Computer vision: models, learning and inference

Chapter 13

Image preprocessing and feature extraction

Preprocessing

- The goal of pre-processing is
  - to try to reduce unwanted variation in image due to lighting, scale, deformation etc.
  - to reduce data to a manageable size
- Give the subsequent model a chance
- Preprocessing definition: deterministic transformation of pixels $\mathbf{p}$ to create data vector $\mathbf{x}$
- Usually heuristics based on experience

Structure

- Per-pixel transformations
- Edges, corners, and interest points
- Descriptors
- Dimensionality reduction

Normalization

- Fix first and second moments to standard values
- Remove contrast and constant additive luminance variations

Before

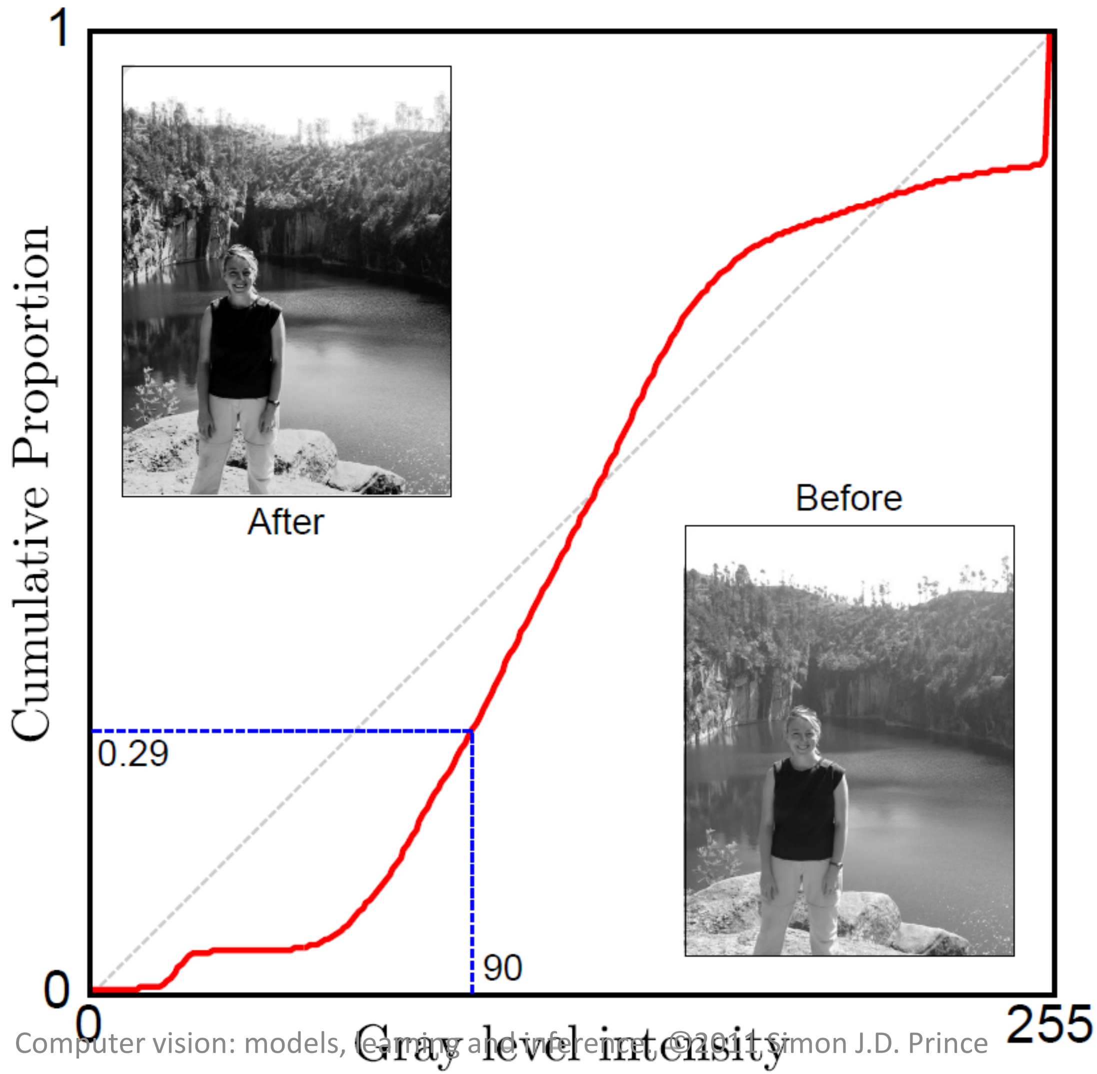
After

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



Convolution

Takes pixel image **P** and applies a 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, where weights given by filter.

Easiest to see with a concrete example

Blurring (convolve with Gaussian)

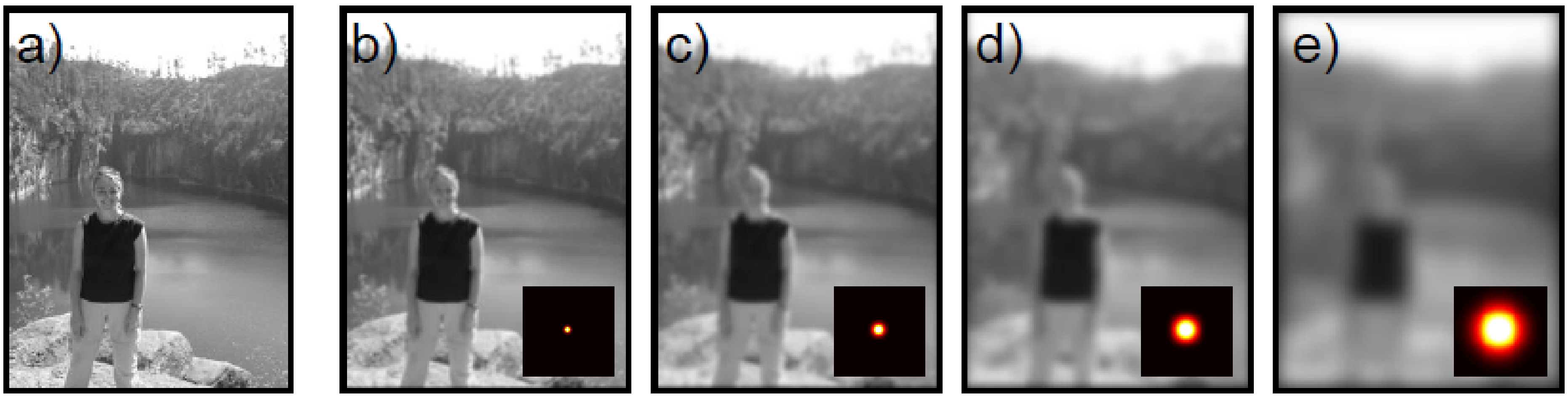
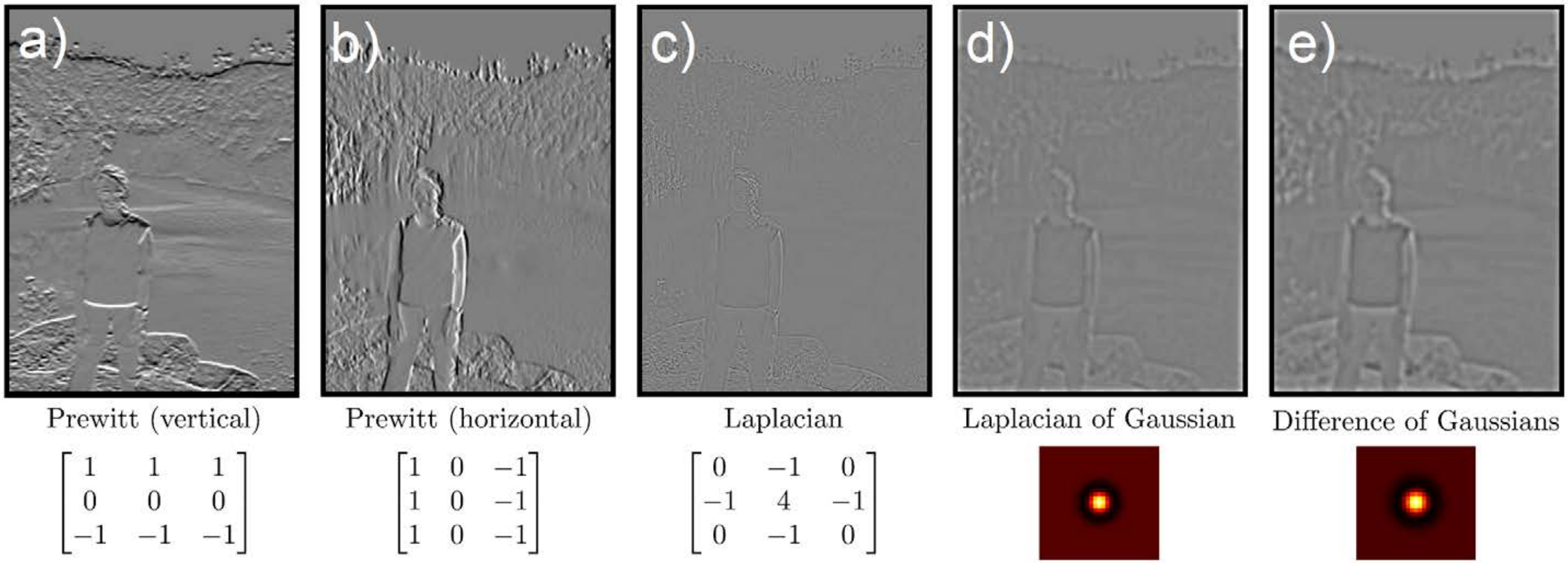


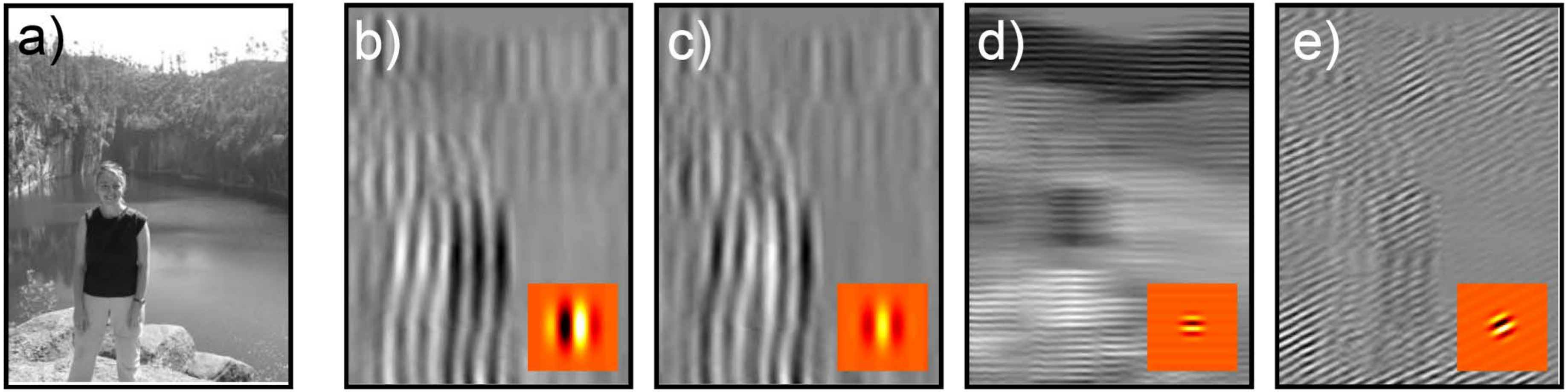
Figure B.3 Image blurring. a) Original image. b) Result of convolving with a Gaussian filter (filter shown in bottom right of image). The image is slightly blurred. c-e) Convolving with a filter of increasing standard deviation causes the resulting image to be increasingly blurred.

Gradient Filters



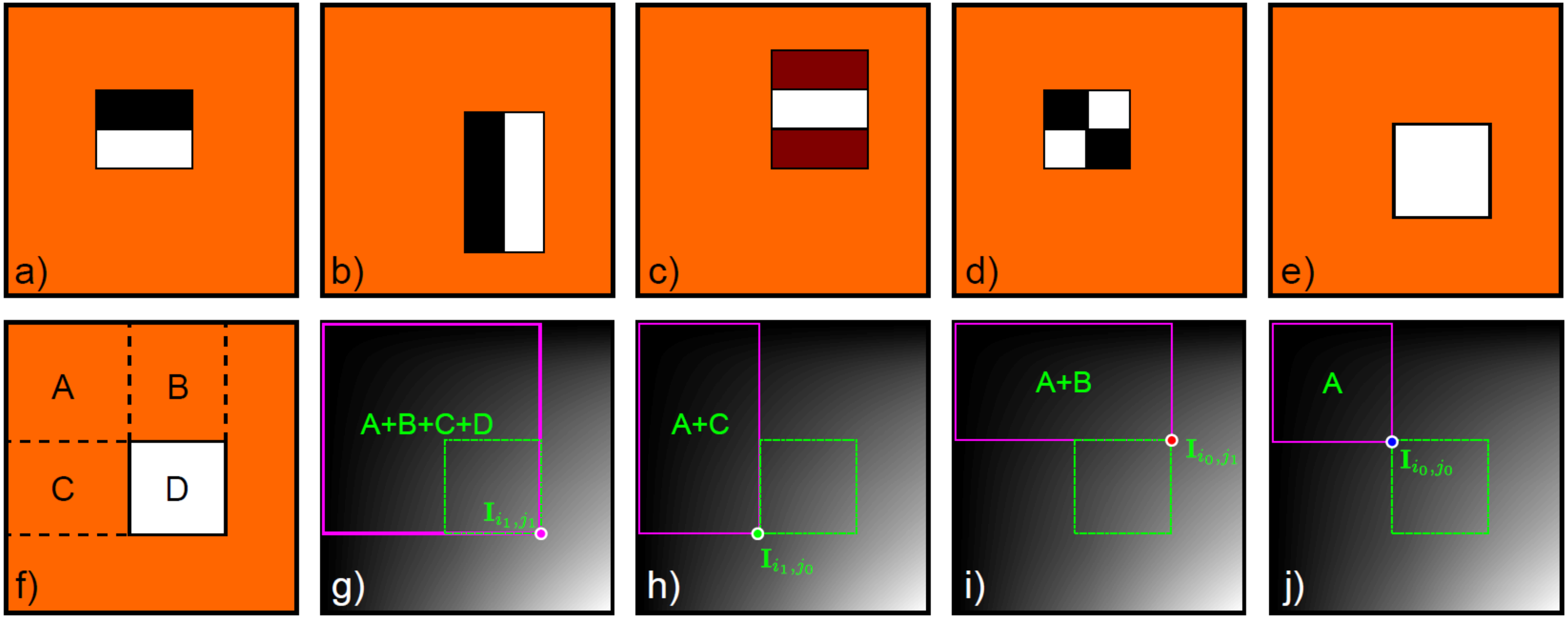
- Rule of thumb: big response when image matches filter

Gabor Filters

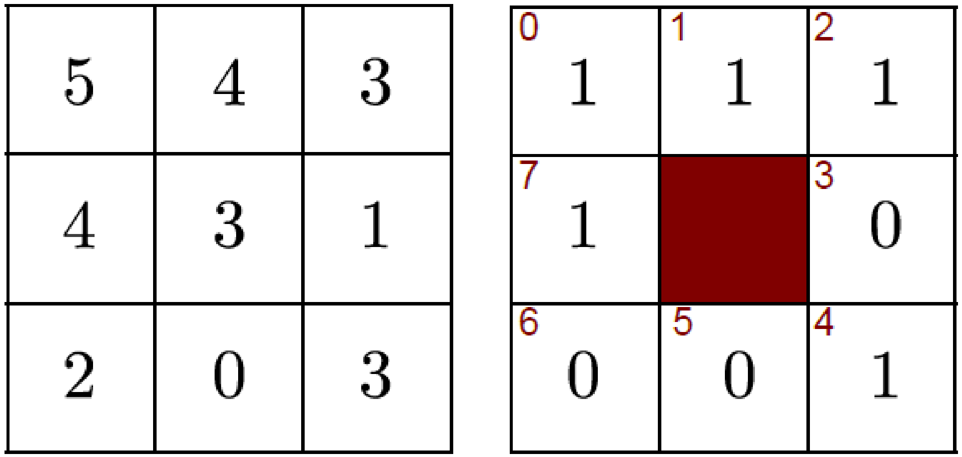


$$f_{mn} = \frac{1}{2\pi\sigma^2}\exp\left[-\frac{m^2+n^2}{2\sigma^2}\right]\sin\left[\frac{2\pi(\cos[\omega]m+\sin[\omega]n)}{\lambda}+\phi\right]$$

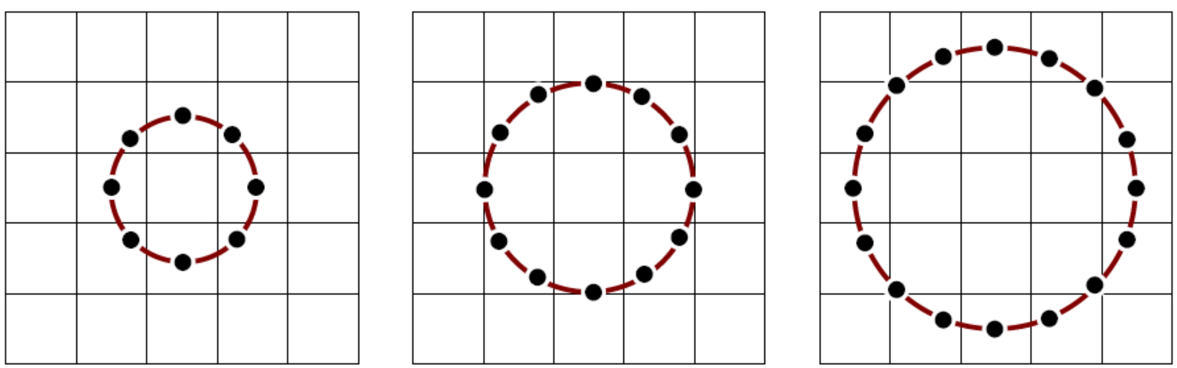
Haar Filters



Local binary patterns



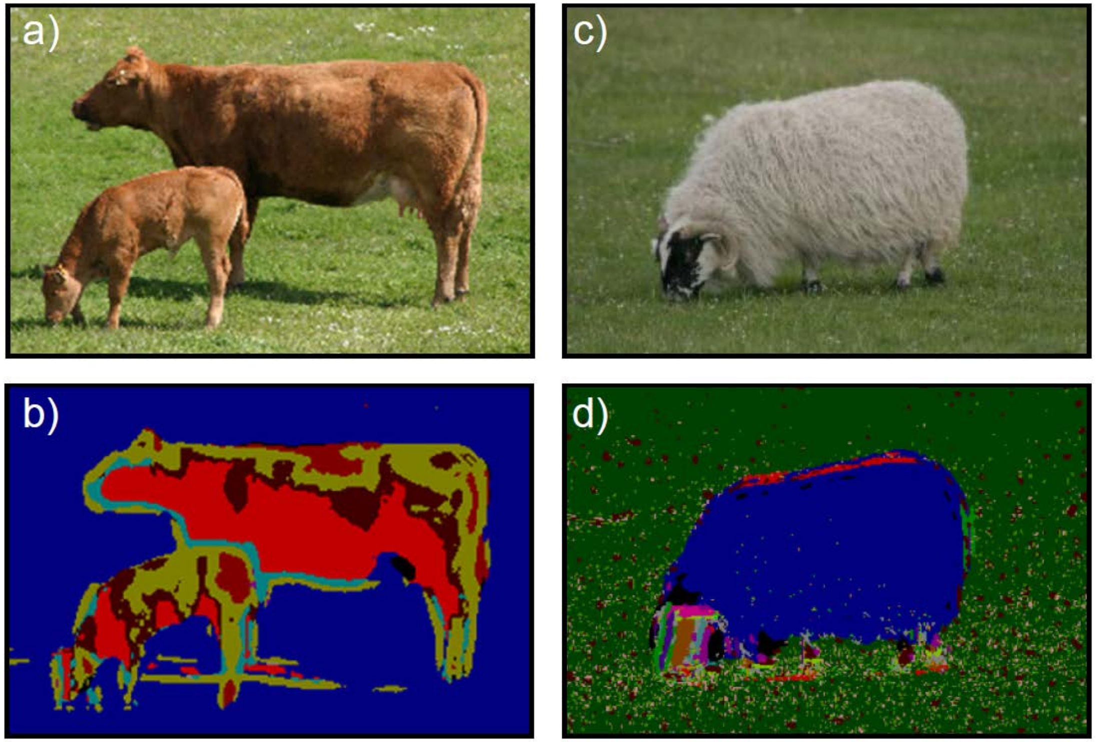
$$\text{LBP} = 10010111 = 151$$



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

Textons

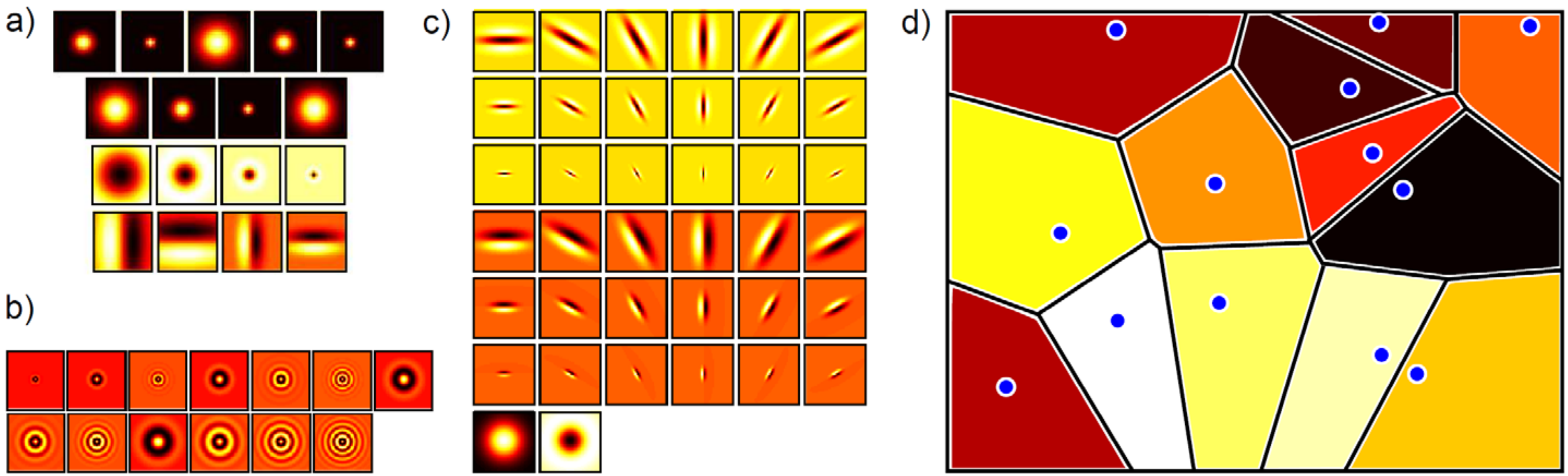
- An attempt to characterize texture
- Replace each pixel with integer representing the texture 'type'



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 bank, and assign to nearest cluster

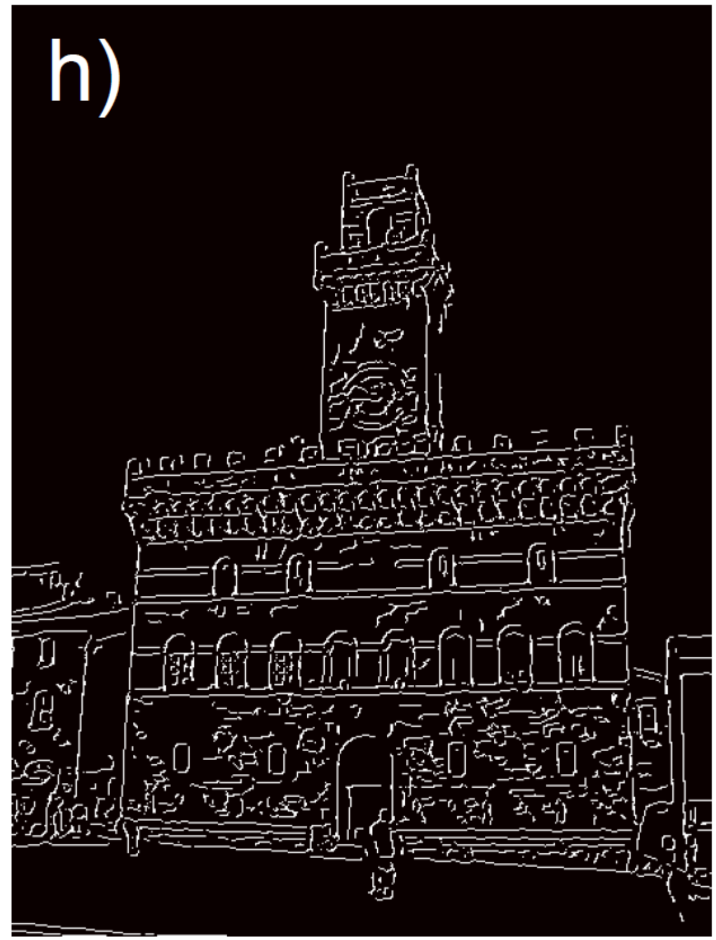
Structure

- Per-pixel transformations
- Edges, corners, and interest points
- Descriptors
- Dimensionality reduction

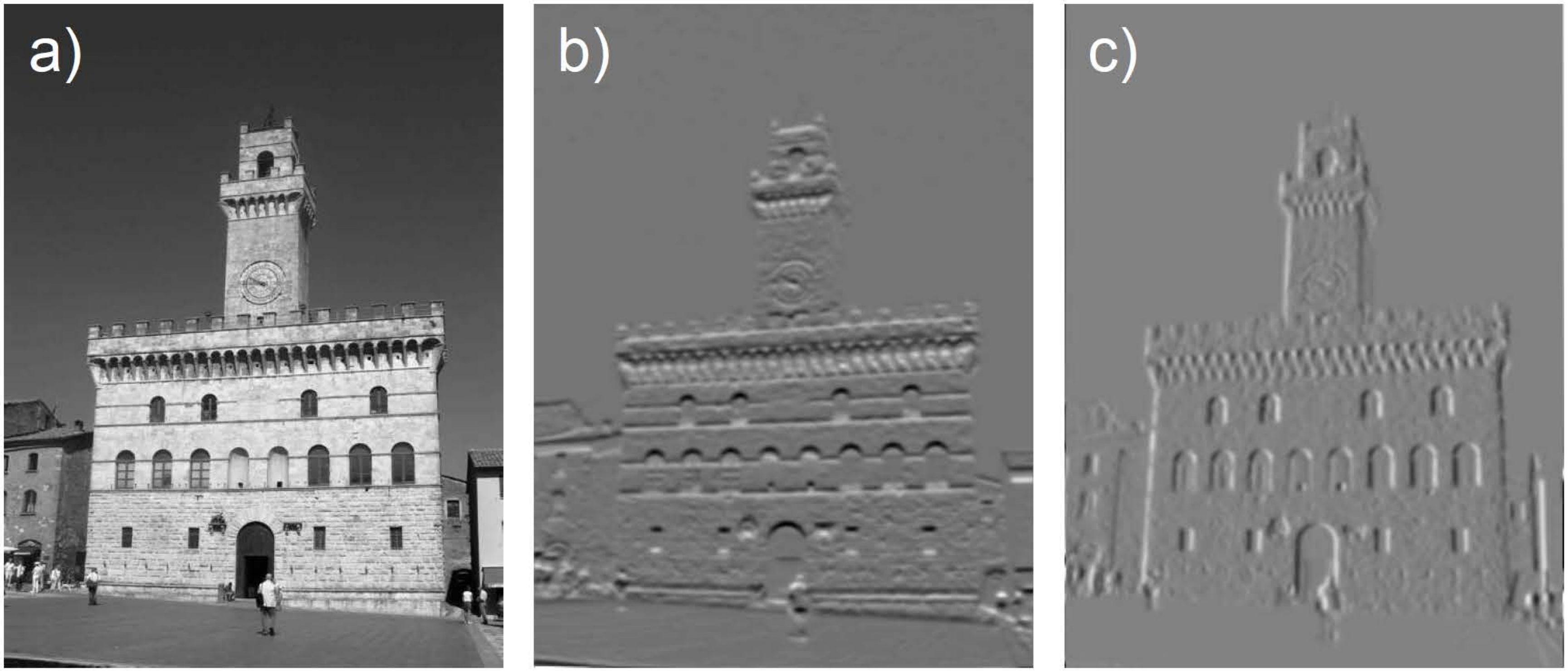
Edges

(from Elder and Goldberg 2000)

Canny Edge Detector

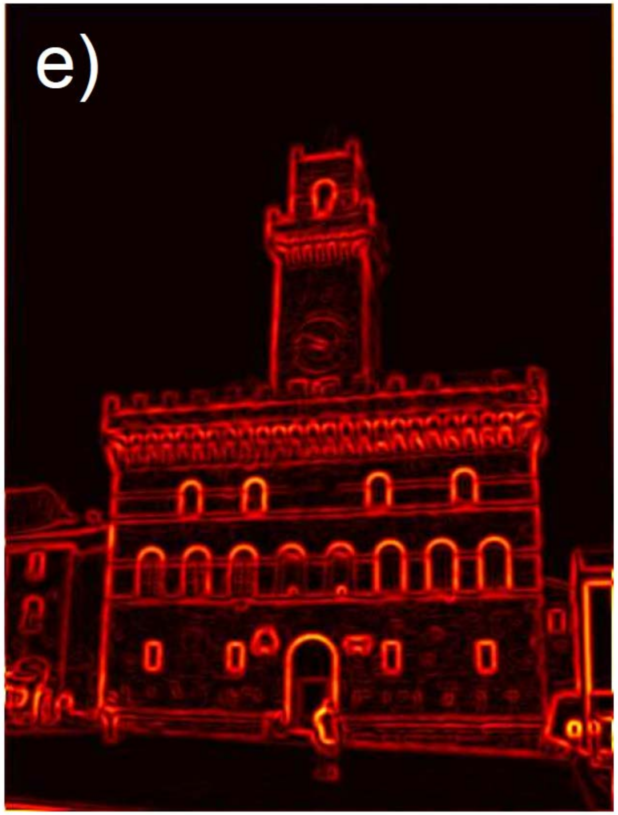


Canny Edge Detector

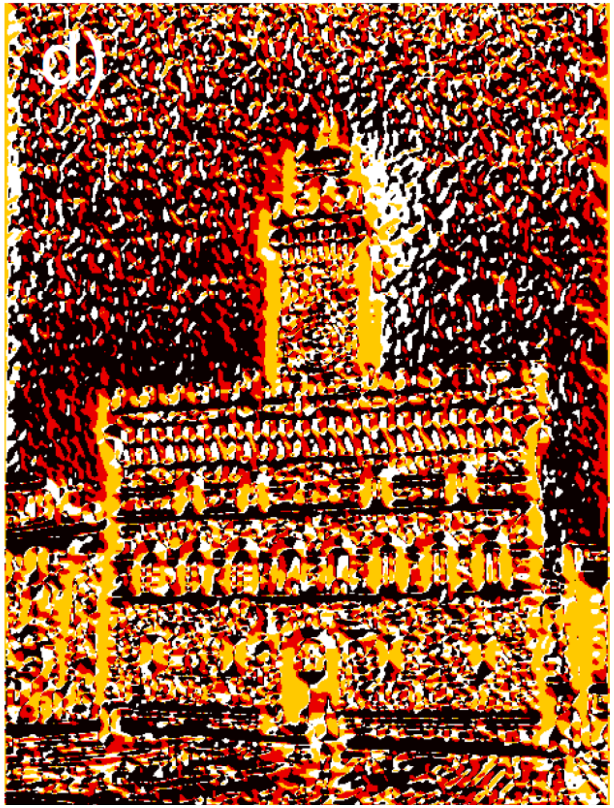


Compute horizontal and vertical gradient images **h** and **v**

Canny Edge Detector



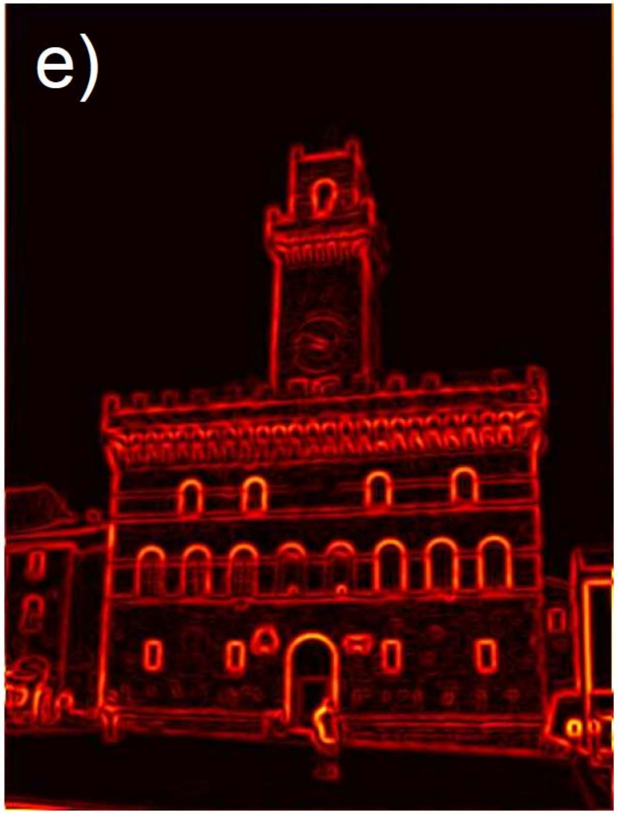
$$a_{ij} = \sqrt{h_{ij}^2 + v_{ij}^2}$$

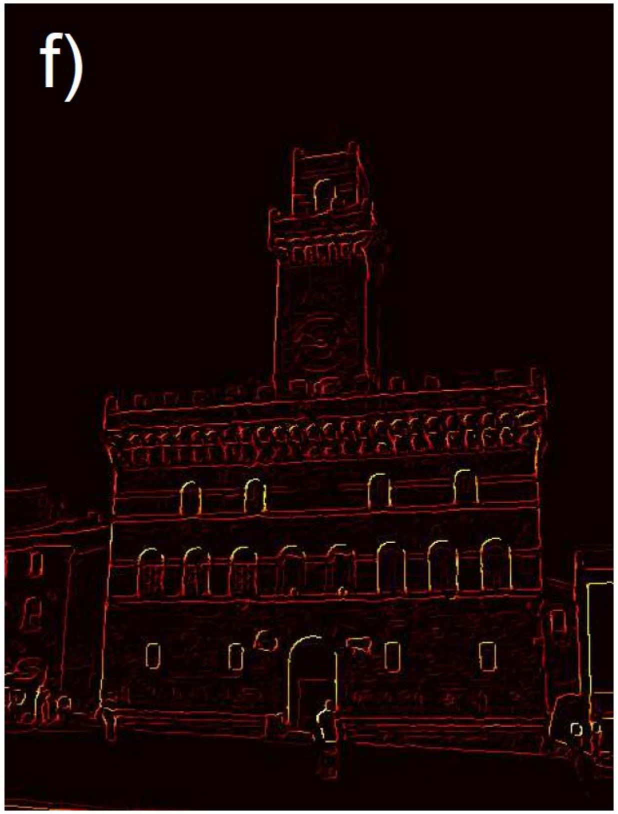


$$\boldsymbol{\theta}_{ij} = \arctan[v_{ij}/h_{ij}]$$

Quantize to 4 directions

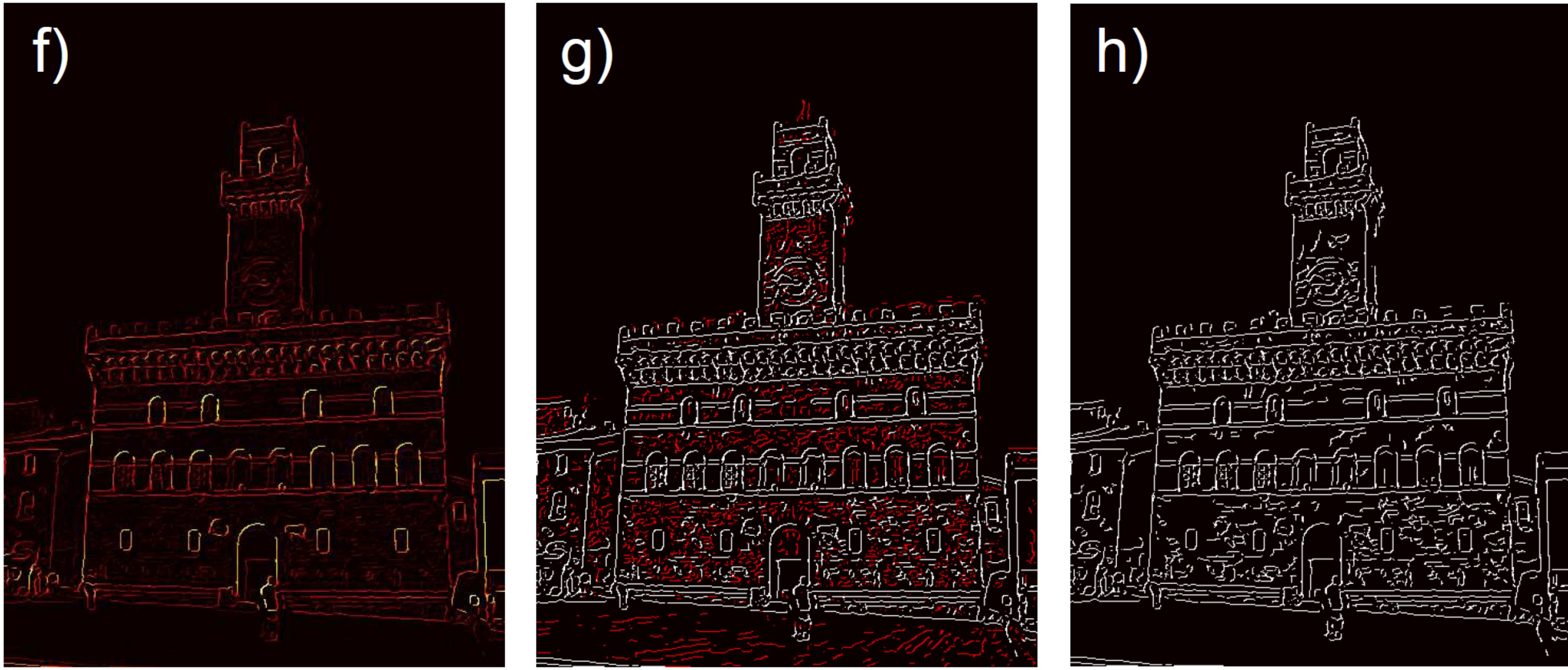
Canny Edge Detector





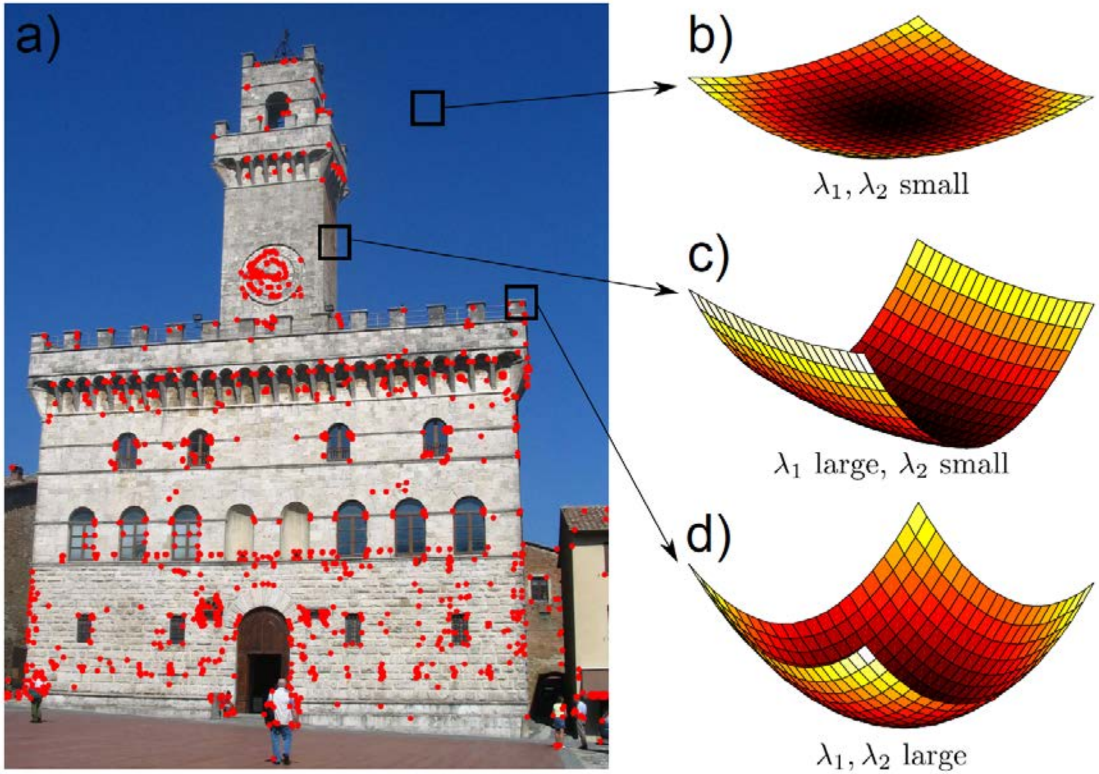
Non-maximal suppression

Canny Edge Detector



Hysteresis Thresholding

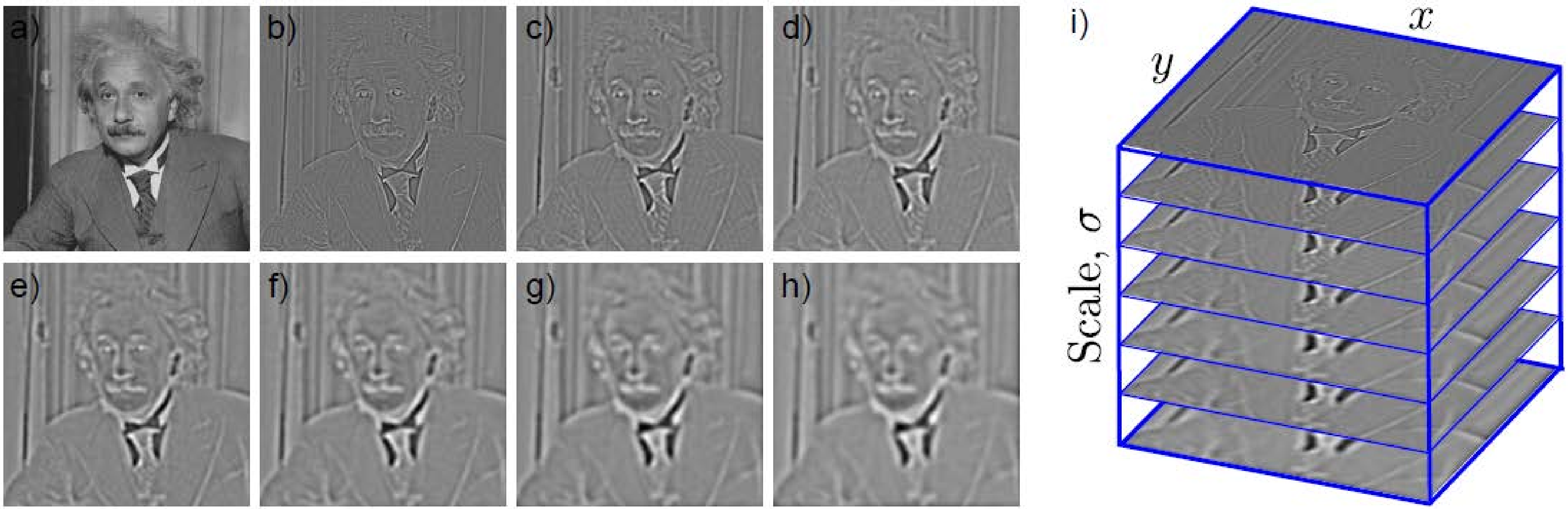
Harris Corner Detector



Make decision based on image structure tensor

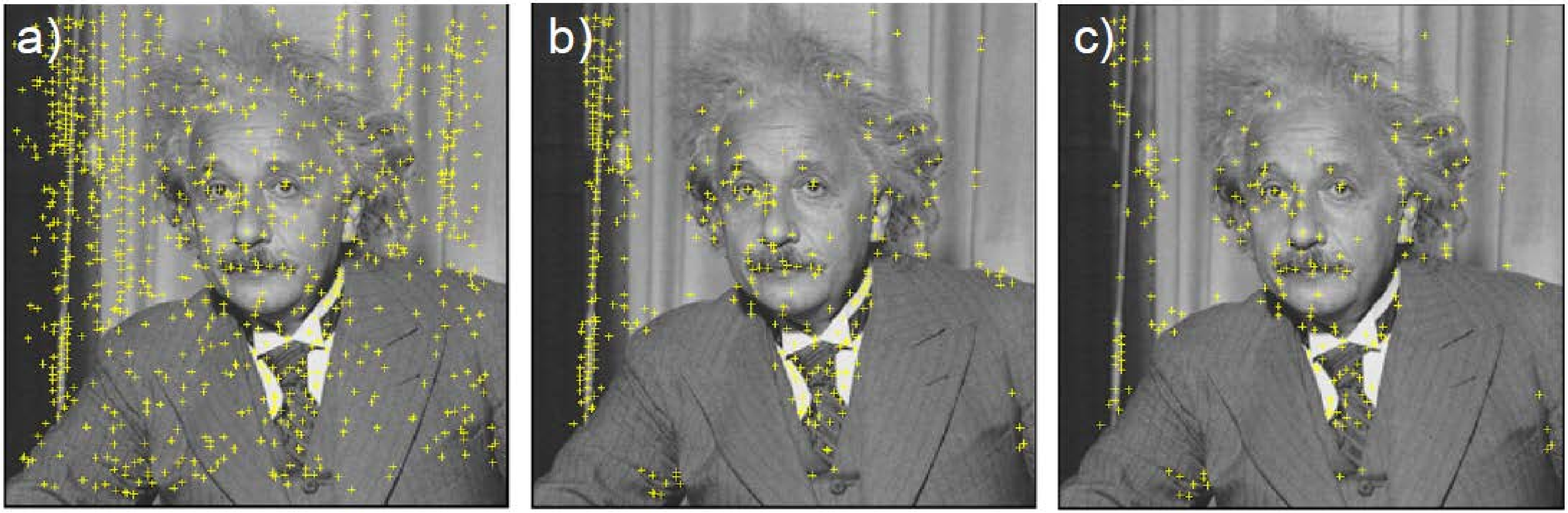
$$\mathbf{S}_{ij} = \sum_{m=i-D}^{i+D} \sum_{n=j-D}^{j+D} w_{mn} \begin{bmatrix} h_{ij}^2 & h_{ij}v_{ij} \\ h_{ij}v_{ij} & v_{ij}^2 \end{bmatrix}$$

SIFT Detector



Filter with difference of Gaussian filters at increasing scales
Build image stack (scale space)
Find extrema in this 3D volume

SIFT Detector



Identified Corners

Remove those on edges

Remove those where contrast is low

Assign Orientation

Orientation assigned by looking at intensity gradients in region around point

Form a histogram of these gradients by binning.

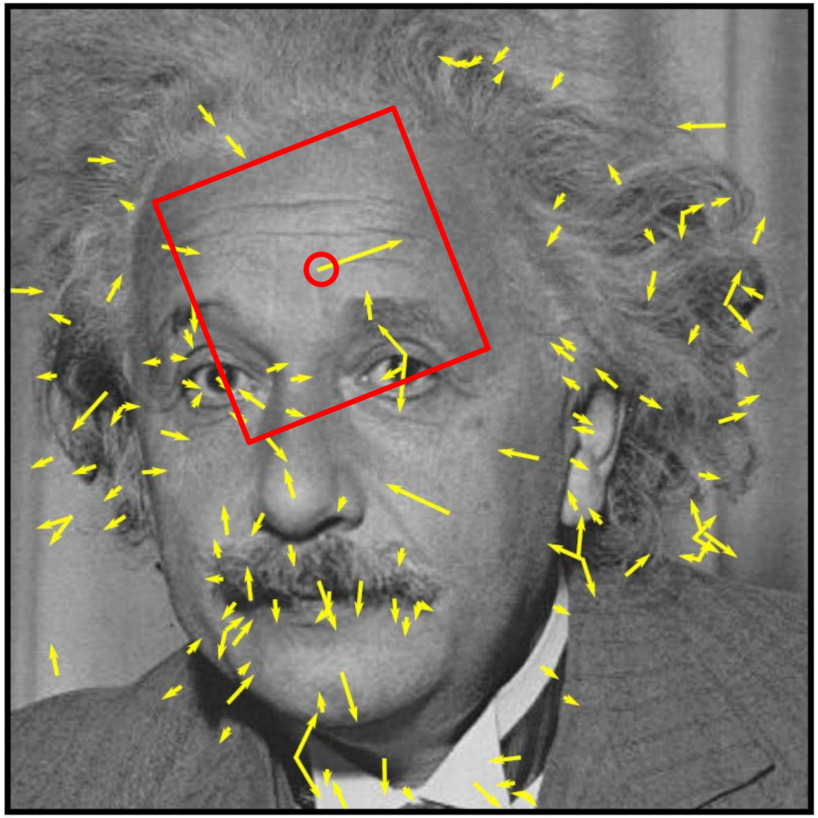
Set orientation to peak of histogram.

Structure

- Per-pixel transformations
- Edges, corners, and interest points
- Descriptors
- Dimensionality reduction

Sift Descriptor

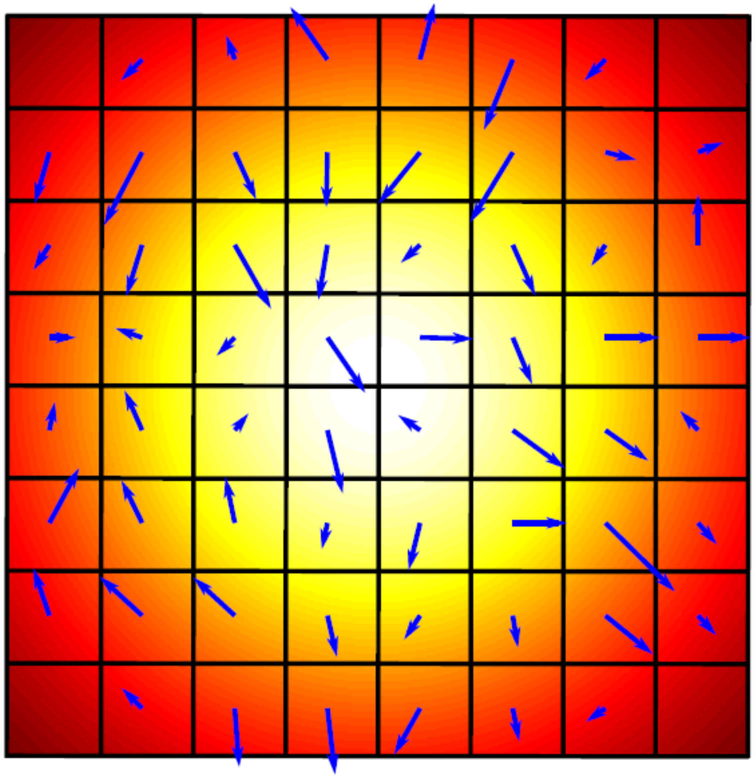
Goal: produce a vector that describes the region around the interest point.

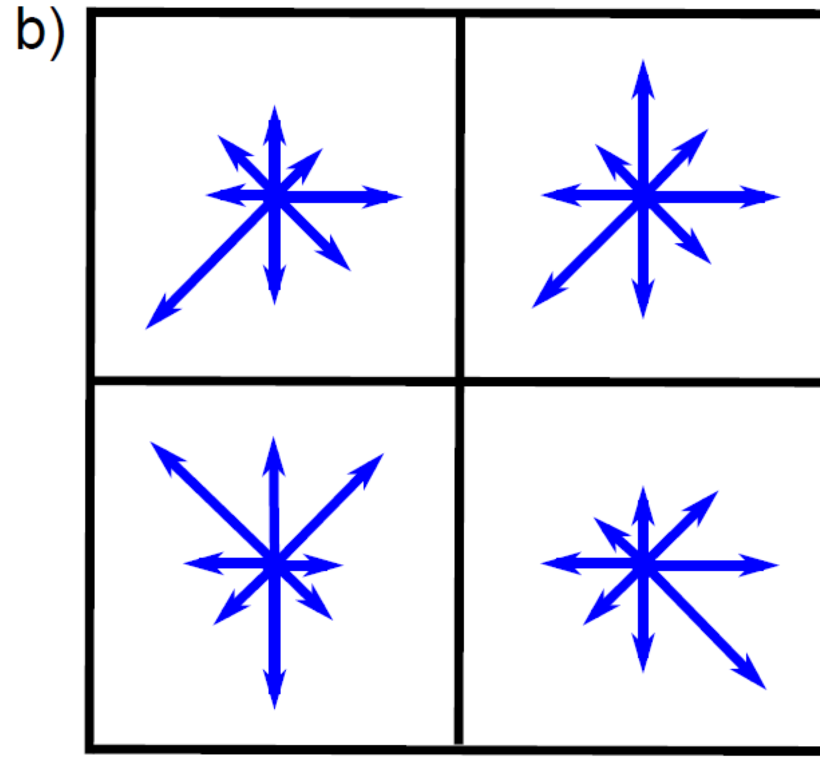


All calculations are relative to the orientation and scale of the keypoint

Makes descriptor invariant to rotation and scale

Sift Descriptor





1. Compute image gradients

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

HoG Descriptor

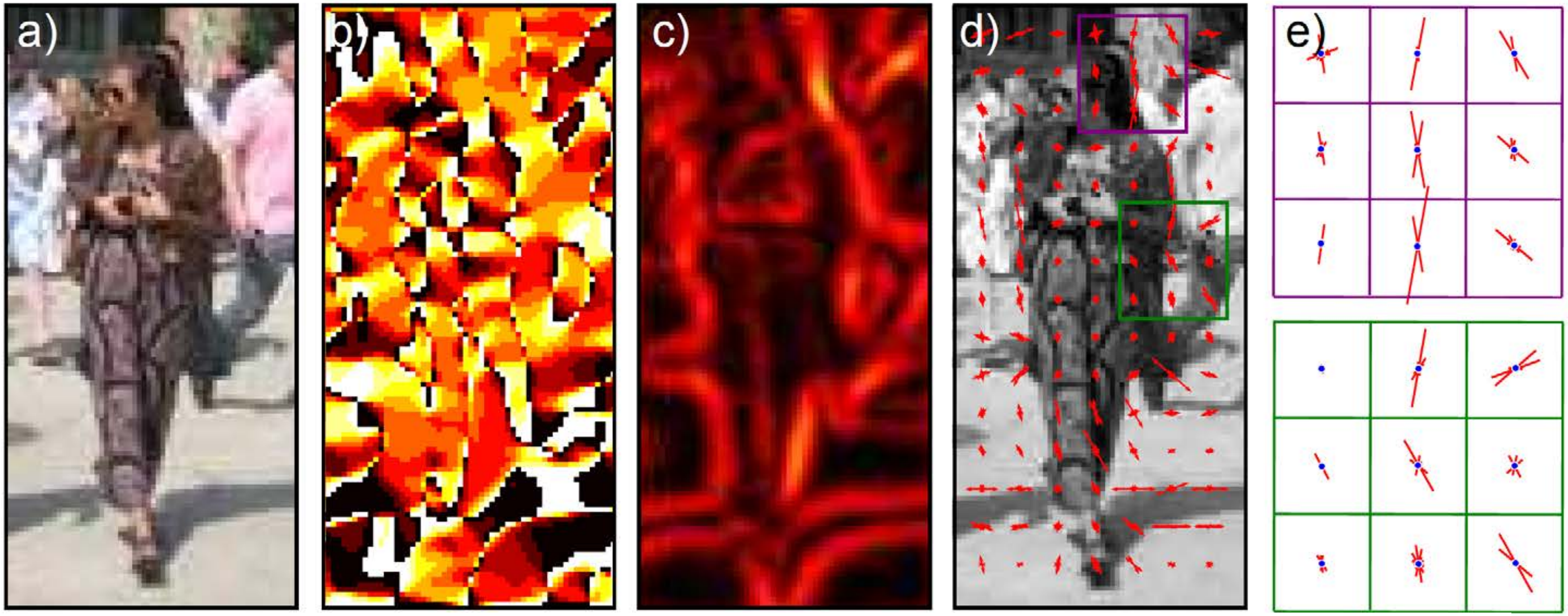
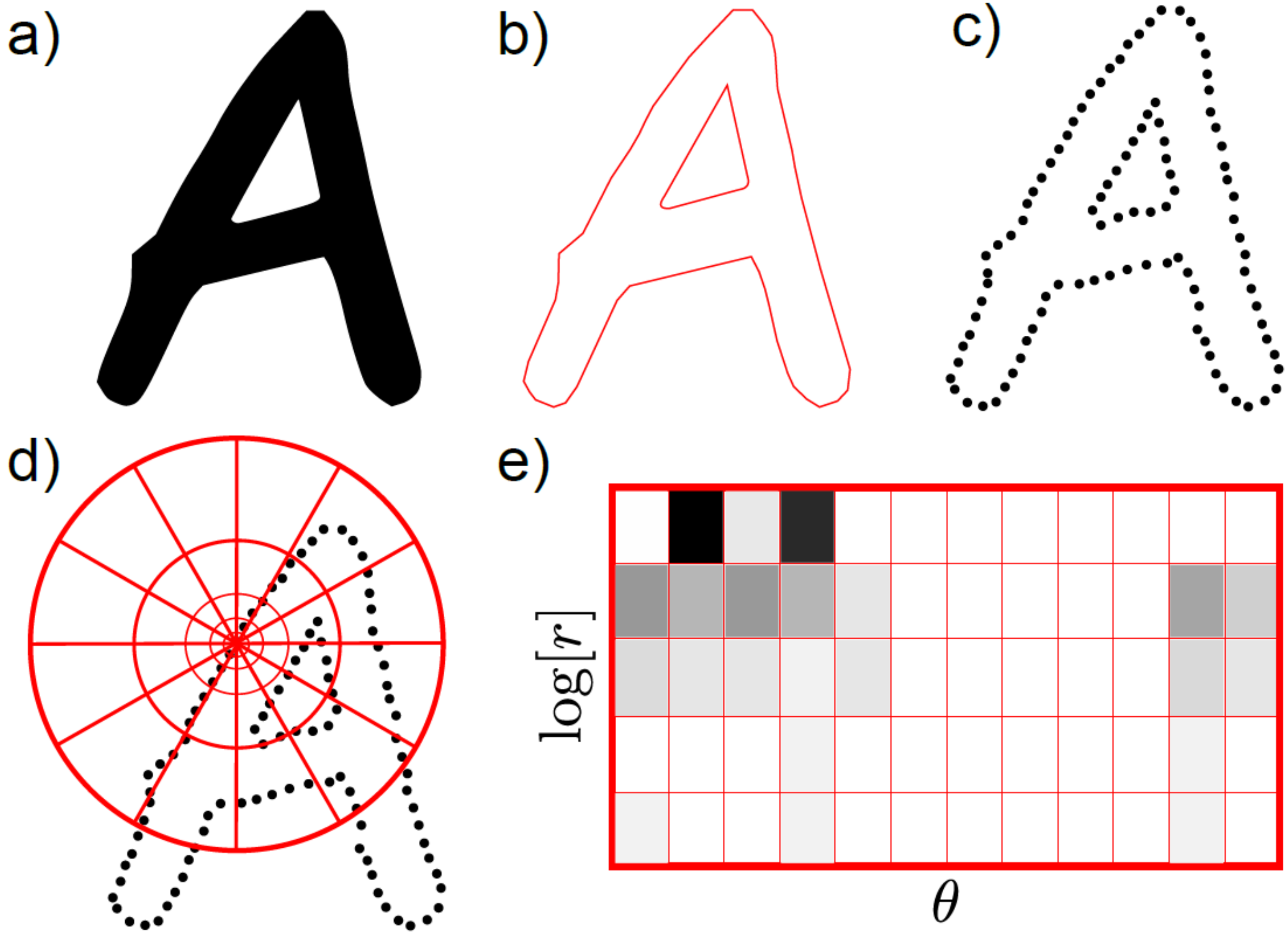


Figure 13.17 HOG descriptor. a) Original image. b) Gradient orientation, quantized into 9 bins from $0 - 180^o$. c) Gradient magnitude. d) Cell descriptors are 9D orientation histograms that are computed within 6×6 pixel regions. e) Block descriptors are computed by concatenating 3×3 blocks of cell descriptors.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

Bag of words descriptor

- Compute visual features in image
- Compute descriptor around each
- Find closest match in library and assign index
- Compute histogram of these indices over the region
- Dictionary computed using K-means

Shape context descriptor



Structure

- Per-pixel transformations
- Edges, corners, and interest points
- Descriptors
- Dimensionality reduction

Dimensionality Reduction

Dimensionality reduction attempt to find a low dimensional (or hidden) representation $\mathbf{h}$ which can approximately explain the data $\mathbf{x}$ so that

$$\mathbf{x} \approx f(\mathbf{h}, \boldsymbol{\theta})$$

where $f[\bullet, \bullet]$ is a function that takes the hidden variable and a set of parameters θ.

Typically, we choose the function family $f[\bullet, \bullet]$ and then learn $\mathbf{h}$ and θ from training data

Least Squares Criterion

$$\hat{\boldsymbol\theta}, \hat{\mathbf{h}}_{1\ldots I} = \underset{\boldsymbol\theta, \mathbf{h}_{1\ldots I}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} (\mathbf{x}_i - f[\mathbf{h}_i, \boldsymbol\theta])^T (\mathbf{x}_i - f[\mathbf{h}_i, \boldsymbol\theta]) \right]$$

Choose the parameters θ and the hidden variables $\mathbf{h}$ so that they minimize the least squares approximation error (a measure of how well they can reconstruct the data $\mathbf{x}$).

Simple Example

$$\mathbf{x}_i \approx \phi h_i + \boldsymbol{\mu}$$

Approximate each data example $\mathbf{x}$ with a scalar value h.

Data is reconstructed by multiplying h by a parameter $\boldsymbol{\phi}$ and adding the mean vector $\boldsymbol{\mu}$.

... or even better, lets subtract $\boldsymbol{\mu}$ from each data example to get mean-zero data

Simple Example

$$\mathbf{x}_i \approx \boldsymbol{\phi} h_i$$

Approximate each data example $\mathbf{x}$ with a scalar value h.

Data is reconstructed by multiplying h by a factor $\boldsymbol{\phi}$.

Criterion:

$$\hat{\boldsymbol{\phi}}, \hat{h}_{1\ldots I} = \underset{\boldsymbol{\phi}, h_{1\ldots I}}{\operatorname{argmin}} \left[E\right] = \underset{\boldsymbol{\phi}, h_{1\ldots I}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} \left(\mathbf{x}_i - \boldsymbol{\phi} h_i\right)^T \left(\mathbf{x}_i - \boldsymbol{\phi} h_i\right)\right]$$

Criterion

$$\hat{\boldsymbol\phi}, \hat{h}_{1\ldots I} = \underset{\boldsymbol\phi, h_{1\ldots I}}{\operatorname{argmin}}\left[E\right] = \underset{\boldsymbol\phi, h_{1\ldots I}}{\operatorname{argmin}}\left[\sum_{i=1}^{I}\left(\mathbf{x}_i - \boldsymbol\phi h_i\right)^T\left(\mathbf{x}_i - \boldsymbol\phi h_i\right)\right]$$

<u>Problem</u>: the problem is non-unique. If we multiply **f** by any constant α and divide each of the hidden variables $h_{1\ldots I}$ by the same constant we get the same cost. (i.e. $(f\alpha)\,(h_i/\alpha) = fh_i$)

<u>Solution</u>: We make the solution unique by constraining the length of **f** to be 1 using a Lagrange multiplier.

Criterion

Now we have the new cost function:

$$
\begin{aligned}
E &= \sum_{i=1}^{I}(\mathbf{x}_i - \phi h_i)^T(\mathbf{x}_i - \phi h_i) + \lambda(\phi^T\phi - 1) \\
&= \sum_{i=1}^{I}\mathbf{x}_i^T\mathbf{x}_i - 2h_i\phi^T\mathbf{x}_i + h_i^2 + \lambda(\phi^T\phi - 1).
\end{aligned}
$$

To optimize this we take derivatives with respect to ϕ and h_i, equate the resulting expressions to zero and re-arrange.

Solution

$$\hat{h}_i = \hat{\boldsymbol{\phi}}^T \mathbf{x}_i$$

To compute the hidden value, take dot product with the vector $\boldsymbol{\phi}$

Solution

$$\hat{h}_i = \hat{\phi}^T \mathbf{x}_i$$

To compute the hidden value, take dot product with the vector ϕ

$$\sum_{i=1}^{I} \mathbf{x}_i \mathbf{x}_i^T \hat{\phi} = \lambda \hat{\phi}$$

or

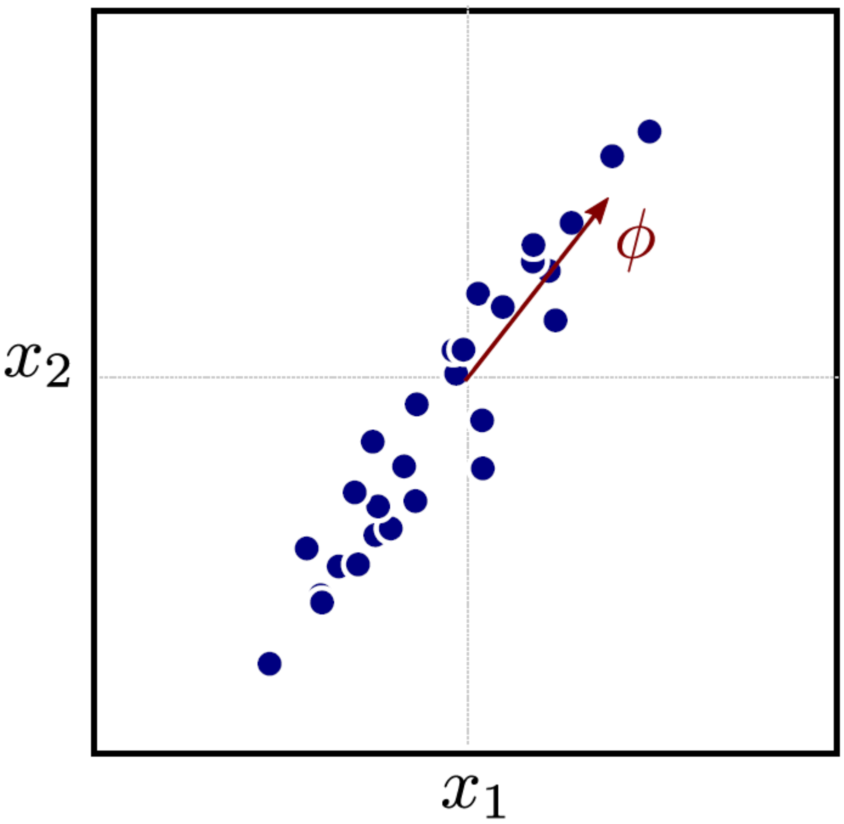
$$\mathbf{X}\mathbf{X}^T \hat{\phi} = \lambda \hat{\phi}$$

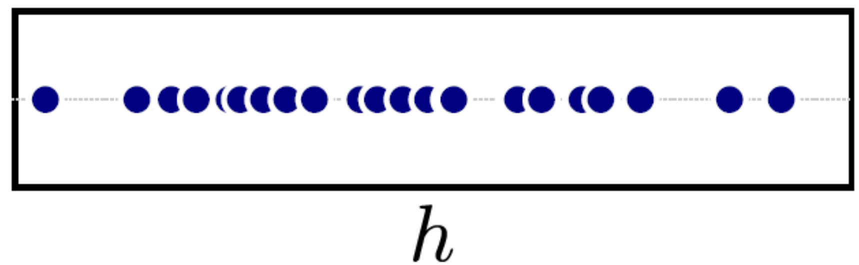
where

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_I]$$

To compute the vector ϕ, compute the first eigenvector of the scatter matrix $\mathbf{X}\mathbf{X}^T$.

Computing h

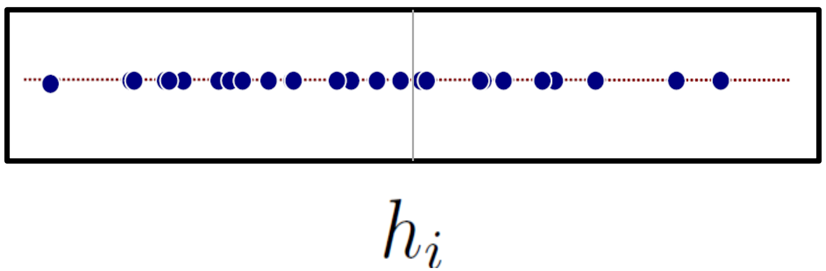


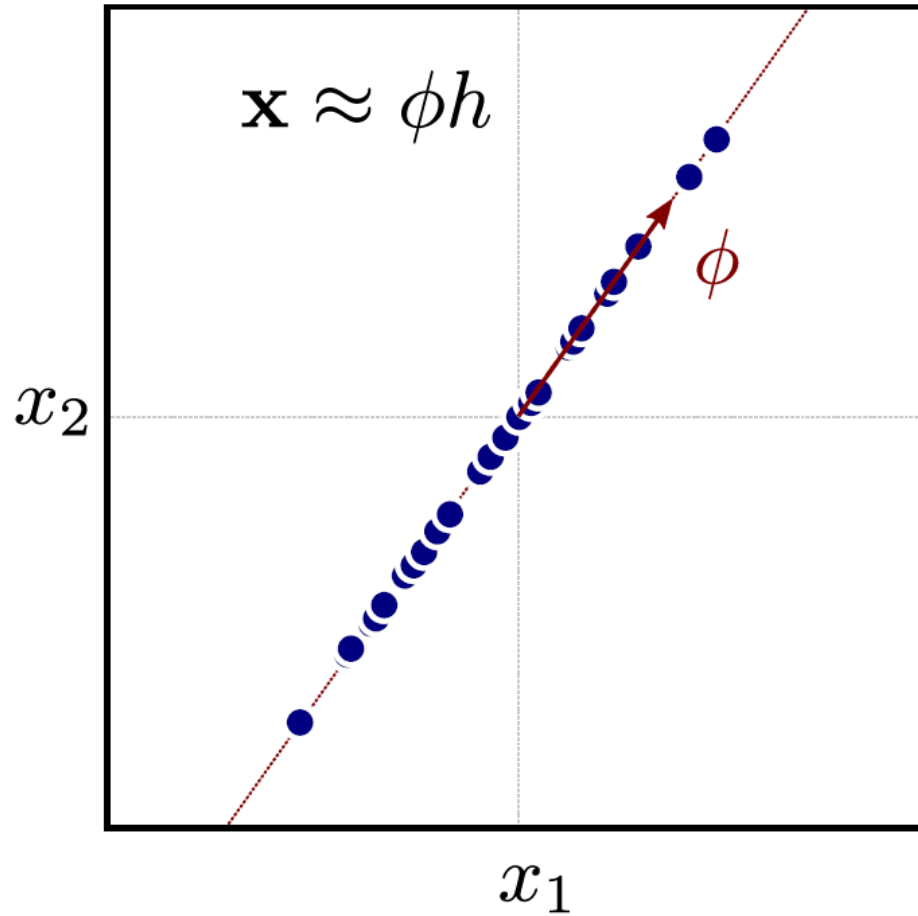


To compute the hidden value, take dot product with the vector ϕ

Reconstruction

$$\mathbf{x}_i \approx \phi h_i$$





To reconstruct, multiply the hidden variable h by vector ϕ.

Principal Components Analysis

Same idea, but not the hidden variable $\mathbf{h}$ is multi-dimensional. Each components weights one column of matrix $\mathbf{F}$ so that data is approximated as

$$\mathbf{x}_i \approx \mathbf{\Phi}\mathbf{h}_i$$

This leads to cost function:

$$\mathbf{\Phi}, \hat{\mathbf{h}}_{1\ldots I} = \underset{\mathbf{\Phi},\mathbf{h}_{1\ldots I}}{\operatorname{argmin}}\left[E\right] = \underset{\mathbf{\Phi},\mathbf{h}_{1\ldots I}}{\operatorname{argmin}}\left[\sum_{i=1}^{I}(\mathbf{x}_i - \mathbf{\Phi}\mathbf{h}_i)^T(\mathbf{x}_i - \mathbf{\Phi}\mathbf{h}_i)\right]$$

This has a non-unique optimum so we enforce the constraint that $\mathbf{F}$ should be a (truncated) rotation matrix and $\mathbf{F}^{\mathrm{T}}\mathbf{F}=\mathbf{I}$

PCA Solution

$$\mathbf{h}_i = \mathbf{\Phi}^T \mathbf{x}_i$$

To compute the hidden vector, take dot product with each column of $\mathbf{\Phi}$.

To compute the matrix $\mathbf{\Phi}$, compute the first D_h eigenvectors of the scatter matrix $\mathbf{X}\mathbf{X}^T$.

The basis functions in the columns of $\mathbf{\Phi}$ are called principal components and the entries of $\mathbf{h}$ are called loadings

Dual PCA

Problem: PCA as described has a major drawback. We need to compute the eigenvectors of the scatter matrix

$$\mathbf{X}\mathbf{X}^T$$

But this has size D_x x D_x. Visual data tends to be very high dimensional, so this may be extremely large.

Solution: Reparameterize the principal components as weighted sums of the data

$$\boldsymbol{\Phi} = \mathbf{X}\boldsymbol{\Psi}$$

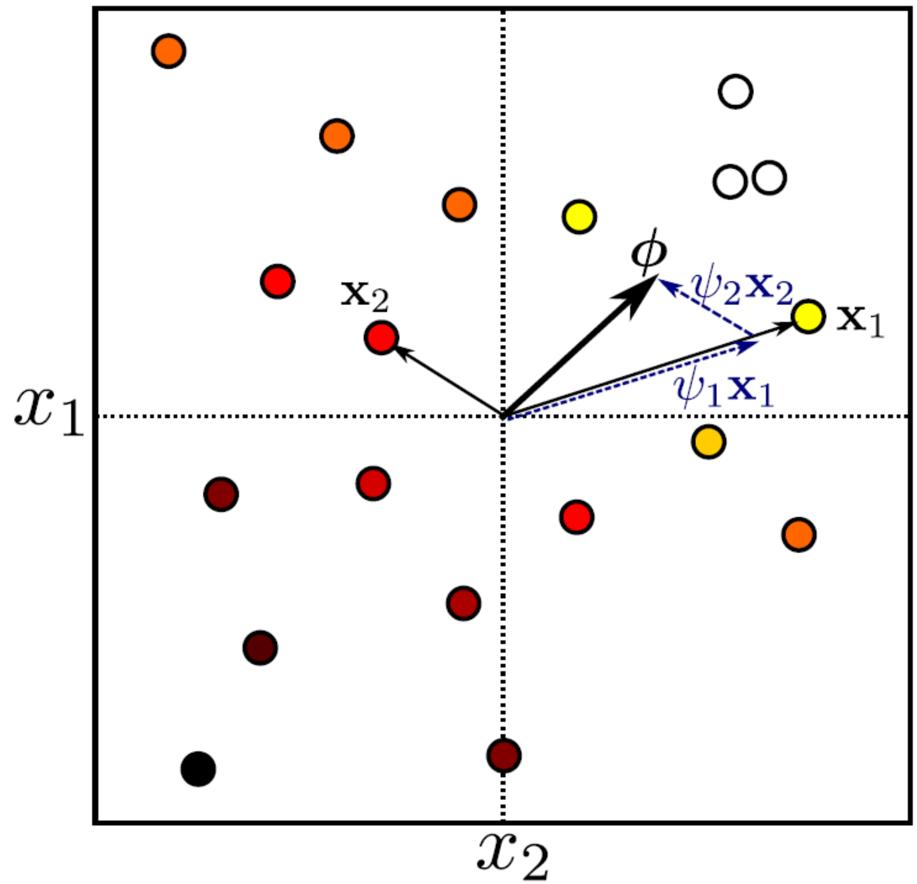
...and solve for the new variables $\boldsymbol{\Psi}$.

Geometric Interpretation

$$\Phi = \mathbf{X}\Psi$$

Each column of Φ can be described as a weighted sum of the original datapoints.

Weights given in the corresponding columns of the new variable Ψ.



Motivation

Solution: Reparameterize the principal components as weighted sums of the data

$$\mathbf{\Phi} = \mathbf{X}\mathbf{\Psi}$$

...and solve for the new variables $\mathbf{\Psi}$.

Why? If the number of datapoints I is less than the number of observed dimension D_x then the $\mathbf{\Psi}$ will be smaller than $\mathbf{\Phi}$ and the resulting optimization becomes easier.

Intuition: we are not interested in principal components that are not in the subspace spanned by the data anyway.

Cost functions

Principal components analysis

$$\hat{\boldsymbol{\Phi}, \mathbf{h}}_{1\ldots I} = \underset{\boldsymbol{\Phi},\mathbf{h}_{1\ldots I}}{\operatorname{argmin}}\left[E\right] = \underset{\boldsymbol{\Phi},\mathbf{h}_{1\ldots I}}{\operatorname{argmin}}\left[\sum_{i=1}^{I}\left(\mathbf{x}_i - \boldsymbol{\Phi}\mathbf{h}_i\right)^T\left(\mathbf{x}_i - \boldsymbol{\Phi}\mathbf{h}_i\right)\right]$$

...subject to $\boldsymbol{\Phi}^T\boldsymbol{\Phi}=\mathbf{I}$.

Dual principal components analysis

$$E = \sum_{i=1}^{I}\left(\mathbf{x}_i - \mathbf{X}\boldsymbol{\Psi}\mathbf{h}_i\right)^T\left(\mathbf{x}_i - \mathbf{X}\boldsymbol{\Psi}\mathbf{h}_i\right)$$

...subject to $\boldsymbol{\Phi}^T\boldsymbol{\Phi}=\mathbf{I}$ or $\boldsymbol{\Psi}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\Psi}=\mathbf{I}$.

Solution

$$\mathbf{h}_i = \boldsymbol{\Psi}^T \mathbf{X}^T \mathbf{x}_i = \boldsymbol{\Phi}^T \mathbf{x}_i$$

To compute the hidden vector, take dot product with each column of $\boldsymbol{\Phi}=\boldsymbol{\Psi}\mathbf{X}$.

Solution

$$\mathbf{h}_i = \mathbf{\Psi}^T \mathbf{X}^T \mathbf{x}_i = \mathbf{\Phi}^T \mathbf{x}_i$$

To compute the hidden vector, take dot product with each column of $\mathbf{\Phi}=\mathbf{\Psi}\mathbf{X}$.

To compute the matrix $\mathbf{\Psi}$, compute the first D_h eigenvectors of the inner product matrix $\mathbf{X}^T\mathbf{X}$.

The inner product matrix has size I x I.

If the number of examples I is less than the dimensionality of the data D_x then this is a smaller eigenproblem.

K-Means algorithm

Approximate data with a set of means

$$\mathbf{x}_i \approx \boldsymbol{\mu}_{h_i}$$

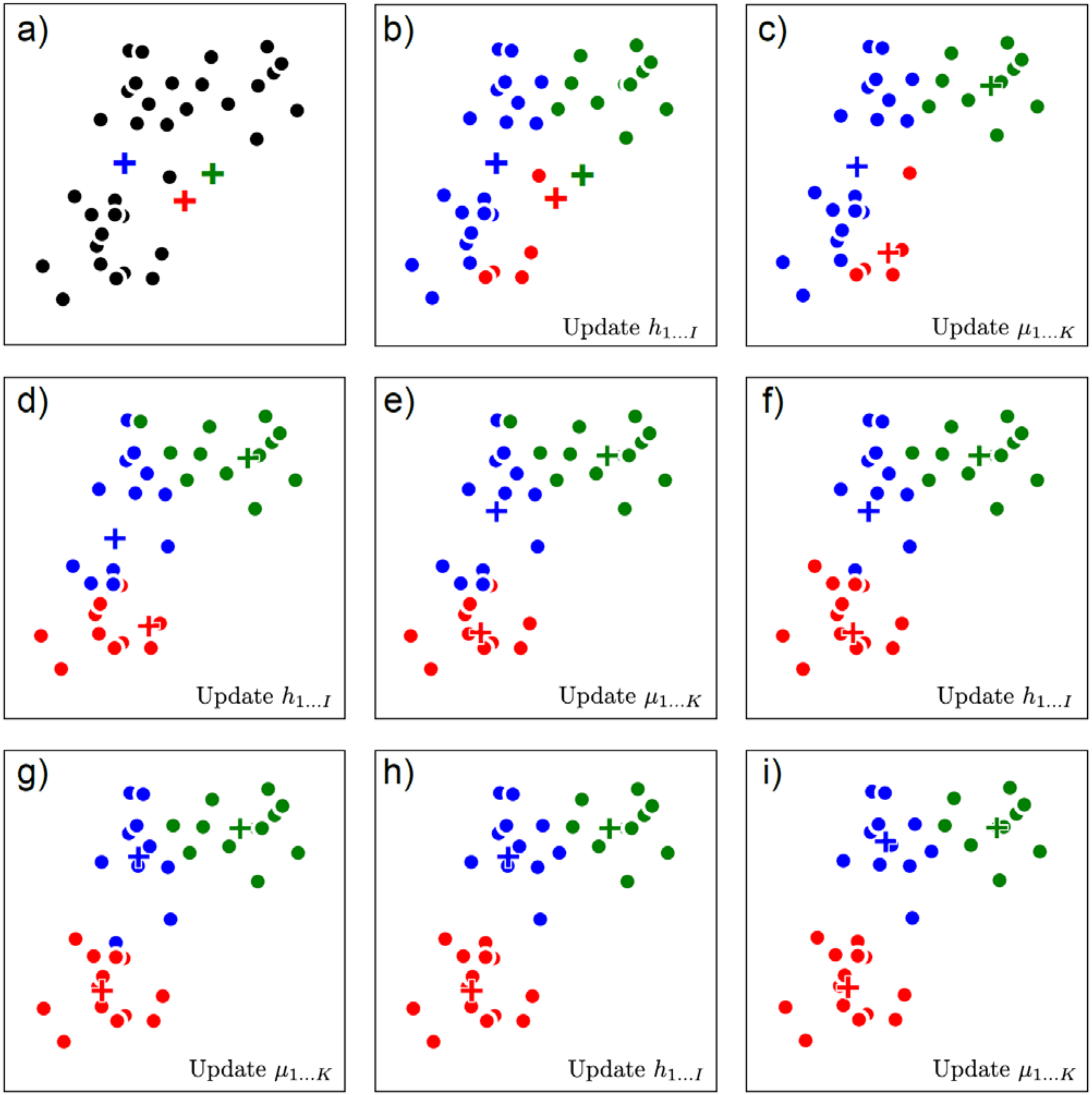
Least squares criterion

$$\hat{\boldsymbol{\mu}}_{1\ldots K}, \hat{h}_{1\ldots I} = \underset{\boldsymbol{\mu},h}{\operatorname{argmin}} \left[\sum_{i=1}^{I} \left(\mathbf{x}_i - \boldsymbol{\mu}_{h_i}\right)^T \left(\mathbf{x}_i - \boldsymbol{\mu}_{h_i}\right) \right]$$

Alternate minimization

$$\hat{h}_i = \underset{h_i}{\operatorname{argmin}} \left[\left(\mathbf{x}_i - \boldsymbol{\mu}_{h_i}\right)^T \left(\mathbf{x}_i - \boldsymbol{\mu}_{h_i}\right) \right]$$

$$\begin{aligned} \hat{\boldsymbol{\mu}}_k &= \underset{\boldsymbol{\mu}_k}{\operatorname{argmin}} \left[\sum_{i=1}^{I} \left[\left(\mathbf{x}_i - \boldsymbol{\mu}_{h_i}\right)^T \left(\mathbf{x}_i - \boldsymbol{\mu}_{h_i}\right) \right] \right] \\ &= \frac{\sum_{i=1}^{I} \mathbf{x}_i \delta[h_i - k]}{\sum_{i=1}^{I} \delta[h_i - k]}, \end{aligned}$$


a)
b)
Update $h_{1...I}$
c)
Update $\mu_{1...K}$
d)
Update $h_{1...I}$
e)
Update $\mu_{1...K}$
f)
Update $h_{1...I}$
g)
Update $\mu_{1...K}$
h)
Update $h_{1...I}$
i)
Update $\mu_{1...K}$