2

S.Lazebnik, UNC

Affine normalization

- Perform affine transformation to map ellipse to circle
- Problem: not unique – we can rotate a unit circle and it still stays a unit circle



But we could deal with this by choosing a canonical orientation in the same way we did with SIFT!

Area/Region Features

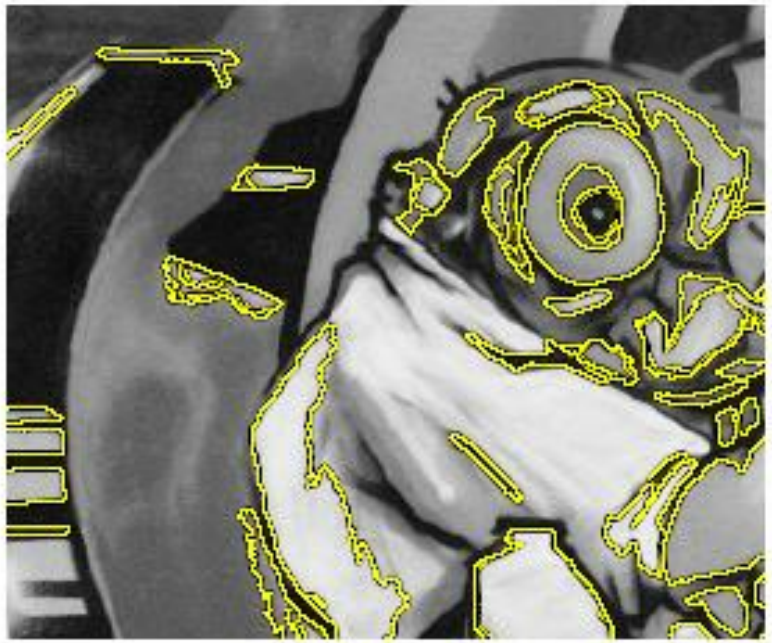
- Some features aren't based on gradients!
- MSER – maximally stable extremal regions
- Superpixels

Maximally Stable Extremal Regions [Matas '02]

- Based on Watershed segmentation algorithm
- Select regions that stay stable over a large parameter range of intensity thresholding

K. Grauman, B. Leibe

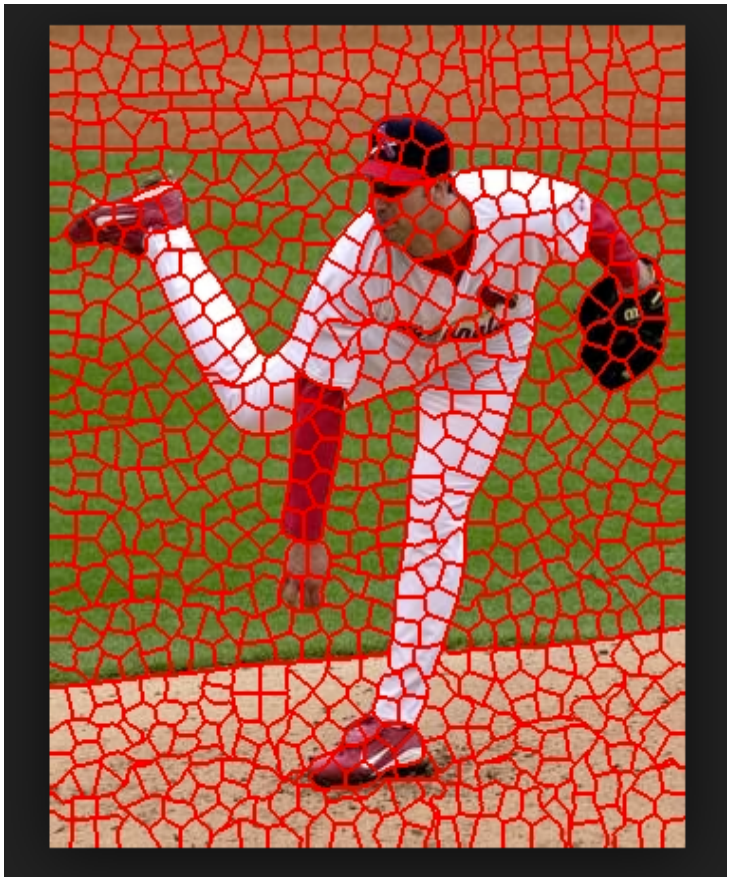
Example Results: MSER

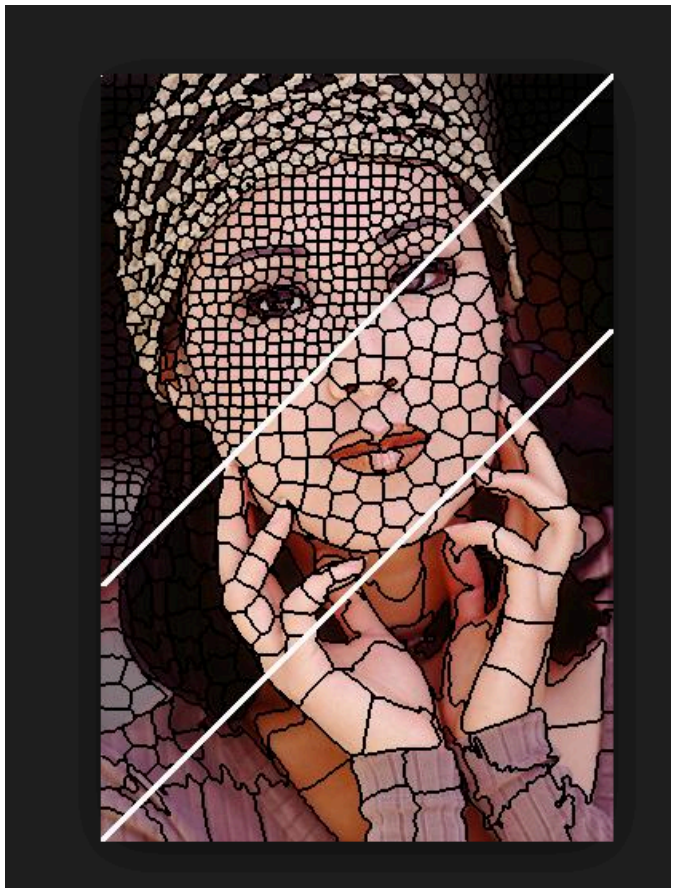


K. Grauman, B. Leibe

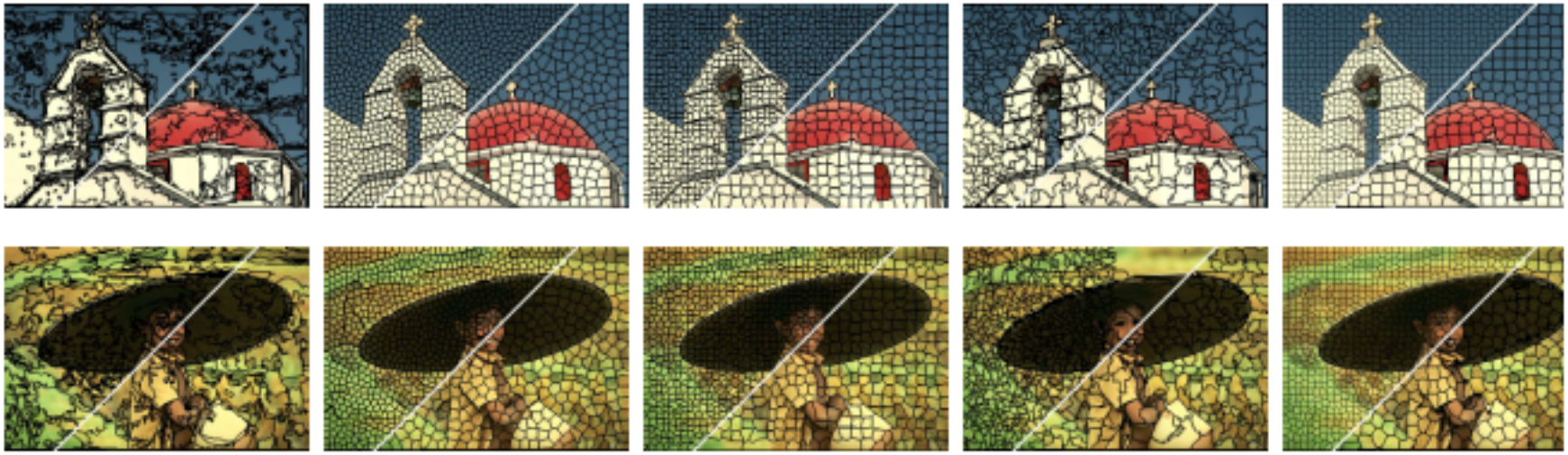
Superpixels

Small, connected (and often compact) regions formed by grouping pixels having similar color or intensity.





Superpixels



[GS04] Graph-based segmentation

[NC05] Normalized cuts

[TP09] Turbopixels

[QS09] QuickShift

SLIC

[GS04] Felzenszwalb, P., Huttenlocher, D.: Efficient graph-based image segmentation. IJCV (2004)
[NC05] G. Mori, Guiding Model Search Using Segmentation. ICCV (2005).
[TP09] Levinshtein, A., Stere, A., Kutulakos, K., Fleet, D., Dickinson, S., Siddiqi, K.:Turbopixels: F superpixels using geometric flows. PAMI (2009)
[QS09] Vedaldi, A., Soatto, S.: Quick shift and kernel methods for mode seeking. ECCV (2008)

Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk, SLIC Superpixels Compared to State-of-the-art Superpixel Methods, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, num. 11, p. 2274 - 2282, May 2012.

Outline

- Per-pixel processing
- Edges, corners, blobs/regions
- Feature descriptors

Recall: Concepts to Keep in Mind

- Dense vs sparse
  - dense: compute descriptor at each pixel (or on a grid)
    (example: convolution with a linear operator)
  - sparse: detect "good" places to compute descriptors
    (example: corner detection)
- Invariant vs covariant
  - Invariant: value does not change after transformation
    (example: normalized correlation is invariant to gain and offset)
  - Covariant: value changes reliably with some parameter of the transformation
    (example: area of a square scales by 4 when image is magnified by 2)

Alternate Outline of this Lecture

Dense Descriptors (computed at every pixel)

- Per-pixel processing

Sparse Descriptors

Where should we compute them?

- Edges, corners, blobs/regions

What should those descriptors be?

- Feature descriptors

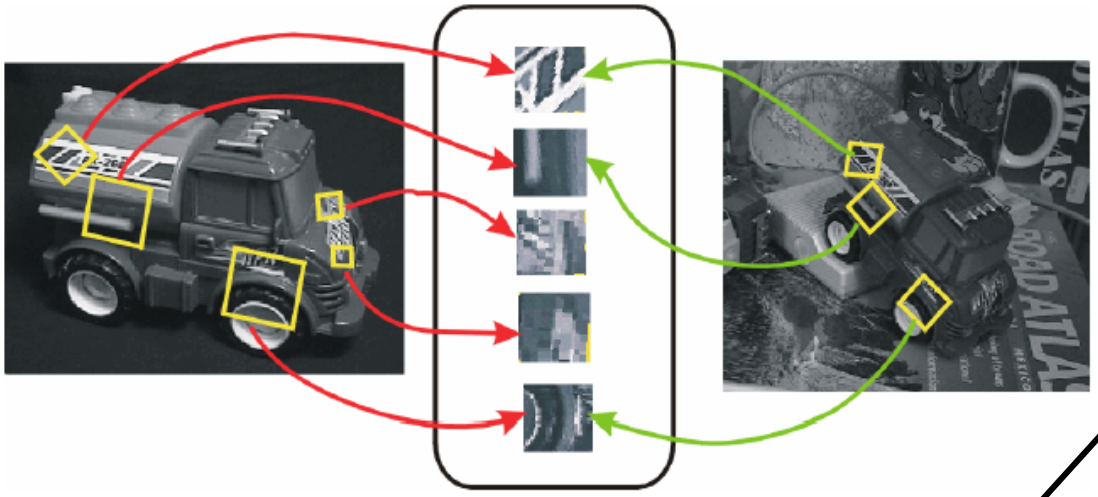
Dense Descriptors

- Everything we talked about in per-pixel processing can be considered a dense descriptor.
- Examples
  - Normalized intensity/color patches
  - Gabor jets
  - Haar wavelets
  - Textons
  - Any vector of filter bank responses

Sparse Descriptors, aka Local Feature Approach

- Interest point detection, for example:
  - Harris corner detector
  - Laplacian of Gaussian blob
- Extract descriptors, for example:
  - Intensity patch normalized for scale and rotation
  - Histograms for robustness to small shifts and translations (e.g. SIFT descriptor)

Normalized Intensity Patches

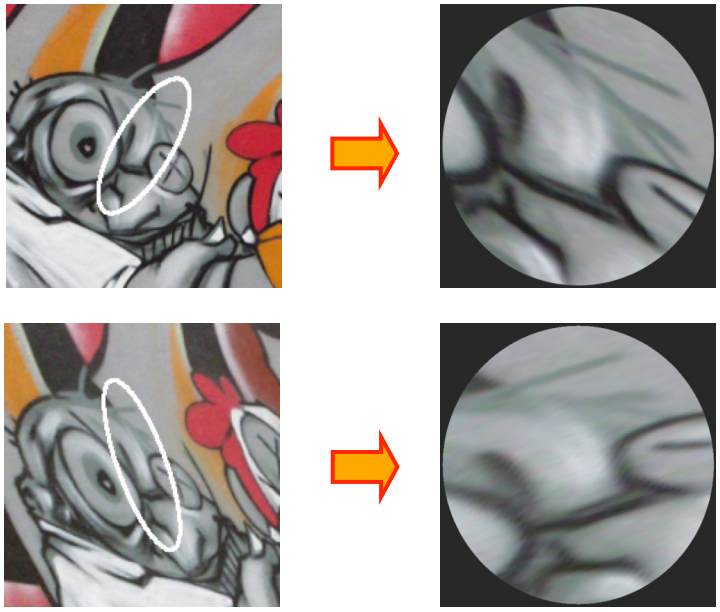


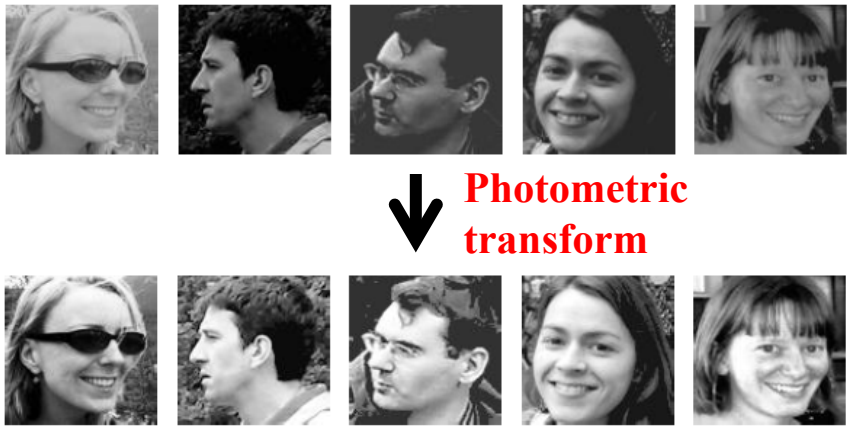
Similarity transform

Normalized patches

Affine transform

Normalized patches



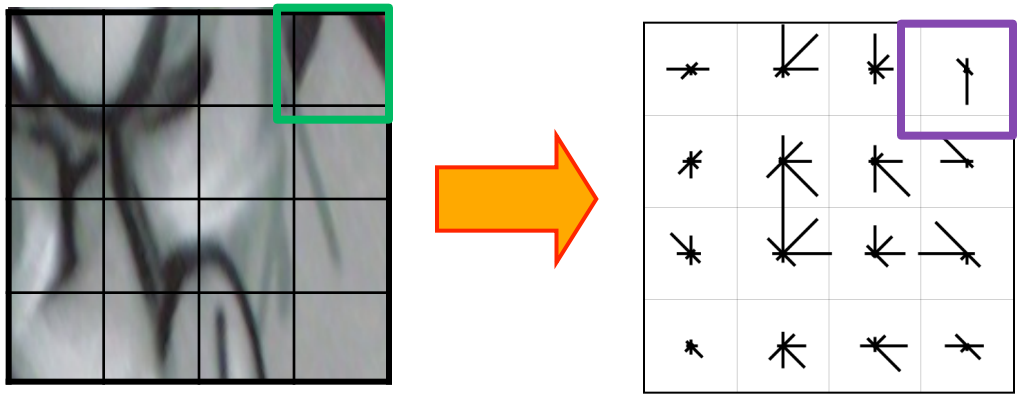


Photometric transform

Normalized patches

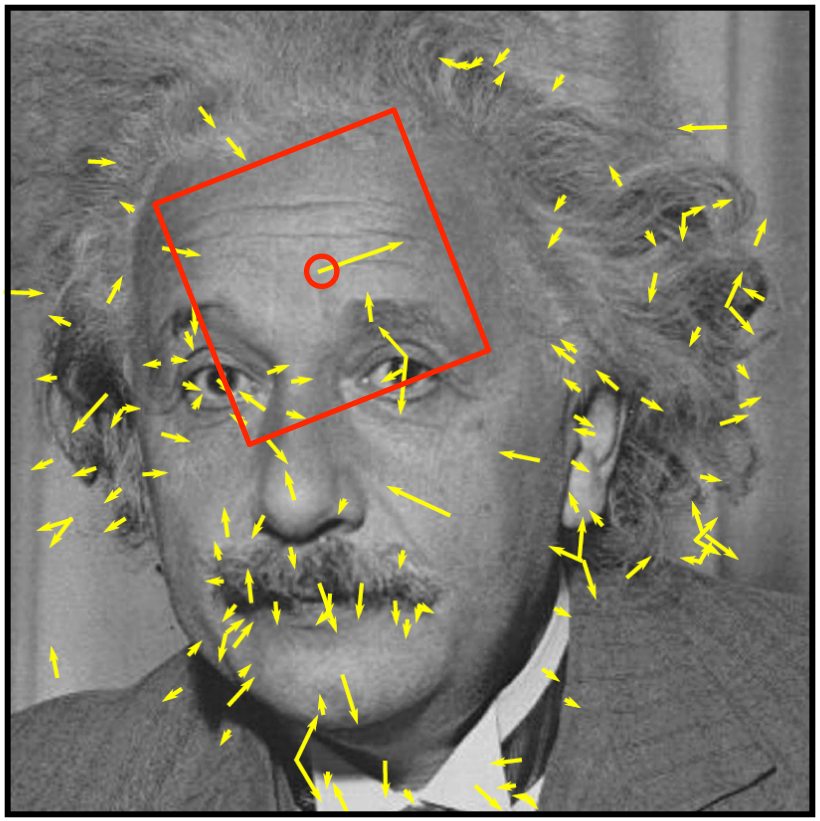
Histogram-based Descriptors

Descriptor vectors based on histograms of gradients (e.g. SIFT and HOG) offer even more resilience to local geometric deformations and changes of illumination.



Sift Descriptor

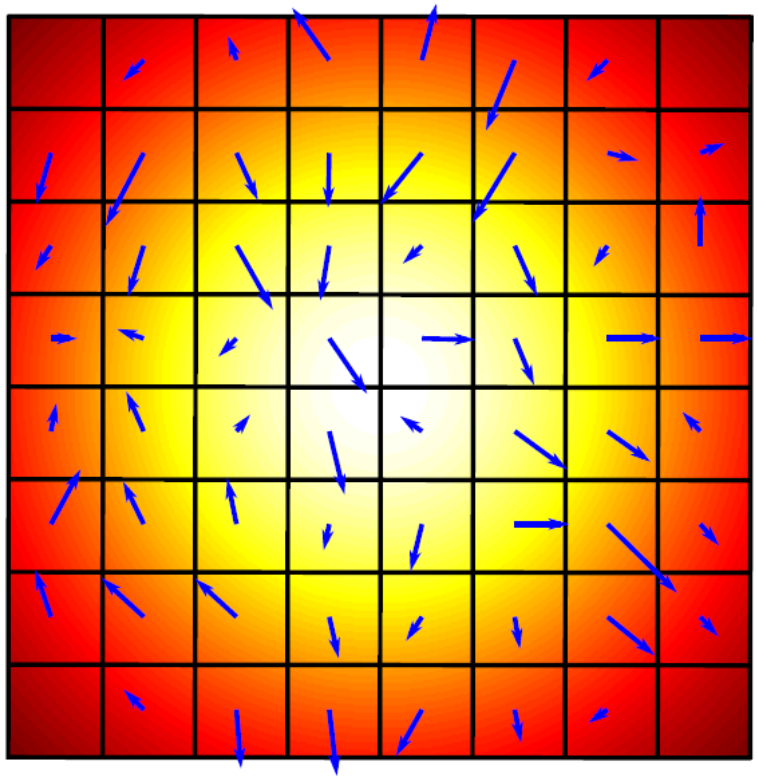
Goal: produce a vector that describes the region around the interest point.



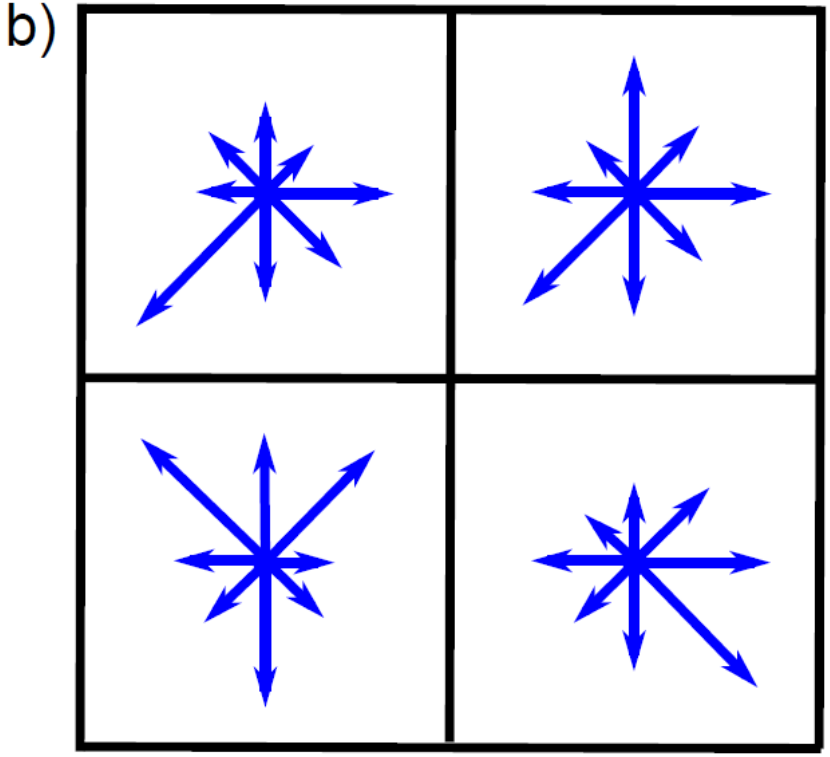
All calculations are relative to the orientation and scale of the keypoint

Makes descriptor invariant to rotation and scale

Sift Descriptor



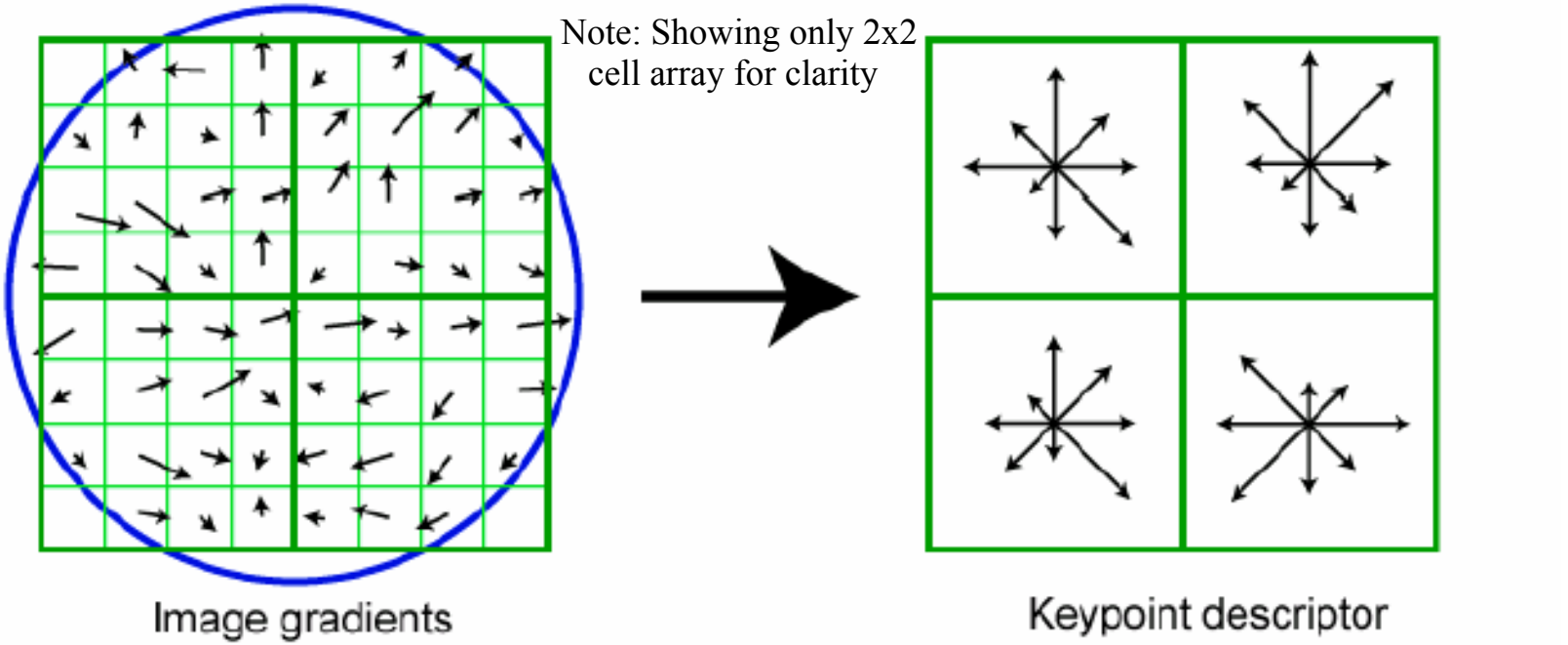
1. Compute image gradients



2. Pool into local histograms
3. Concatenate histograms
4. Normalize histograms

SIFT Descriptor Vector

- Take image gradients sampled over 16x16 array of locations at the appropriate scale level in scale-space
- Form cells containing 4x4 sets of gradients and compute a gradient direction histogram of each cell.
- 8 directions * 4x4 array of cells = 128 numbers
- Normalize the resulting 128-dimensional vector (e.g. unit vector)

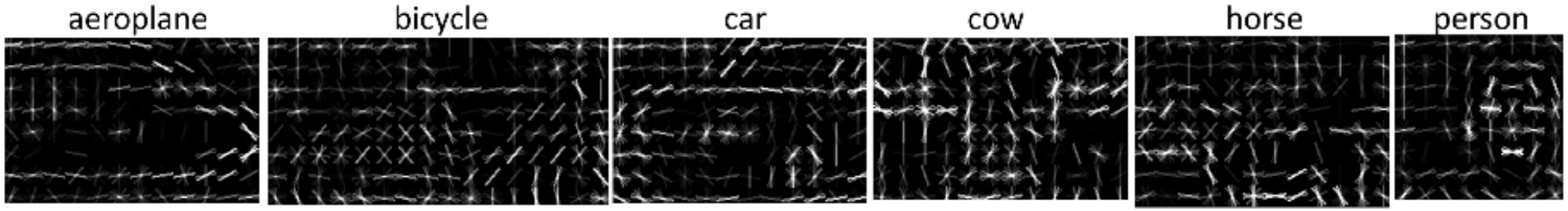


Each descriptor is thus a 128-dimensional vector

HOG Descriptors

Dalal and Triggs, 2005, Histograms of Oriented Gradients for Human Detection

- HOG = Histogram of Oriented Gradients
- Similar to SIFT descriptors in that they are formed from histograms of gradients over an array of cells, but with without regard to sign of contrast (represents orientation instead of direction)
- Computed at a single predetermined scale
- No rotation compensation. Originally developed for detecting pedestrians, who are mainly upright in the image



HoG Descriptor

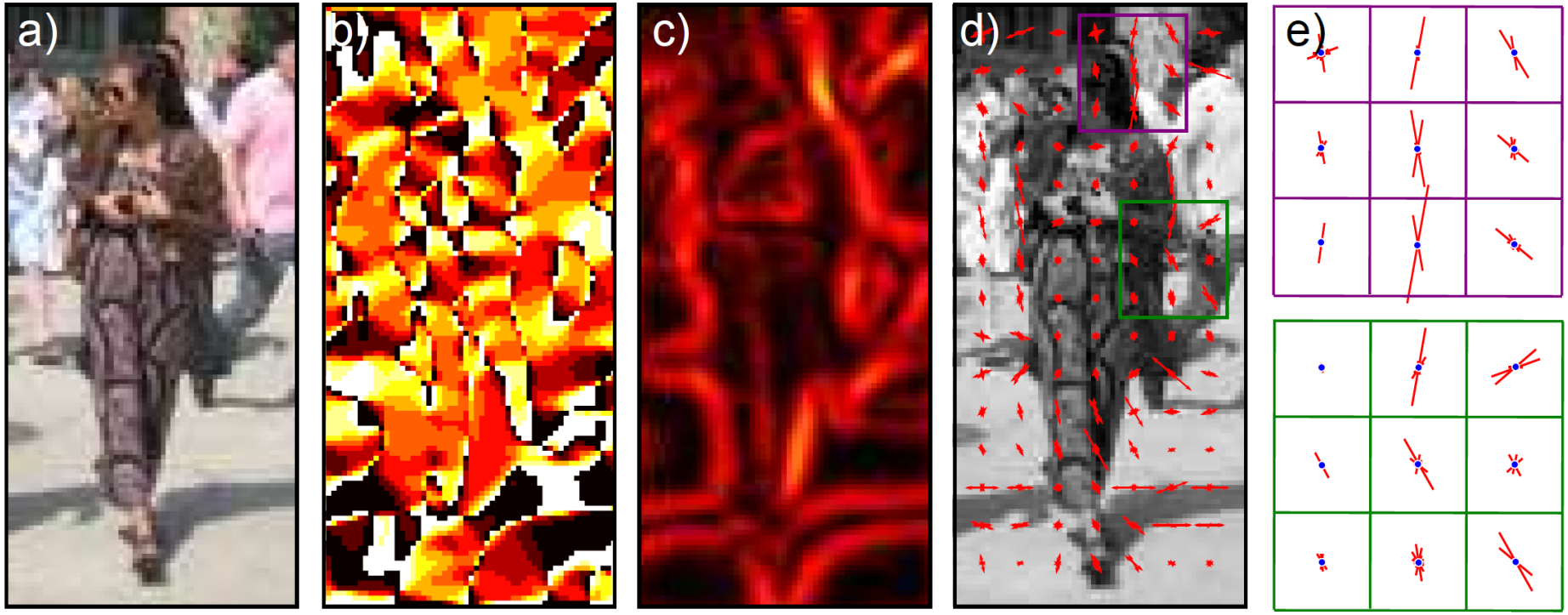
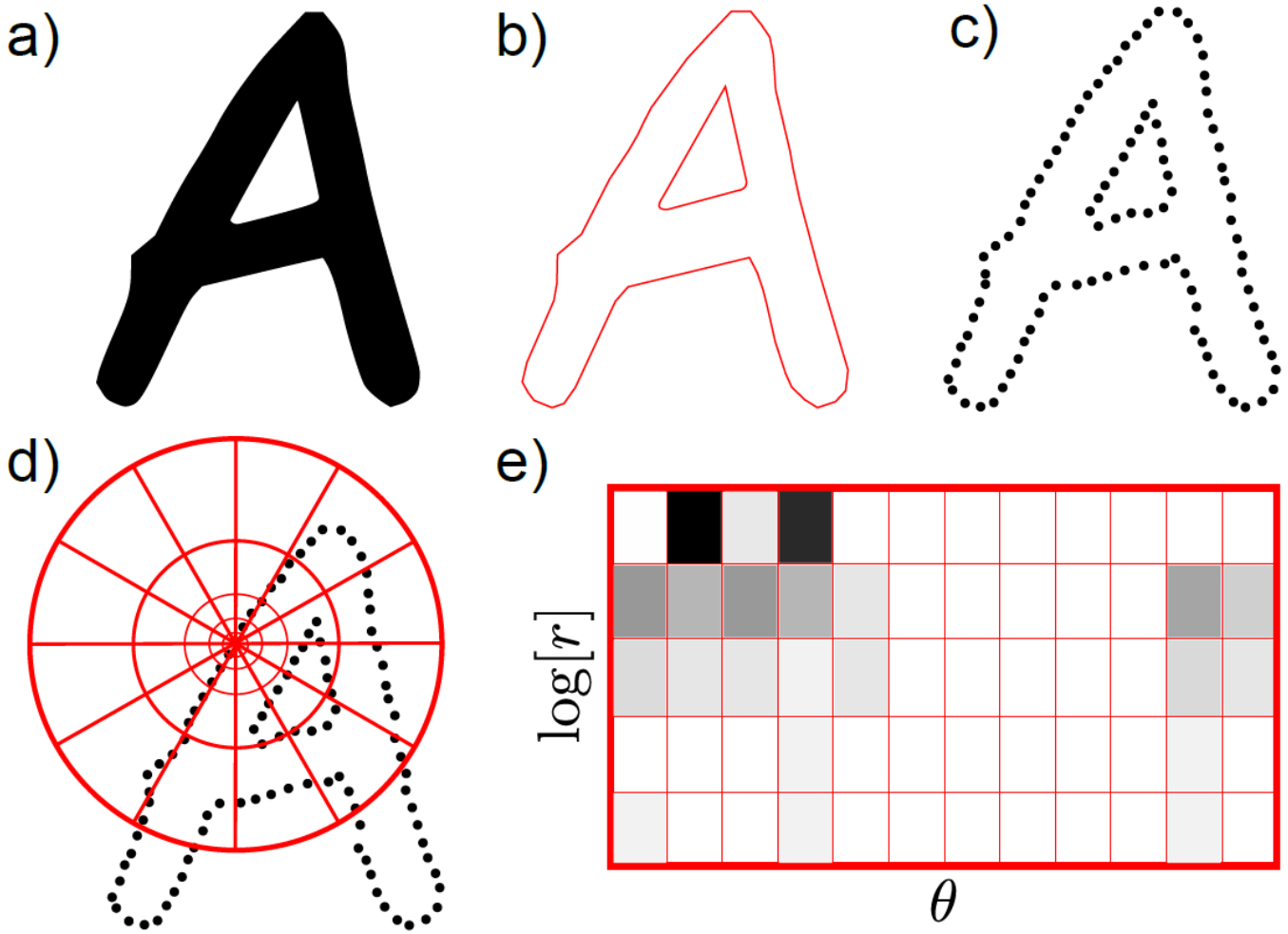


Figure 13.17 HOG descriptor. a) Original image. b) Gradient orientation, quantized into 9 bins from $0 - 180^o$. c) Gradient magnitude. d) Cell descriptors are 9D orientation histograms that are computed within 6×6 pixel regions. e) Block descriptors are computed by concatenating 3×3 blocks of cell descriptors. The block descriptors are normalized. The final HOG descriptor consists of the concatenated block descriptors.

Bag of words descriptor

- Find interest points in a region (could be whole image)
- Compute descriptor around each
- Find closest match in vocabulary and assign index (word) for each descriptor
- Compute histogram of these indices over the region
- Note: Dictionary (words) computed over a training set using K-means

Shape context descriptor



What else is in Chapter 13?

Chapter 13 also discusses PCA

A method for dimensionality reduction. You can think of it as computing new lower-dimensional descriptors from original high-dimensional ones.

and K-means clustering

Also reduces dimensionality by clustering into a discrete set of clusters (exemplars; visual words). We will talk about K-means and other clustering methods in the next few lectures.