universität innsbruck

Lecture 4. Feature detection and matching

703142. Computer Vision

Assoz.Prof. Antonio Rodríguez-Sánchez, PhD.

Outline

- Introduction
- Point and patches
- Edges
- Corners

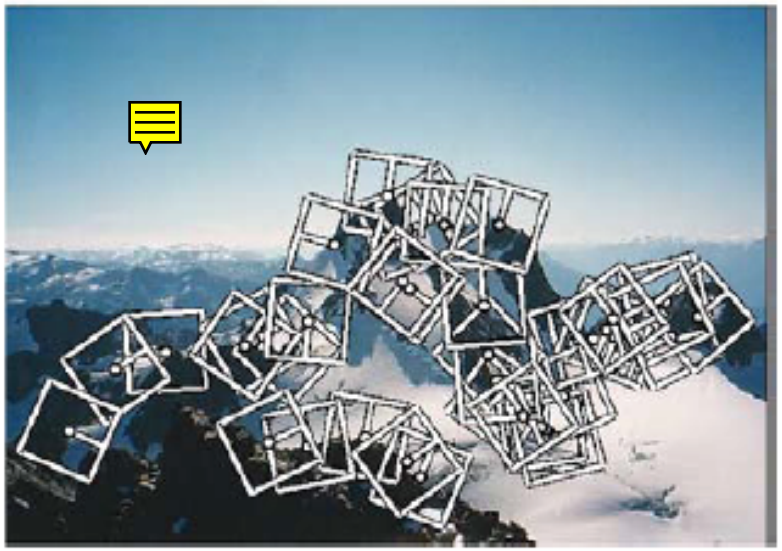
Introduction

- What features to use for matching?

Introduction

- What features to use for matching?

Introduction



(a)

(b)

(c)

(d)

Introduction

- Feature detection and matching are essential in many computer vision applications
- We want to establish a set of *correspondences*
  - A 3D model can be constructed
  - An in-between view can be generated

Introduction

- Feature detection and matching are essential in many computer vision applications
- We want to establish a set of *correspondences*
  - A 3D model can be constructed
  - An in-between view can be generated

What features do we select?

Introduction

- Information content in images can be easily localized
  - Abrupt changes in image radiance: edges, corners, …
  - Configurations of intensity changes corresponding to simple patterns: extended lines, circles
- Image **features** are local, meaningful and detectable parts of an image
  - Local implies limited spatial support
  - Meaningful implies that they can be of use of subsequent operations
  - Detectable implies that we can develop an algorithm for extracting the position and description of these entities given image data

Outline

- Introduction
- Point and patches
- Edges
- Corners

Points and patches

- *Keypoints* are used to perform object instance and category recognition
  - They permit matching even in the presence of
    - Occlusion
    - Large scale
    - Orientation changes
  - Its origin comes from stereo and are used presently for image-stitching applications

Points and patches

- Two main approaches to finding feature points and their correspondences
  - Find features in one image that can be accurately tracked using local search techniques
    - i.e. correlation
    - Least squares
  - Detect features and then match features based on their local appearance

Points and patches

4 stages:

Feature detectors

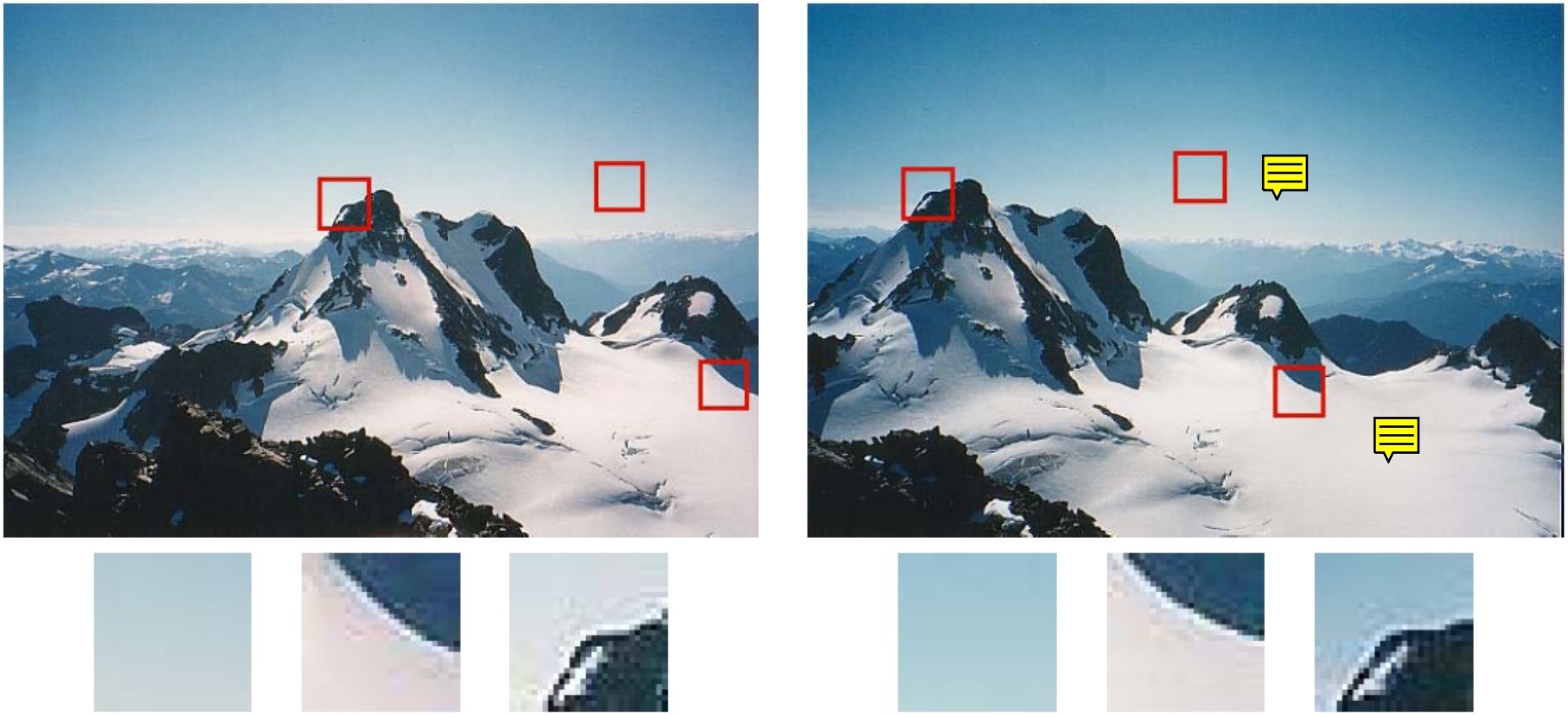
Feature descriptors

Feature matching

Feature tracking

Feature detectors

- What are the *good* features?

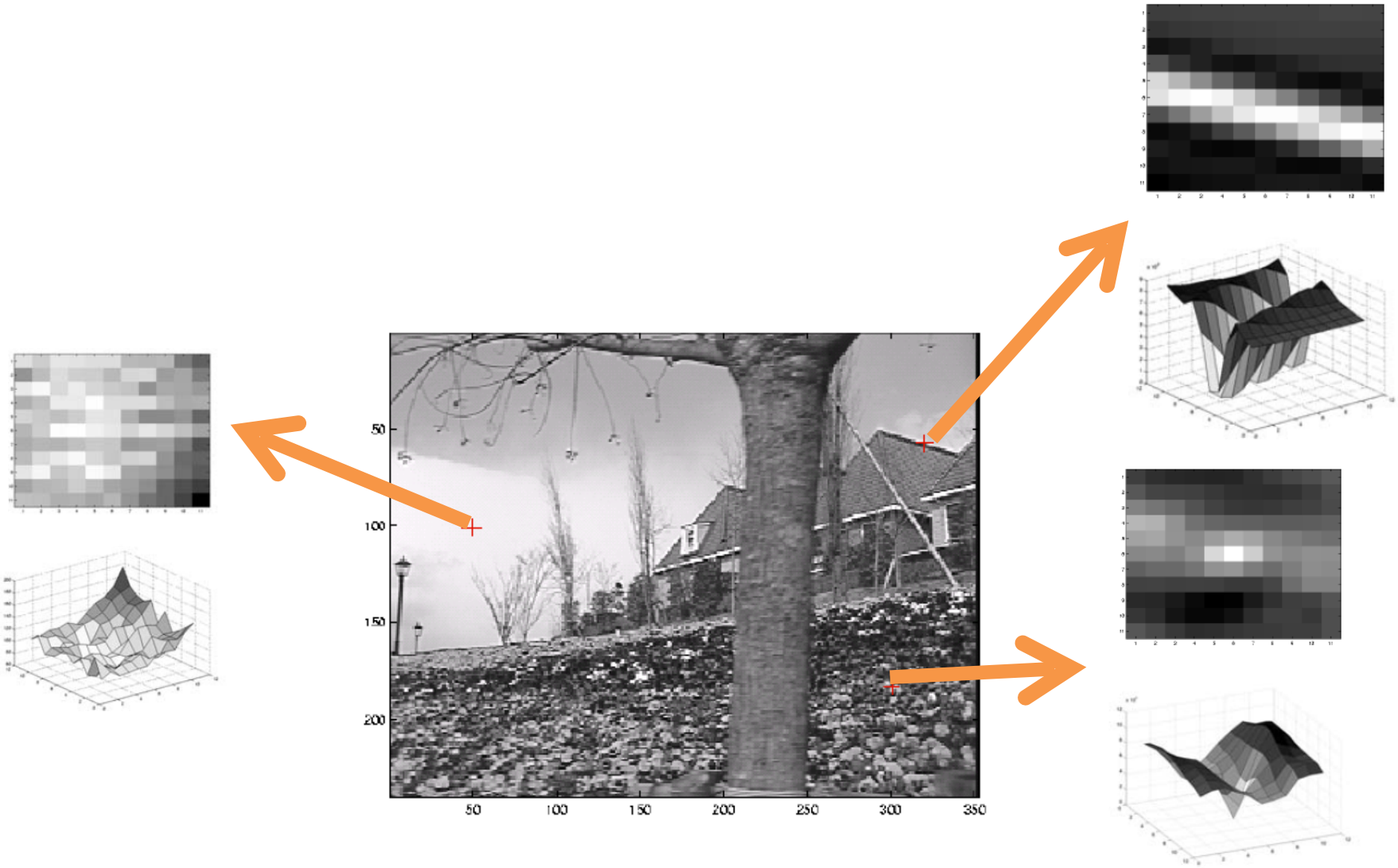


Feature descriptors

Feature descriptors

Feature descriptors

Feature descriptors

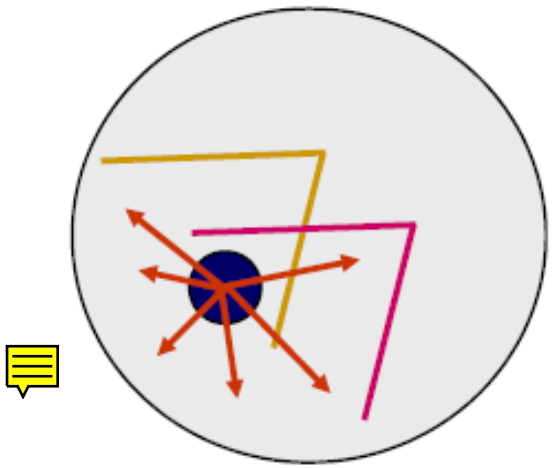


Feature detectors

- What are the *good* features?

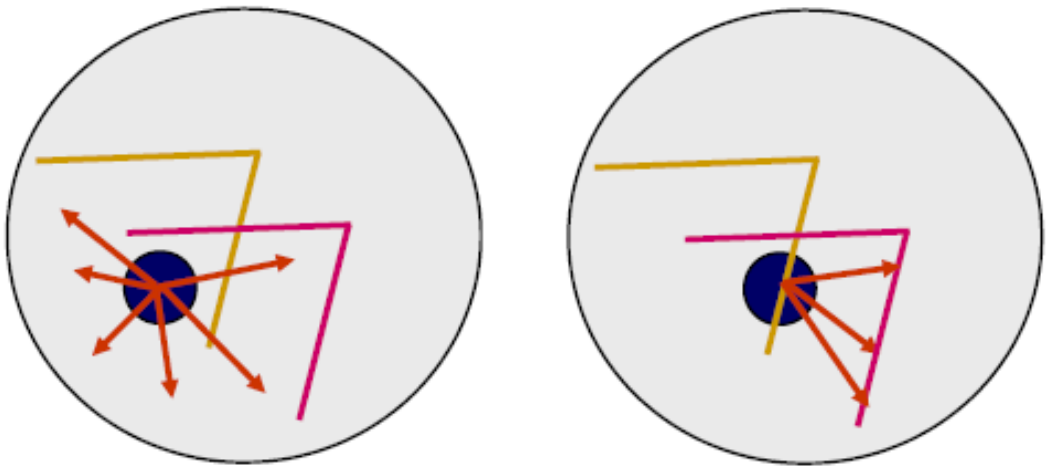
Feature detectors

- What are the *good* features?
  - Textureless patches are nearly impossible to localize



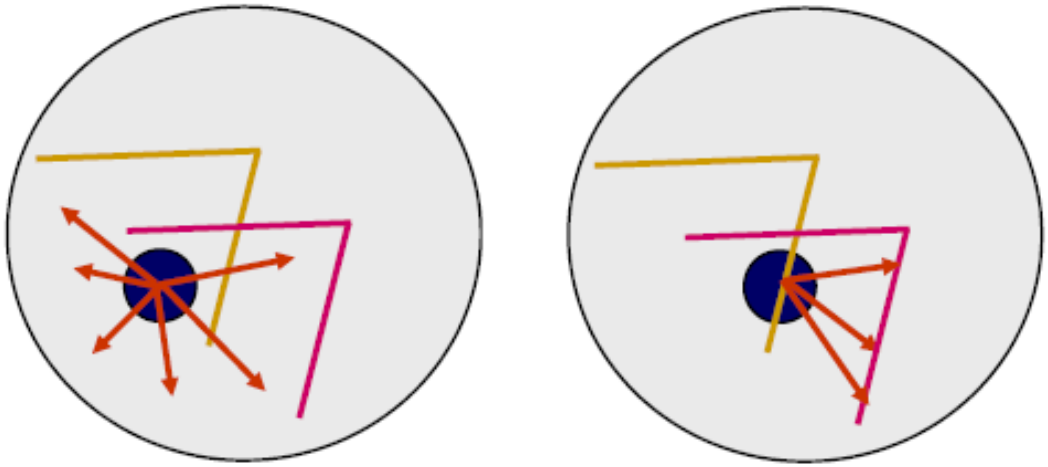
Feature detectors

- What are the *good* features?
  - Textureless patches are nearly impossible to localize
  - Patches with large contrast changes are easier to localize



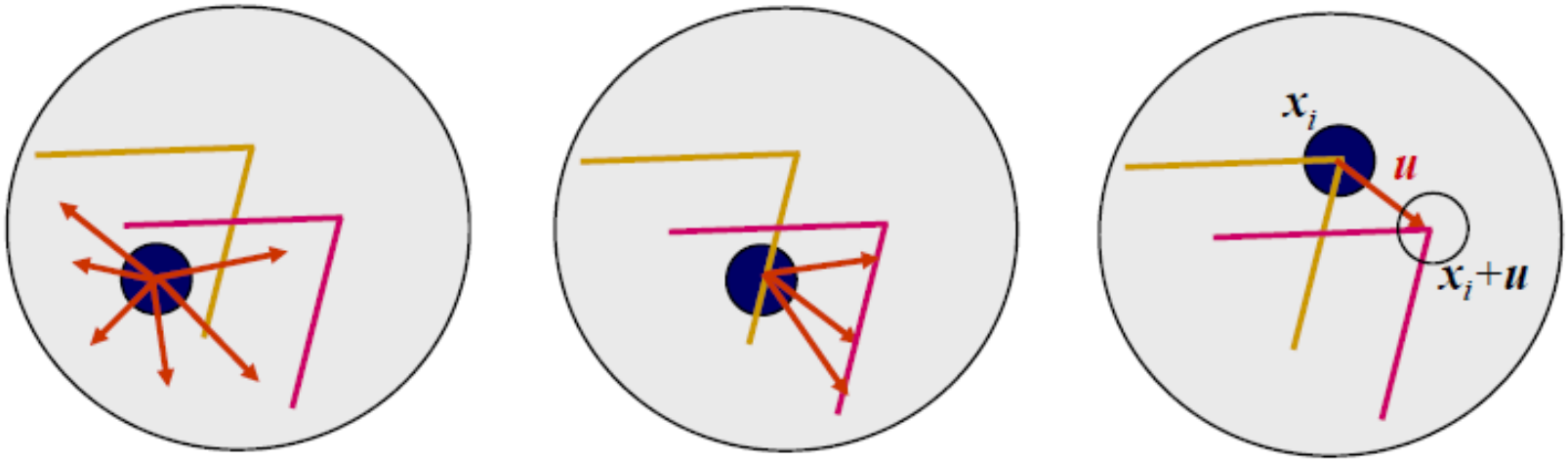
Feature detectors

- What are the *good* features?
  - Textureless patches are nearly impossible to localize
  - Patches with large contrast changes are easier to localize
  - But lines suffer of the *aperture problem*



Feature detectors

- What are the *good* features?
  - Textureless patches are nearly impossible to localize
  - Patches with large contrast changes are easier to localize
  - But lines suffer of the *aperture problem*
  - Two orientations better than one



Feature detectors

- Local appearance of features will change in
  - Orientation
  - Scale
  - Affine deformations
  - Intensity
  - …

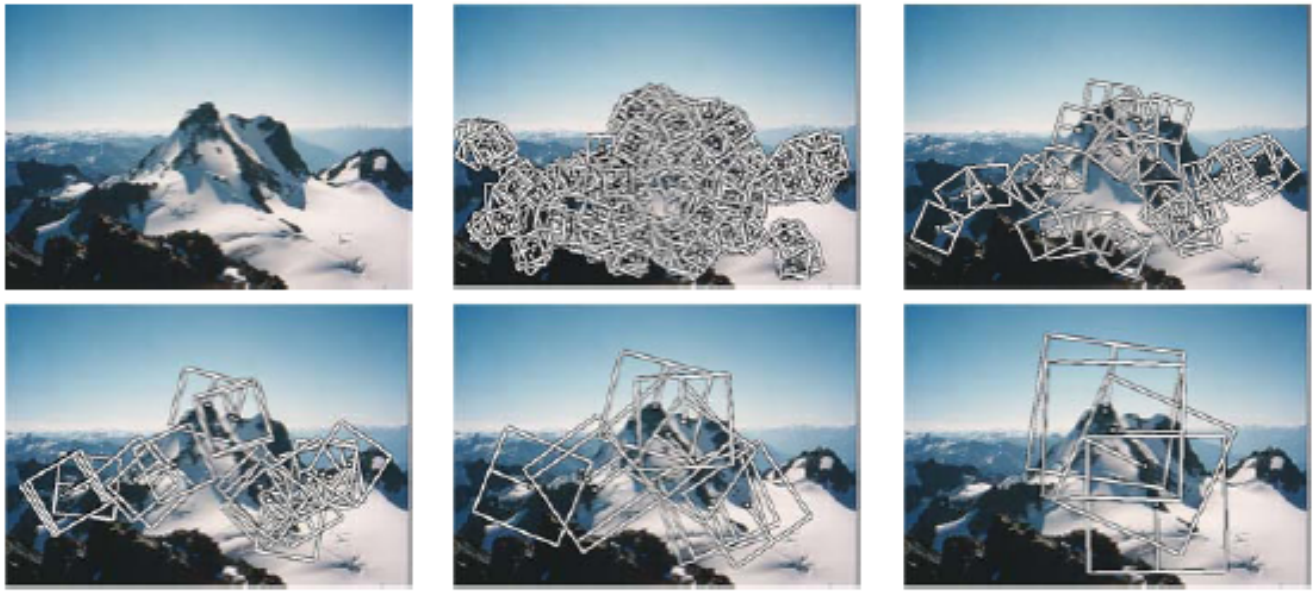
Feature detectors

- Interesting characteristics
  - Repeatibility
    - Frequency with which keypoints are found
  - Scale invariance
    - Extract features at a variety of scales (i.e. pyramid) or
    - Extract features that are stable in scale
  - Rotation invariance and orientation estimation
    - Design descriptors rotationally invariant or
    - Estimate a dominant orientation at each keypoint
      - Once the local orientation and scale keypoint have been estimated, a scaled and oriented patch around the detected point can be extracted
      - We can use steerable filters
  - Affine invariance (deformations)

Feature detectors

- Scale invariance

1. Extract features at a variety of scales



E.g. MOPS descriptors extracted at five pyramid levels

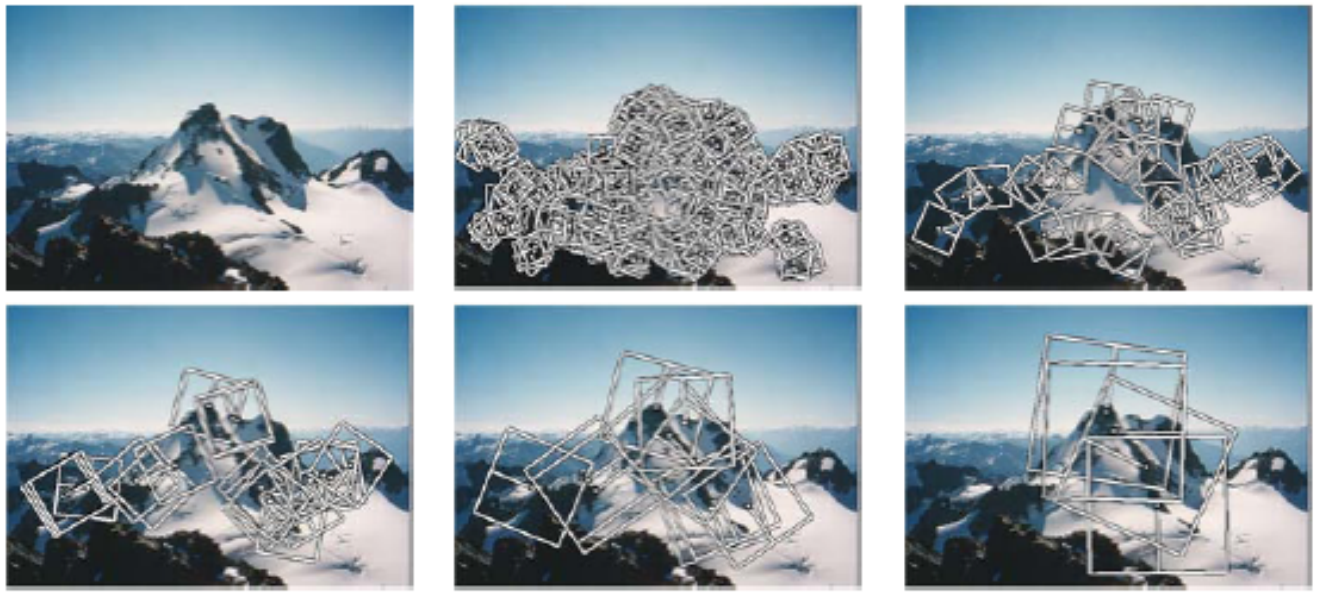
2. Extract features stable at location and scale (e.g. SIFT)

Feature detectors

- Rotation invariance and orientation estimation
  1. Design descriptors that are rotationally invariant
  2. Estimate a dominant orientation

     Average gradient within a region around the keypoint

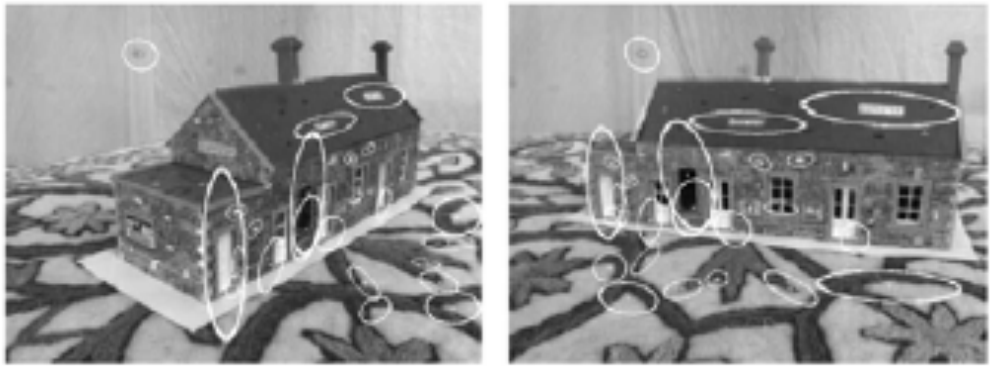
     Histogram of orientation



E.g. MOPS descriptors extracted at five pyramid levels

Feature detectors

- Affine invariance
  - Fit an ellipse to the autocorrelation or Hessian matrix



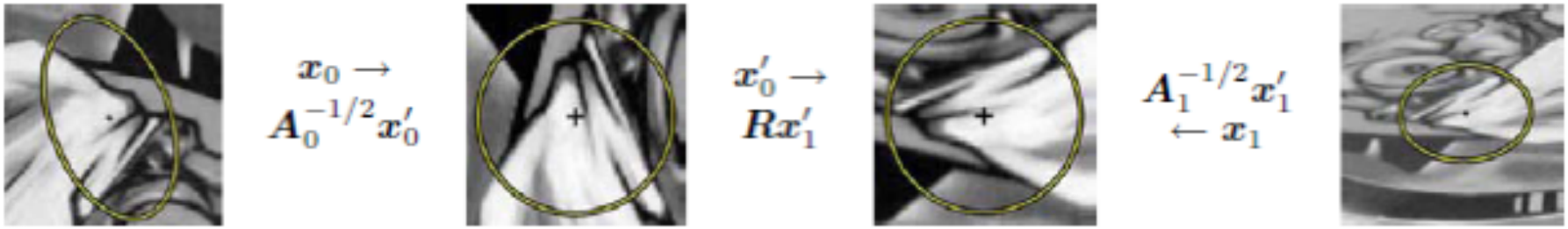


Figure 4.14 Affine normalization using the second moment matrices, as described by Mikolajczyk, Tuytelaars, Schmid *et al.* (2005) © 2005 Springer. After image coordinates are transformed using the matrices $A_0^{-1/2}$ and $A_1^{-1/2}$, they are related by a pure rotation R, which can be estimated using a dominant orientation technique.

Feature descriptors

- Affine invariance
  - Bias and gain normalization (MOPS)
    - Brown, Szeliski and Winder (2005)
  - Scale invariant feature transform (SIFT)
  - PCA-SIFT
  - Gradient location-orientation histogram (GLOH)
    - Mikolajczyk and Schmid (2005)
  - Steerable filters
    - Freeman and Adelson (1991)

Feature matching

- After detecting features, we must match them
  - We need to define a metric

Feature matching

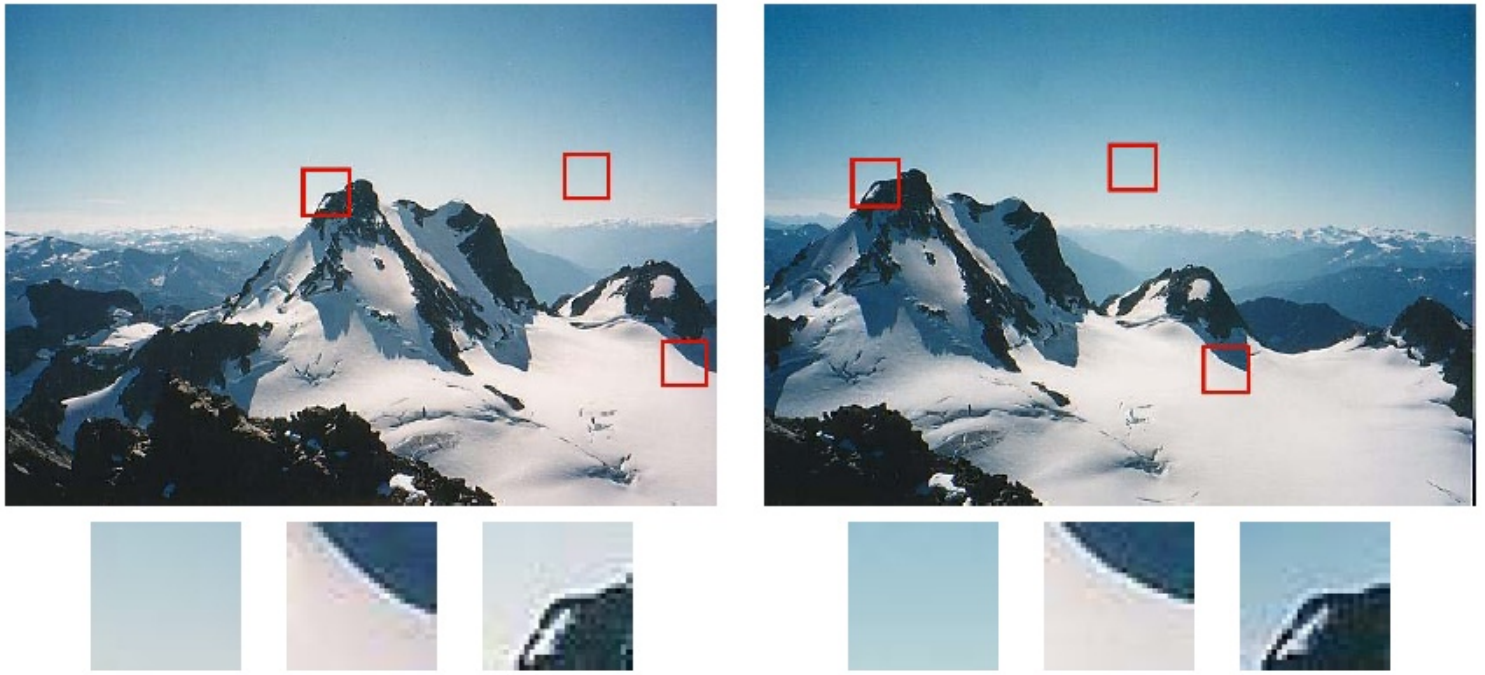
- Matching criterion
  - E.g. Weighted summed square difference

$$E_{\mathrm{WSSD}}(\boldsymbol{u}) = \sum_i w(\boldsymbol{x}_i)[I_1(\boldsymbol{x}_i + \boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2,$$

Feature matching

- Matching criterion
  - E.g. Weighted summed square difference

$$E_{\mathrm{WSSD}}(\boldsymbol{u}) = \sum_i w(\boldsymbol{x}_i)[I_1(\boldsymbol{x}_i + \boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2,$$



Feature matching

- Matching criterion
  - E.g. Weighted summed square difference

$$E_{\mathrm{WSSD}}(\boldsymbol{u}) = \sum_i w(\boldsymbol{x}_i)[I_1(\boldsymbol{x}_i + \boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2,$$

  - But there are many other measures, sometimes depending on the task
  - Indexing to make matching faster

Feature matching

- Matching criterion
  - We need also some measure of how “good” is our method for matching

Feature matching

- Matching criterion
  - We need also some measure of how “good” is our method for matching
    - TP: True positives, number of correct matches
    - FN: False negatives, matches not correctly detected
    - FP: False positives, proposed matches that are incorrect
    - TN: True negatives, not-matches that were correctly detected
  - We usually create a *confusion matrix*.

Feature matching

- Matching criterion
  - We usually create a *confusion matrix*.

	Cylinder	Circular	Box	Paper	Wrench
Cylinder	80%	0	0	20%	0
Circular	0	100%	0	0	0
Box	0	0	68.68%	31.43%	0
Paper	6.67%	0	16.67%	70%	6.67%
Wrench	0	0	6.67%	0	93.33%

Feature matching

- Matching criterion
  - We usually create a *confusion matrix*.

	Cylinder	Circular	Box	Paper	Wrench
Cylinder	80%	0	0	20%	0
Circular	0	100%	0	0	0
Box	0	0	68.68%	31.43%	0
Paper	6.67%	0	16.67%	70%	6.67%
Wrench	0	0	6.67%	0	93.33%

  - TP=82.4%

Feature matching

- Matching criterion

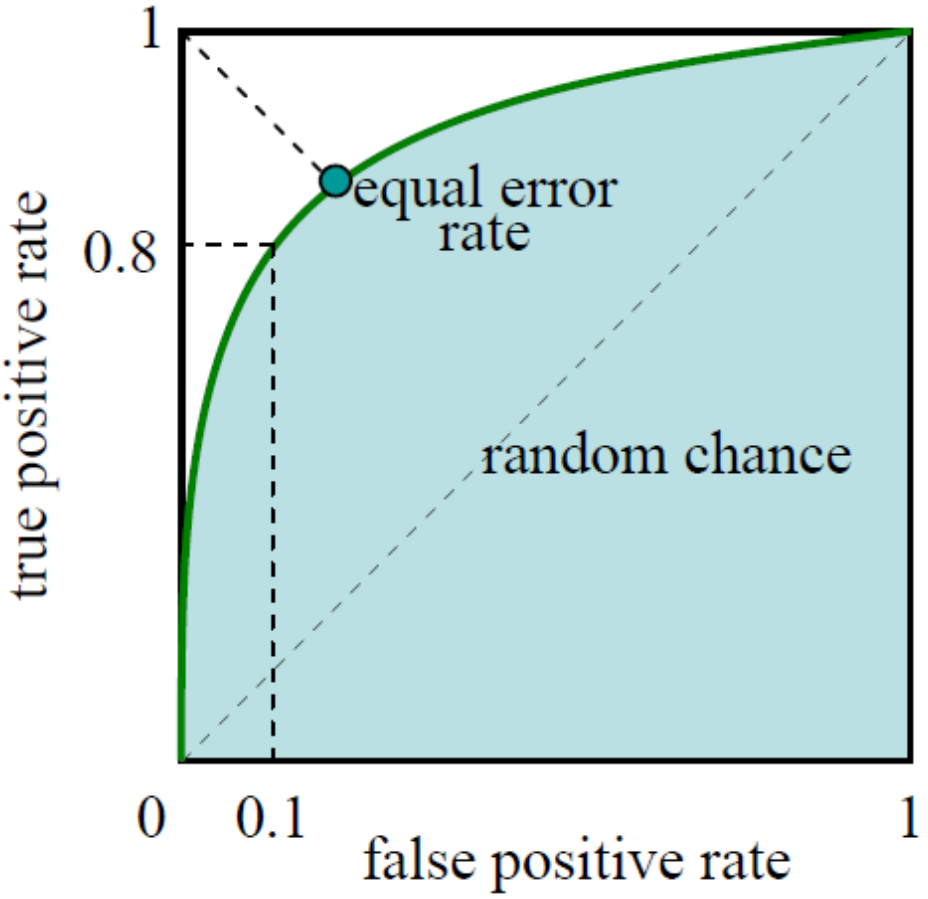
	True matches	True non-matches	
Predicted matches	TP = 18	FP = 4	P' = 22
Predicted non-matches	FN = 2	TN = 76	N' = 78
	P = 20	N = 80	Total = 100

TPR = 0.90	FPR = 0.05

  - True Positive Rate (TPR) $TPR = \frac{TP}{TP + FN}$
  - False Positive Rate (FPR) $FPR = \frac{FP}{FP + TN}$

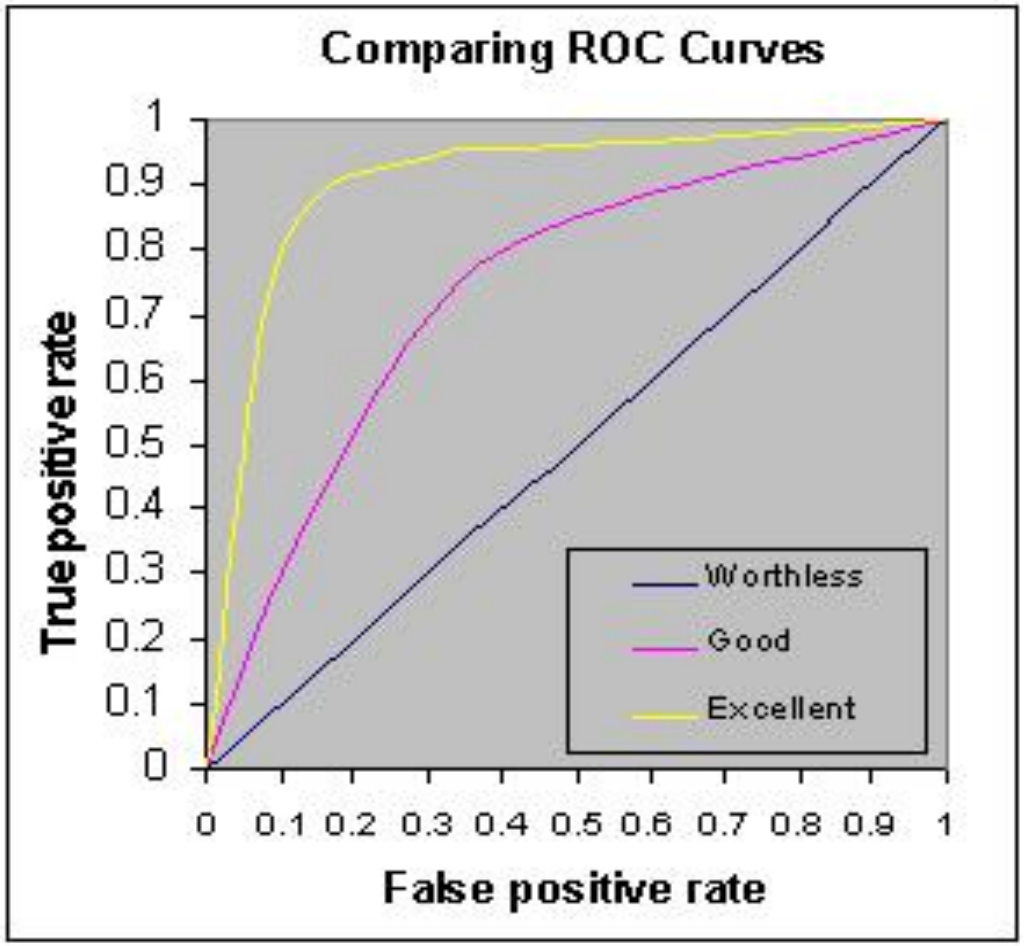
Feature matching

- Matching criterion
  - ROC curve



Feature matching

- Matching criterion
  - ROC curve



Feature matching

- Matching criterion

	True matches	True non-matches	
Predicted matches	TP = 18	FP = 4	P' = 22
Predicted non-matches	FN = 2	TN = 76	N' = 78
	P = 20	N = 80	Total = 100

TPR = 0.90	FPR = 0.05

– TPR $TPR = \frac{TP}{TP + FN}$

– FPR $FPR = \frac{FP}{FP + TN}$

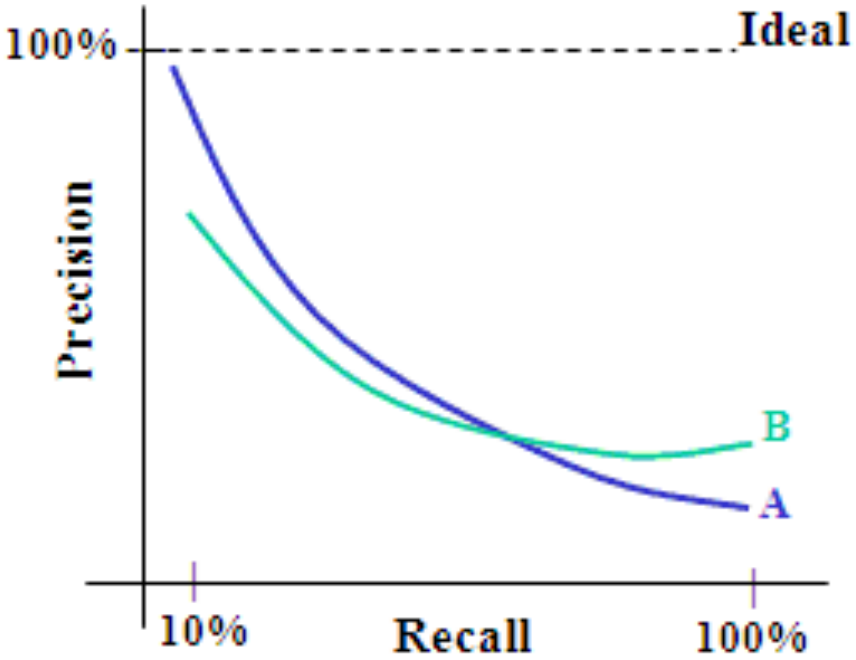
$\text{Pr}\,ecision = \frac{TP}{TP + FP}$

$\text{Re}\,call = \frac{TP}{TP + FN}$

$Accuracy = \frac{TP + TN}{TP + FN + FP + FN}$

Feature matching

- Matching criterion



Feature tracking

- Feature tracking
  - Find a set of likely feature locations and then search for locations in subsequent images
  - Detect then track: used in video applications

Feature tracking

- Feature tracking
  - Find a set of likely feature locations and then search for locations in subsequent images
  - Detect then track: used in video applications
  - Example: Facial puppetry

Case study: SIFT

- David G. Lowe, **"Object recognition from local scale-invariant features,"** *International Conference on Computer Vision,* Corfu, Greece (September 1999), pp. 1150-1157
- David G. Lowe, **"Local feature view clustering for 3D object recognition,"** *IEEE Conference on Computer Vision and Pattern Recognition,* Kauai, Hawaii (December 2001), pp. 682-688
- David G. Lowe, **"Distinctive image features from scale-invariant keypoints,"** *International Journal of Computer Vision,* 60, 2 (2004), pp. 91-110
- http://www.cs.ubc.ca/~lowe/keypoints/

Case study: SIFT

1. Scale-space extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor

Case study: SIFT

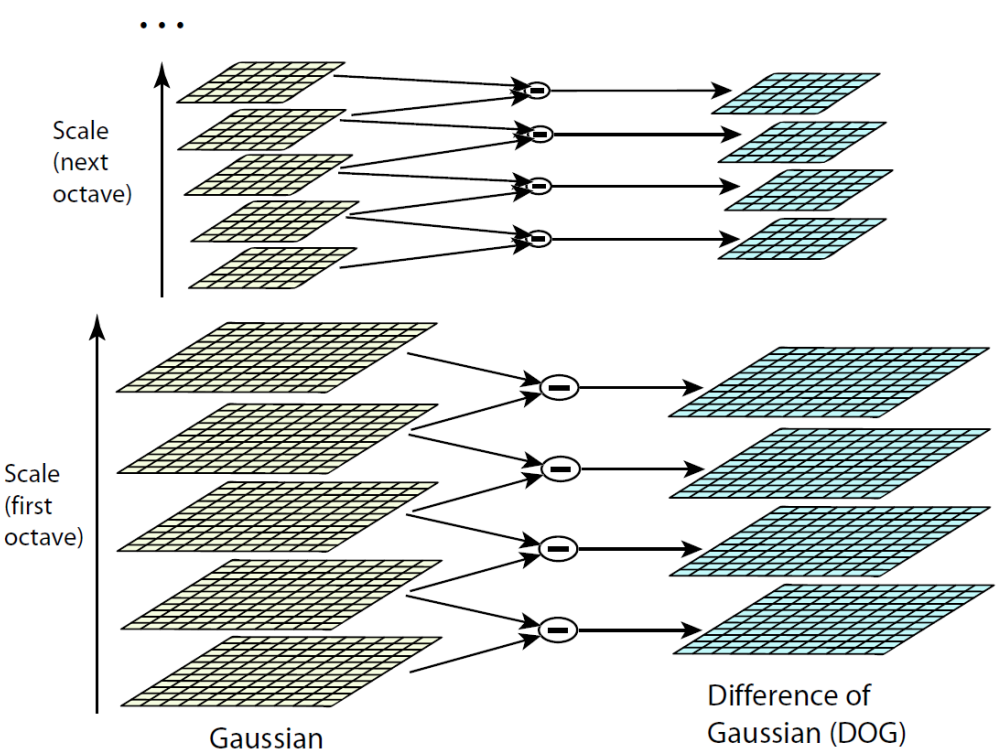
- Scale-space extrema detection
  - Identify locations and scales that can be repeatably assigned under differing views of the same objects
  - Search feature across all possible scales
    - Using difference of Gaussian function of two nearby scales separated by a factor ***k***

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned}$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

Case study: SIFT

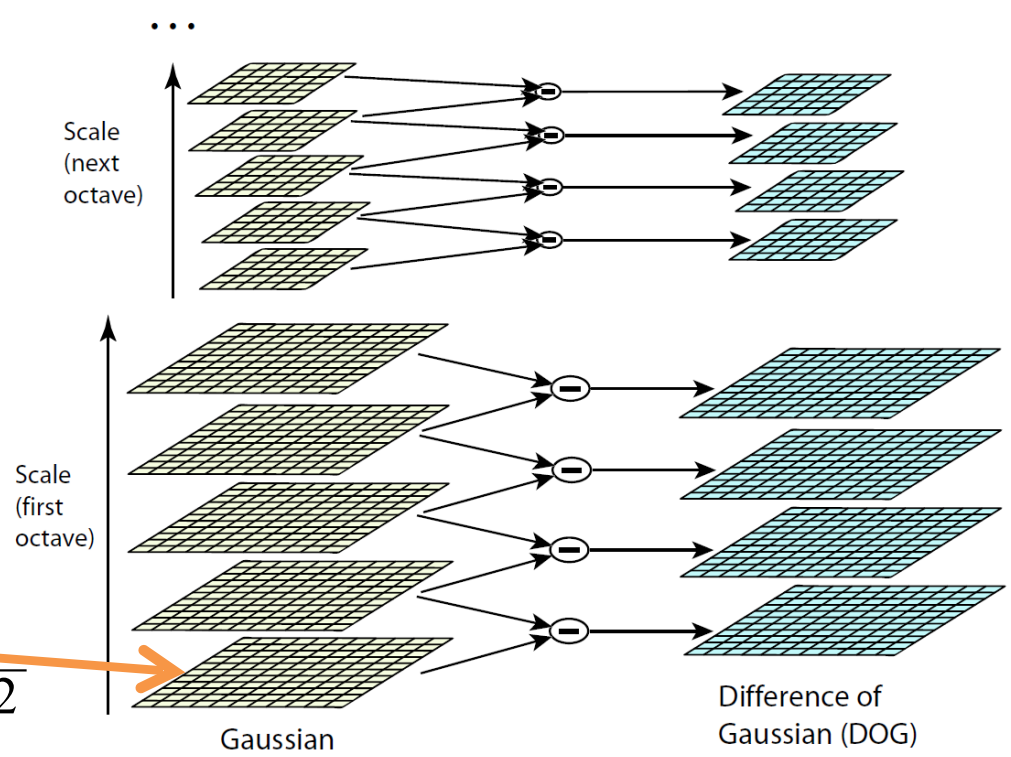
- Scale-space extrema detection
  - Search feature across all possible scales
    - Using difference of Gaussian function of two nearby scales separated by a factor k



$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned}$$

Case study: SIFT

- Scale-space extrema detection
  - Search feature across all possible scales
    - Using difference of Gaussian function of two nearby scales separated by a factor k

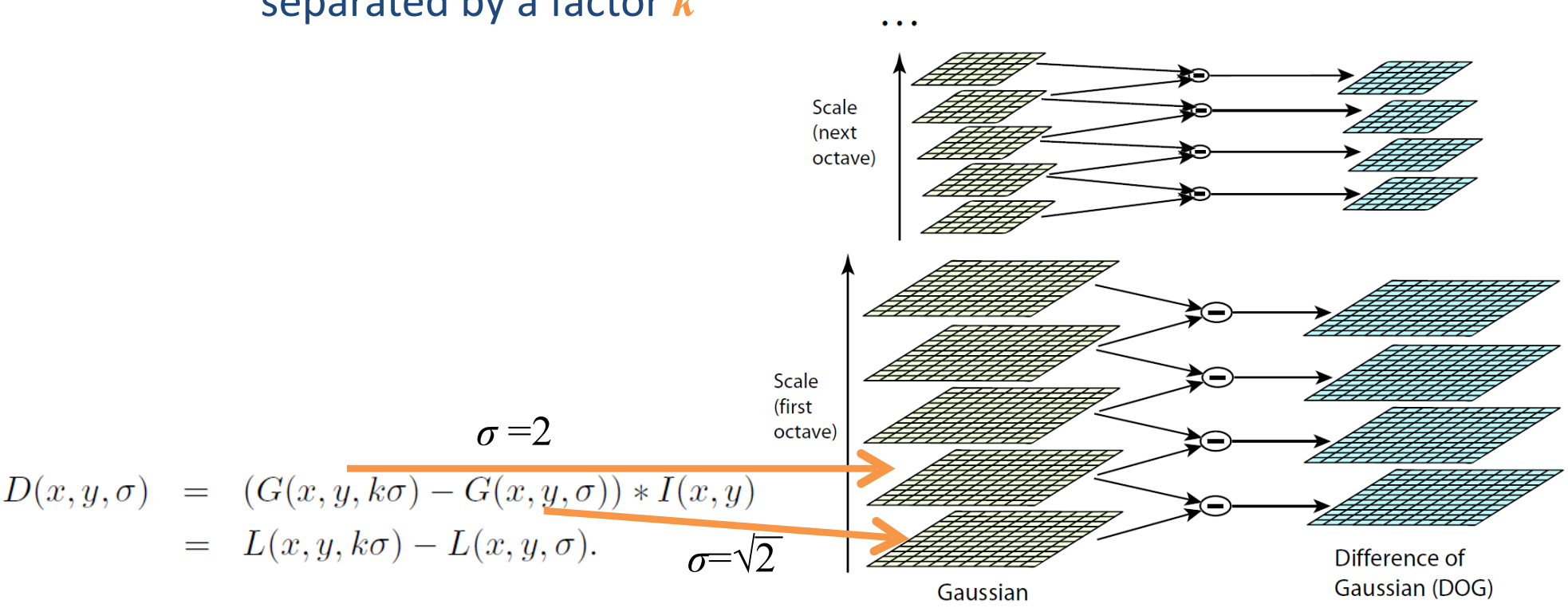


$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y)$$
$$= L(x,y,k\sigma) - L(x,y,\sigma).$$

$\sigma=\sqrt{2}$

Case study: SIFT

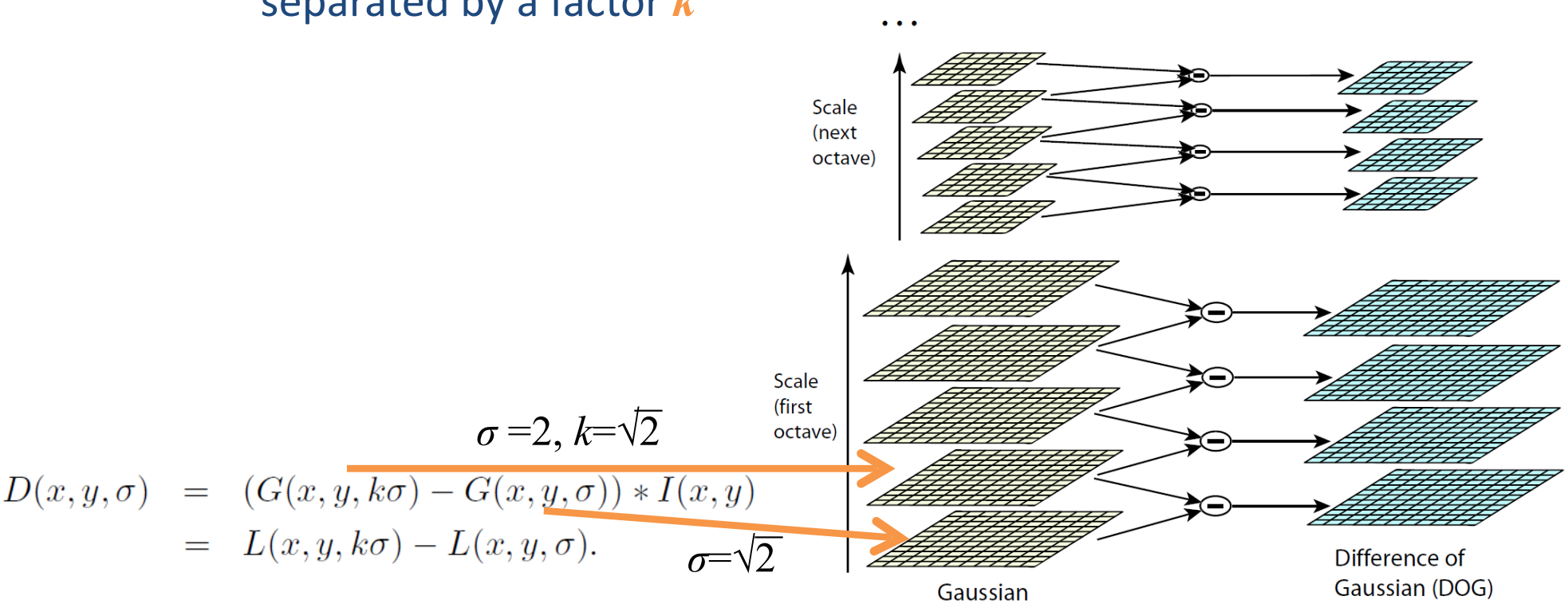
- Scale-space extrema detection
  - Search feature across all possible scales
    - Using difference of Gaussian function of two nearby scales separated by a factor k



$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$$
$$= L(x, y, k\sigma) - L(x, y, \sigma).$$

Case study: SIFT

- Scale-space extrema detection
  - Search feature across all possible scales
    - Using difference of Gaussian function of two nearby scales separated by a factor k



$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$$
$$= L(x, y, k\sigma) - L(x, y, \sigma).$$

Case study: SIFT

- Scale-space extrema detection
  - Search feature across all possible scales
    - Using difference of Gaussian function of two nearby scales separated by a factor k
    - Note: This is similar to normalizing the Laplacian by σ^2

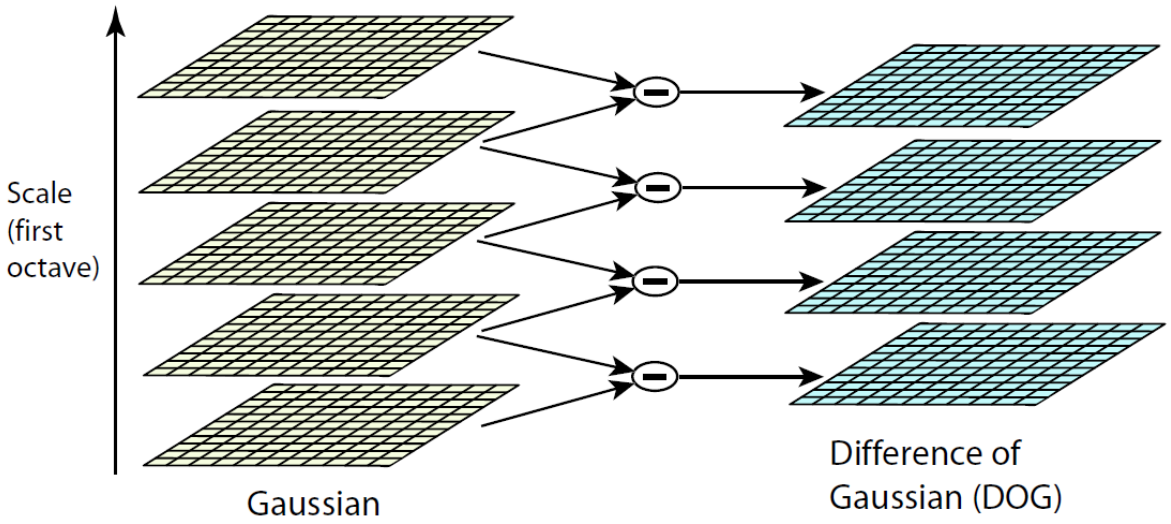
$$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G. \Rightarrow G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G.$$

Case study: SIFT

- Scale-space extrema detection
  - Search feature across all possible scales
    - Using difference of Gaussian function of two nearby scales separated by a factor k
    - Then, maxima and minima are detected by comparing a pixel with its 26 neighbors in 3x3 regions at the current and adjacent scales

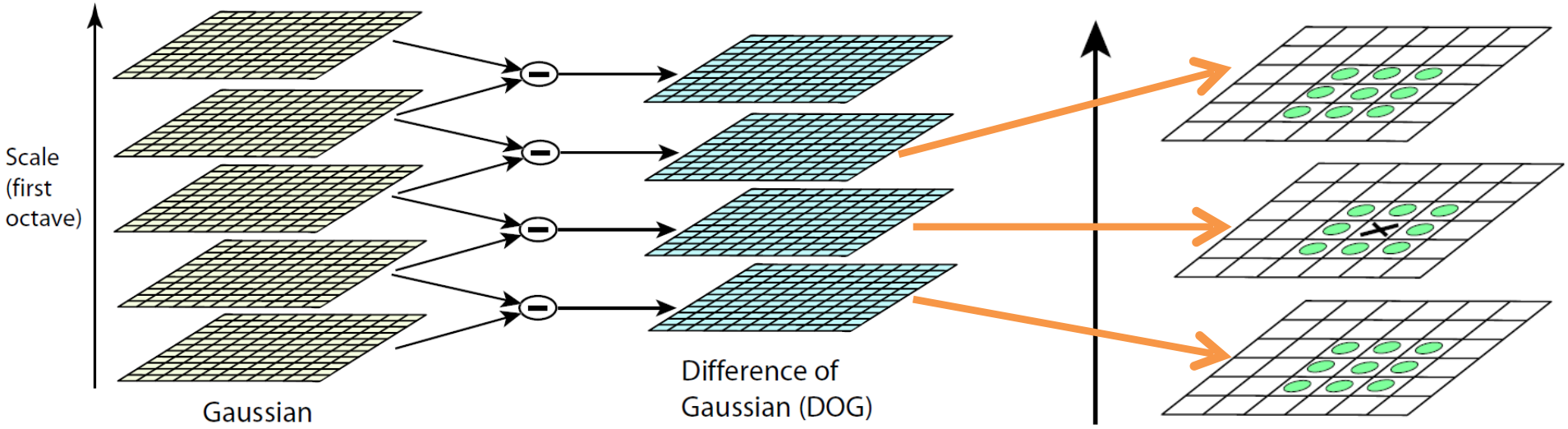
Case study: SIFT

- Scale-space extrema detection
  - Search feature across all possible scales
    - Using difference of Gaussian function of two nearby scales separated by a factor k
    - Then, maxima and minima are detected by comparing a pixel with its 26 neighbors in 3x3 regions at the current and adjacent scales



Case study: SIFT

- Scale-space extrema detection
  - Search feature across all possible scales
    - Using difference of Gaussian function of two nearby scales separated by a factor k
    - Then, maxima and minima are detected by comparing a pixel with its 26 neighbors in 3x3 regions at the current and adjacent scales



Case study: SIFT

1. Scale-space extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor

Case study: SIFT

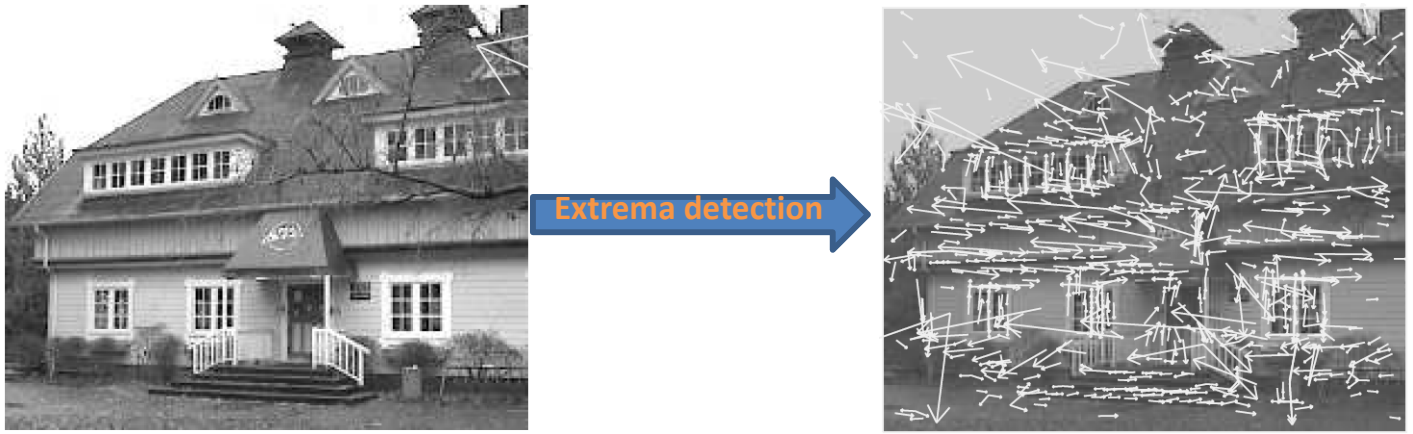
- Keypoint localization
  - Keypoint candidates selected in the previous step are fitted to the nearby data for location scale and ratio of principal curvatures
    - This allows to reject points of low contrast (sensitive to noise) or are poorly localized along the edge

Case study: SIFT

- Keypoint localization

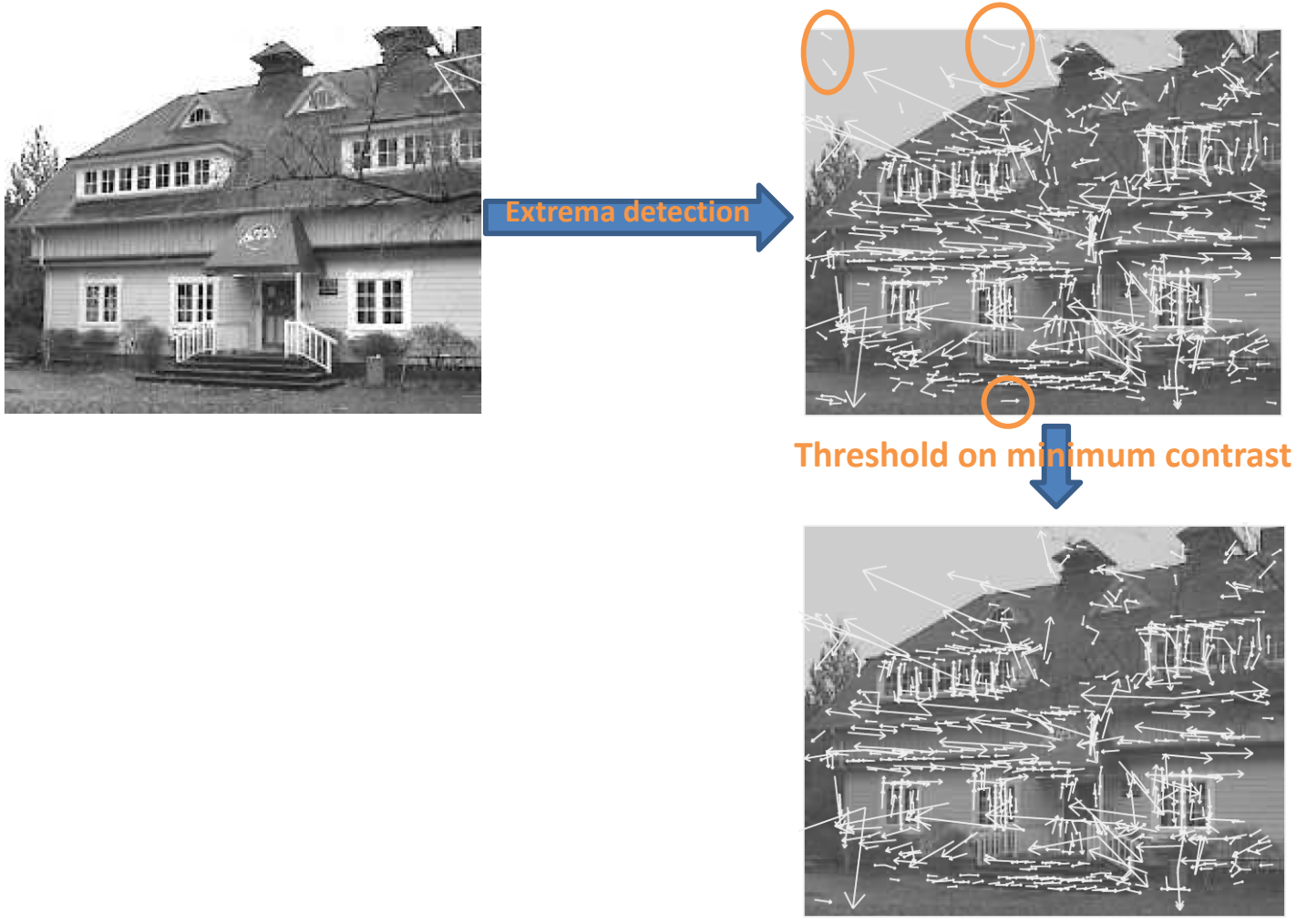
Case study: SIFT

- Keypoint localization



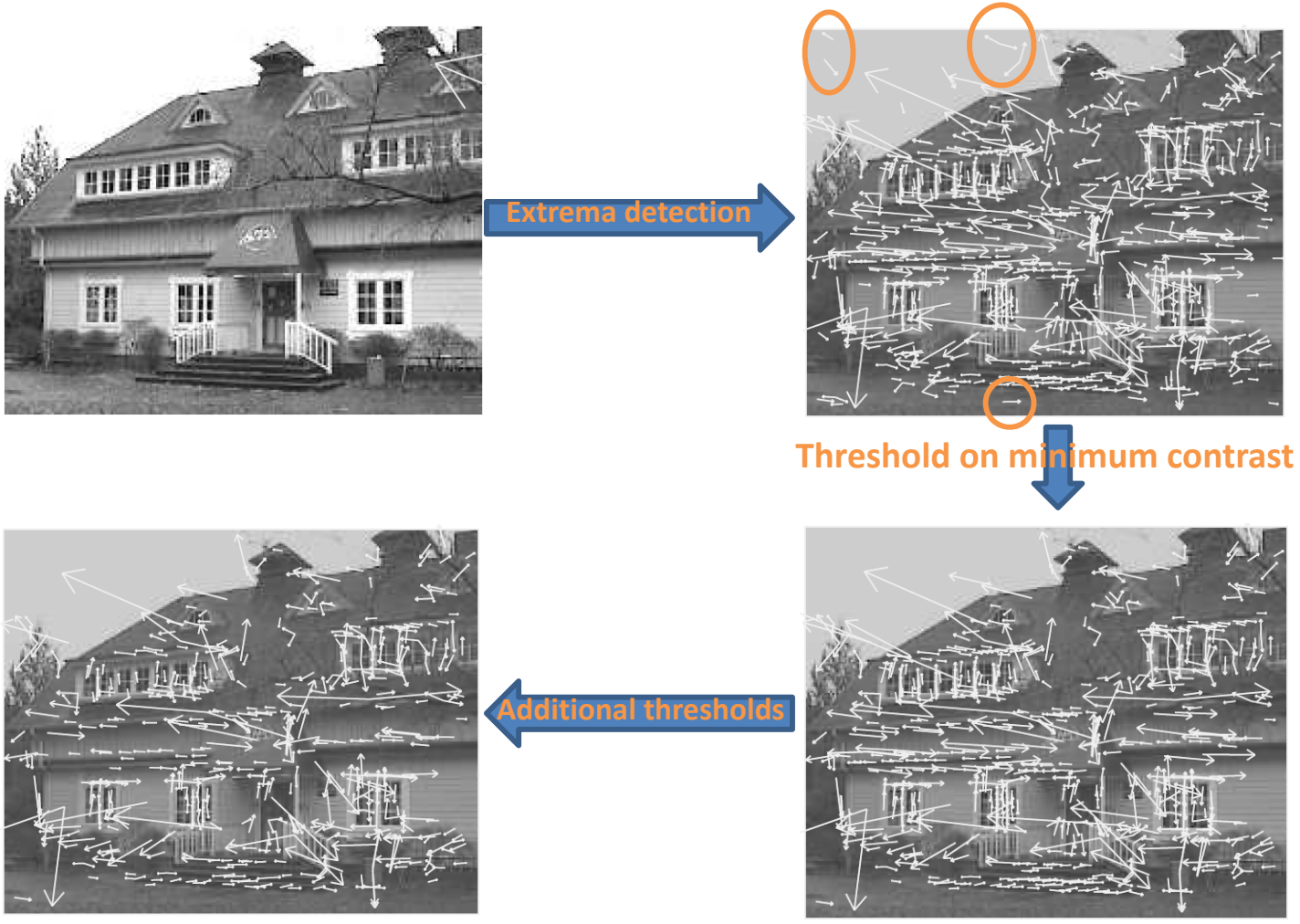
Case study: SIFT

- Keypoint localization



Case study: SIFT

- Keypoint localization



Case study: SIFT

1. Scale-space extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor

Case study: SIFT

- Orientation assignment
  - Each keypoint descriptor is represented relative to a consistent orientation based on local image properties
    - For?

Case study: SIFT

- Orientation assignment
  - Each keypoint descriptor is represented relative to a consistent orientation based on local image properties
    - For? …. Rotation invariance !

Case study: SIFT

- Orientation assignment
  - Each keypoint descriptor is represented relative to a consistent orientation based on local image properties
    - For? …. Rotation invariance !
    - For each image sample, $L(x,y)$ at the scale of the keypoint, the gradient magnitude $m(x,y)$ and orientation $\theta(x,y)$ is precomputed using pixel differences

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

Case study: SIFT

- Orientation assignment
  - Each keypoint descriptor is represented relative to a consistent orientation based on local image properties
    - For? …. Rotation invariance !
    - For each image sample, $L(x,y)$ at the scale of the keypoint, the gradient magnitude $m(x,y)$ and orientation $\theta(x,y)$ is precomputed using pixel differences.
    - Then, an orientation histogram is formed within a region around a keypoint.
    - Each sample added to the histogram is weighted by its gradient magnitude.
    - Peaks in the orientation histogram correspond to dominant directions of local gradients.
    - The orientations corresponding to the highest peak and local peaks that are within 80% of the highest peaks are assigned to the keypoint

Case study: SIFT

- Orientation assignment
  - Each keypoint descriptor is represented relative to a consistent orientation based on local image properties
    - For? …. Rotation invariance !
    - For each image sample, $L(x,y)$ at the scale of the keypoint, the gradient magnitude $m(x,y)$ and orientation $\theta(x,y)$ is precomputed using pixel differences.

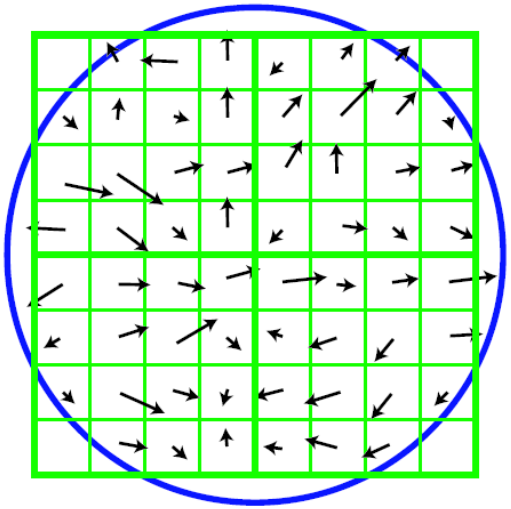


Image gradients

Case study: SIFT

1. Scale-space extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor

Case study: SIFT

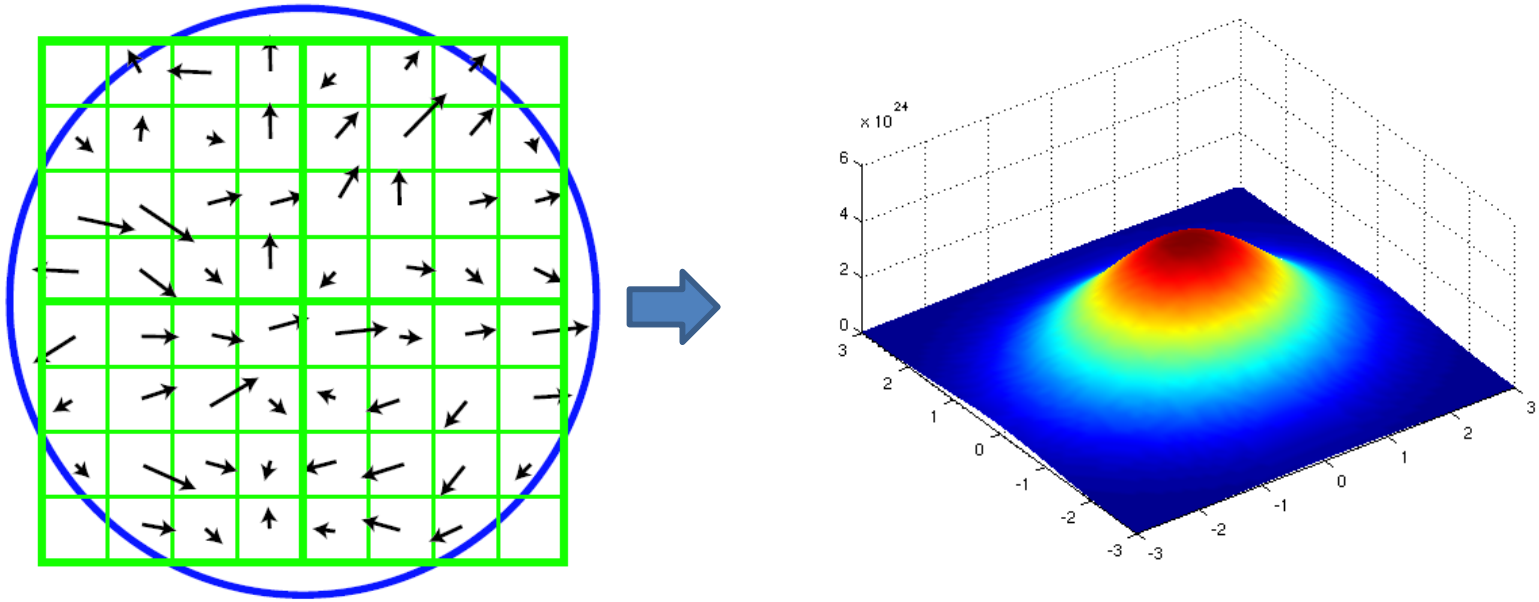
- Keypoint descriptor
  - So far, we have for each keypoint
    - Location
    - Scale and
    - Orientation

Case study: SIFT

- Keypoint descriptor
  - So far, we have for each keypoint
    - Location
    - Scale and
    - Orientation
  - We also need, invariance to
    - Illumination
    - 3D viewpoint

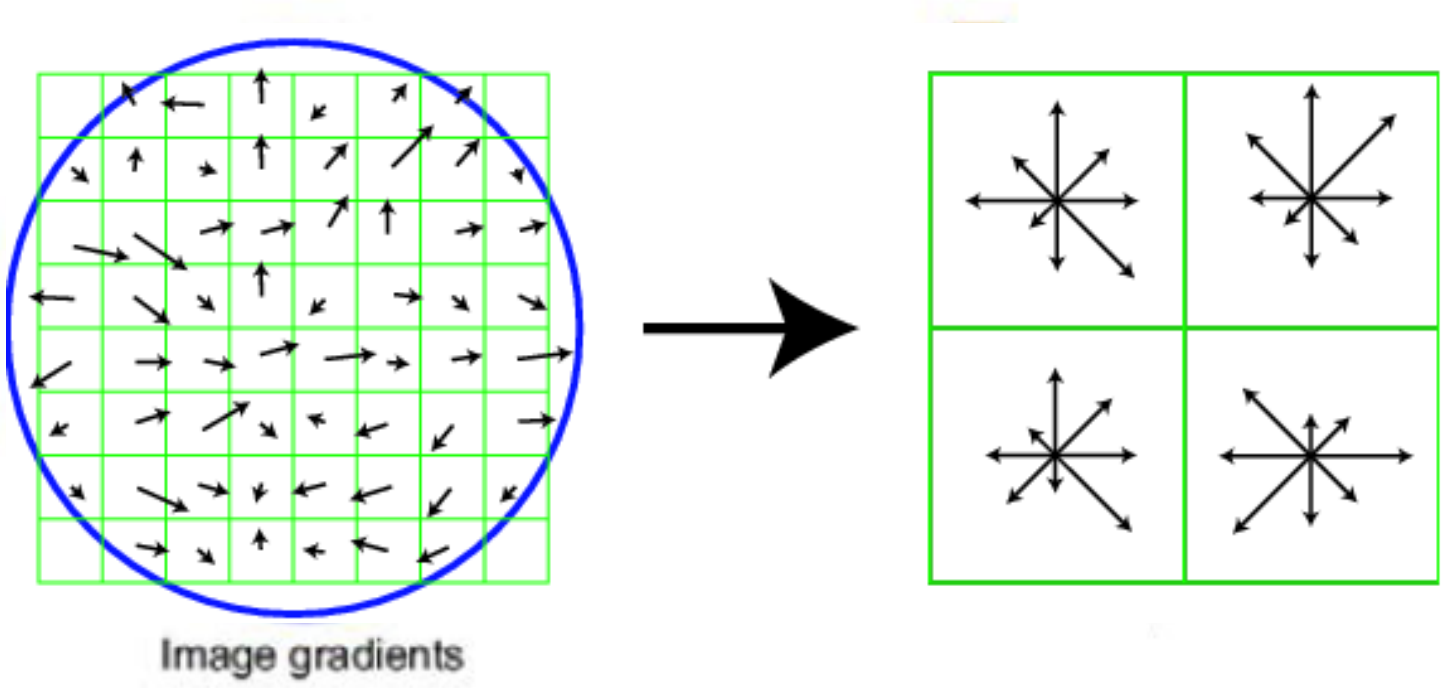
Case study: SIFT

- Keypoint descriptor
  - First, window is Gaussian weighted



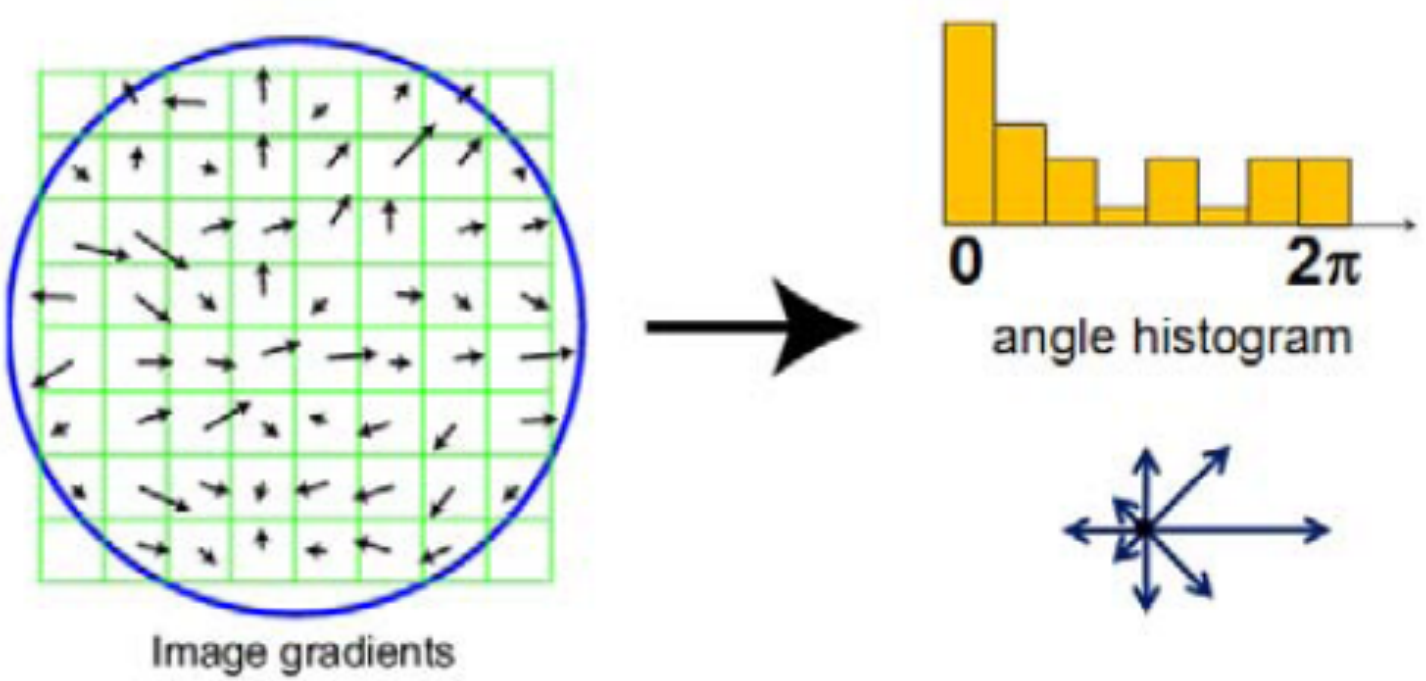
Case study: SIFT

- Keypoint descriptor
  - First, window is Gaussian weighted
  - Then, a histogram is computed in subregions using trilinear interpolations



Case study: SIFT

- Keypoint descriptor
  - First, window is Gaussian weighted
  - Then, a histogram is computed in subregions using trilinear interpolations

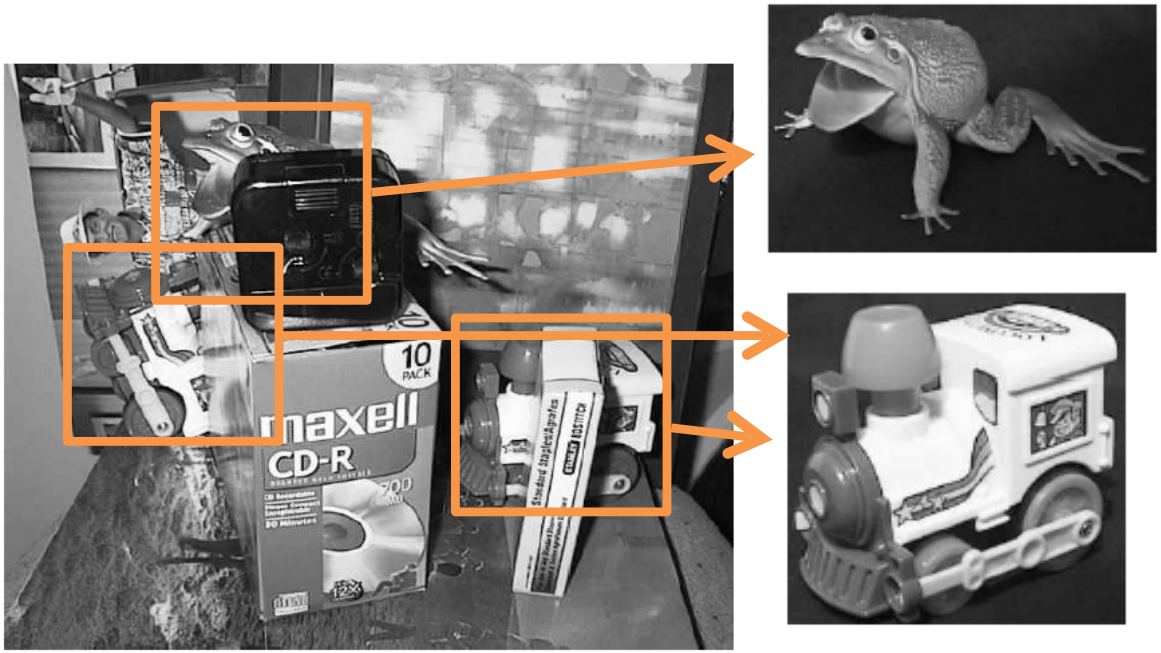


Case study: SIFT

- Results in object recognition

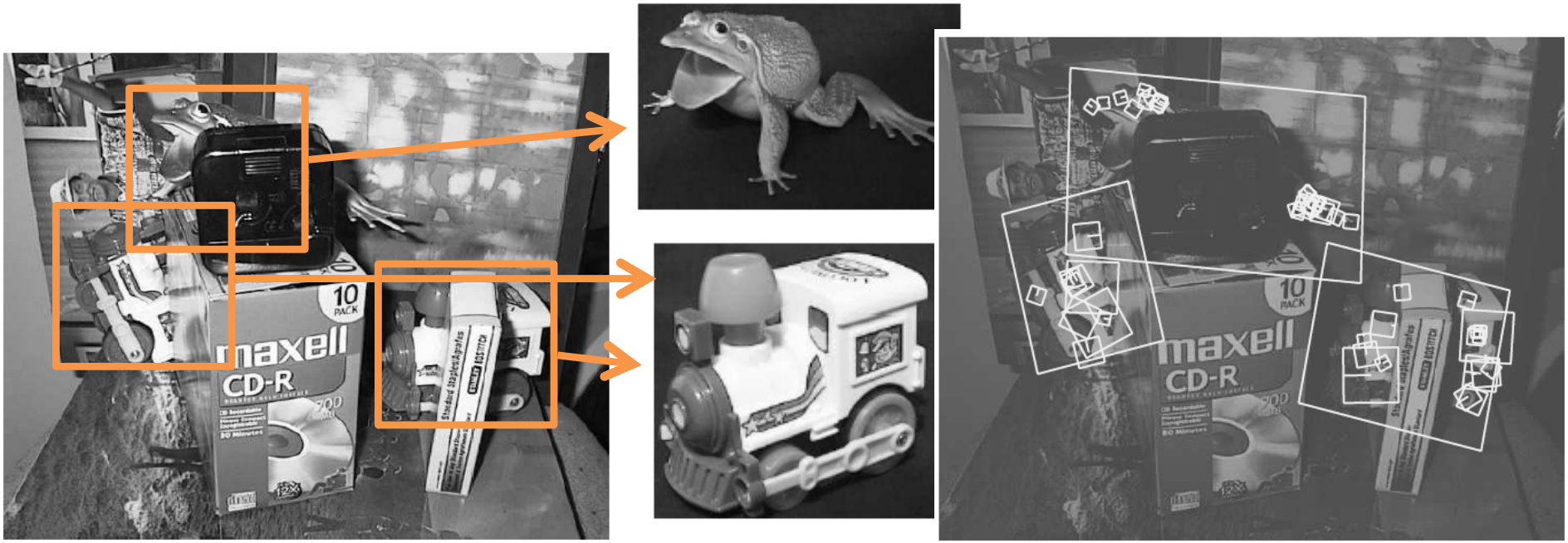
Case study: SIFT

- Results in object recognition



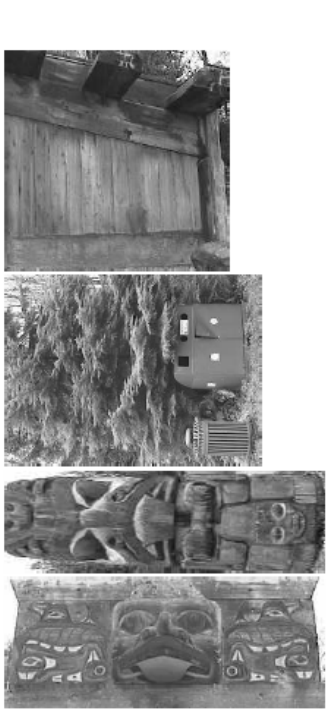
Case study: SIFT

- Results in object recognition



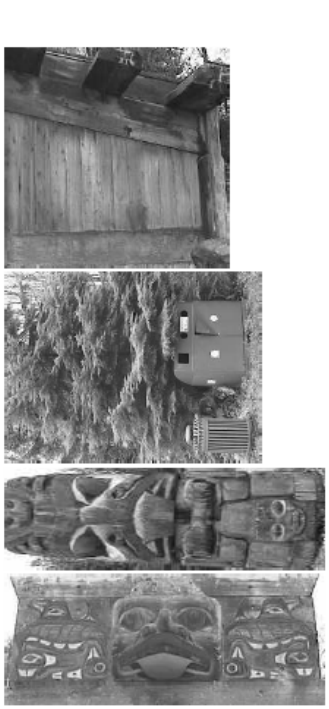
Case study: SIFT

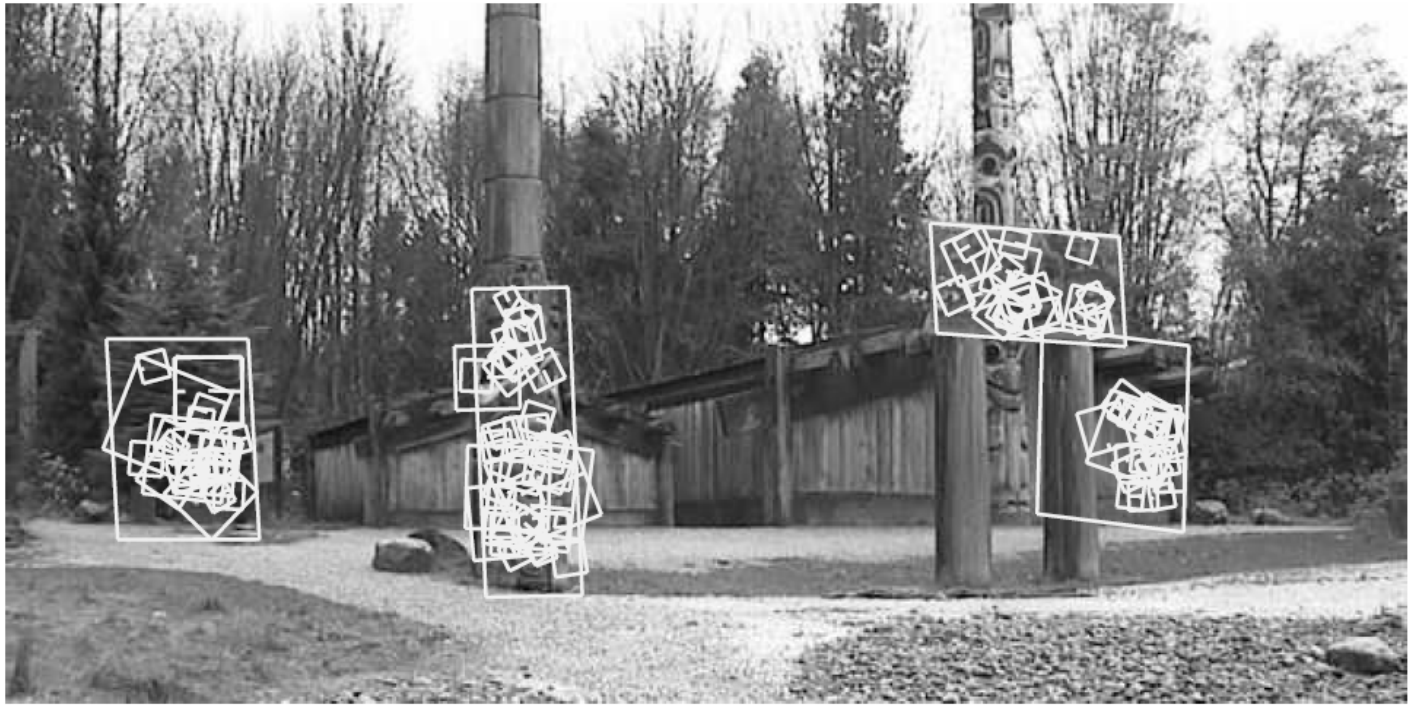
- Results in object recognition



Case study: SIFT

- Results in object recognition





Outline

- Introduction
- Point and patches
- Edges
- Corners

Edge detection

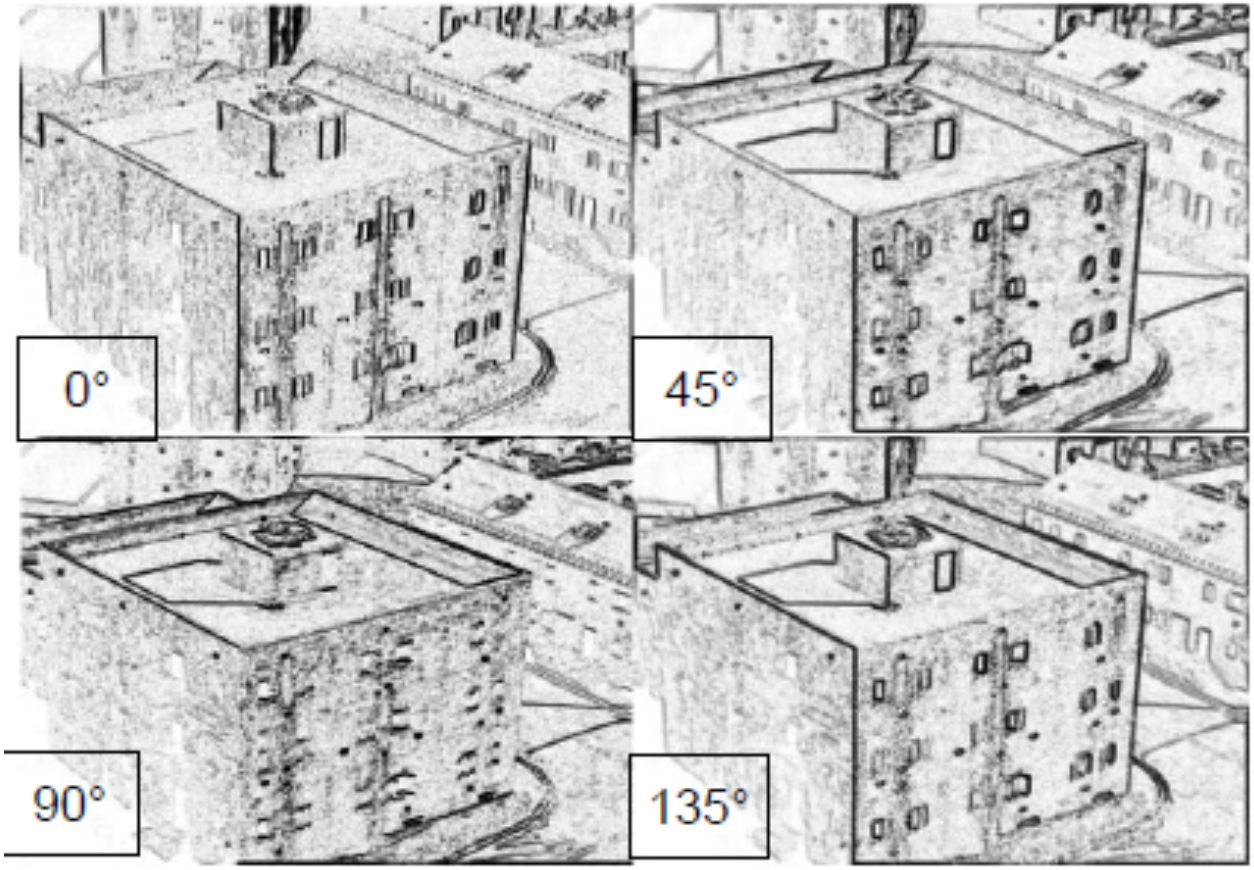
- Sharp changes in image brightness are interesting.
  - Object boundaries generate sharp changes in brightness.
  - Reflectance changes often generate sharp changes in brightness.
  - Sharp changes in surface orientation is also associated with sharp changes in image brightness.
  - Note : Shadows also generate sharp changes in brightness.
- Points in the image where brightness changes sharply are often called edges or edge points.
  - We want those that can be associated with boundaries.
  - But it is difficult to tell a meaningful edge from a nuisance edge.

Edge detection

- What is an edge
  - A border between two regions, each of which have approximately uniform brightness.
  - Edges often arise as the result of occluding contours in an image
    - The two image regions correspond to two different surfaces.

Source: Richard Wildes slides

Edge detection



Source: Richard Wildes slides

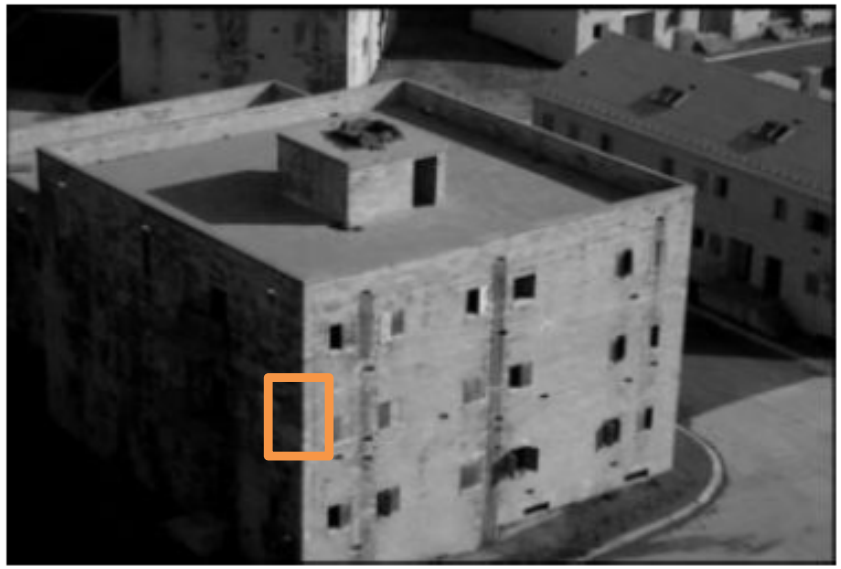
Edge detection

- What is an edge
  - A border between two regions, each of which have approximately uniform brightness.
  - Edges often arise as the result of occluding contours in an image
    - The two image regions correspond to two different surfaces.

Source: Richard Wildes slides

Edge detection

- What is an edge
  - A border between two regions, each of which have approximately uniform brightness.
  - Edges often arise as the result of occluding contours in an image.
    - The two image regions correspond to two different surfaces.
  - But again there are other sources of image edges.
    - Abrupt changes in surface orientation.



Source: Richard Wildes slides

Edge detection

- What is an edge
  - A border between two regions, each of which have approximately uniform brightness.
  - Edges often arise as the result of occluding contours in an image.
    - The two image regions correspond to two different surfaces.
  - But again there are other sources of image edges.
    - Abrupt changes in surface orientation.
    - Discontinuities in surface reflectance.

Source: Richard Wildes slides

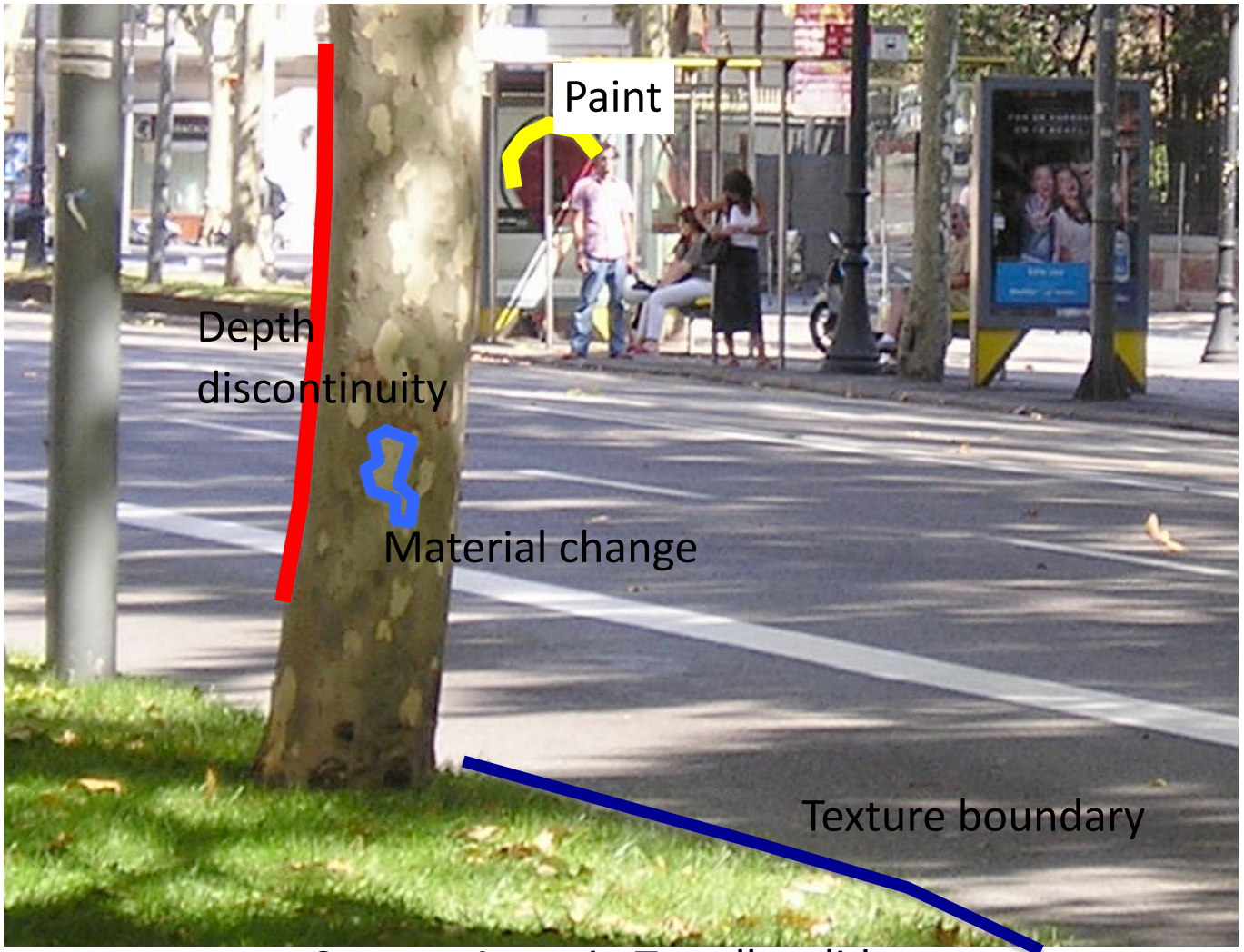
Edge detection

- What is an edge

Source: Antonio Torralba slides

Edge detection

- What is an edge



Source: Antonio Torralba slides

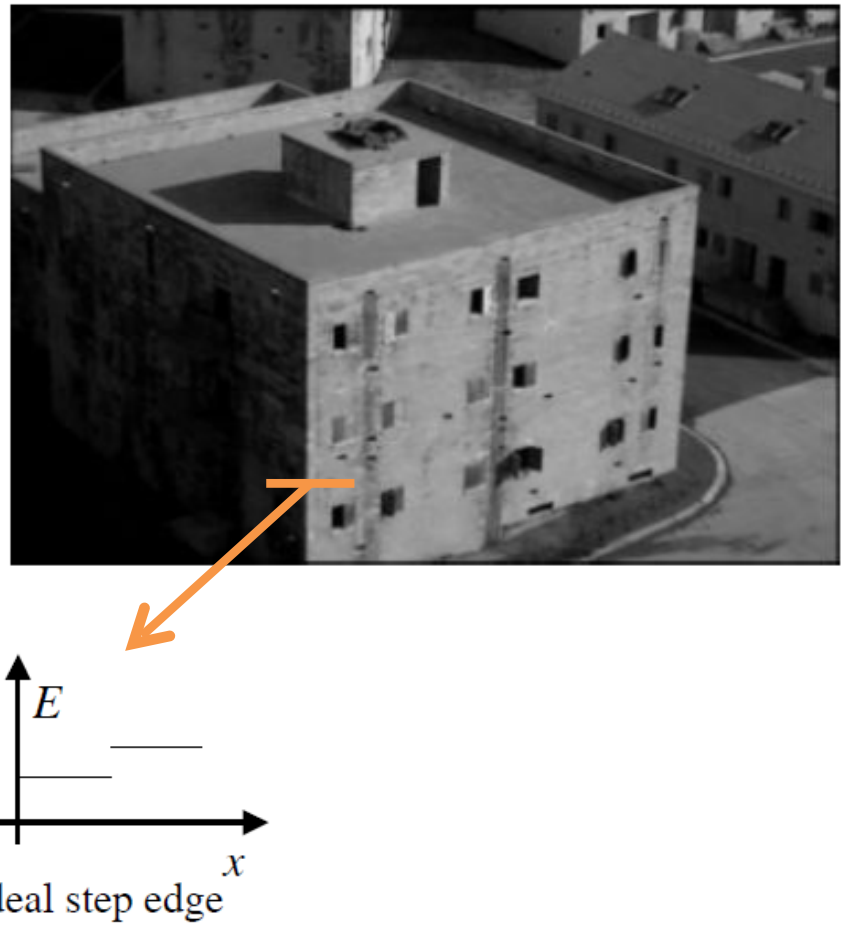
Edge detection

- Simple model
  - Consider a one-dimensional slice through an image in the vicinity of an abrupt change in image brightness.

Source: Richard Wildes slides

Edge detection

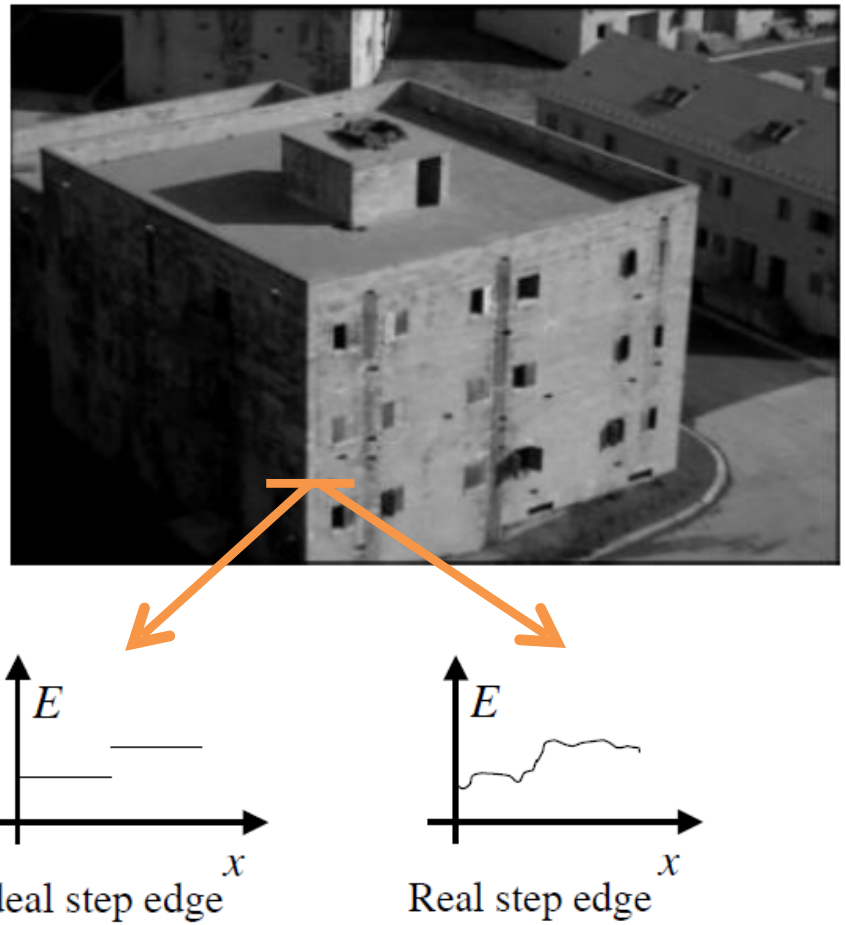
- Simple model
  - Consider a one-dimensional slice through an image in the vicinity of an abrupt change in image brightness.
  - Ideally, we might expect to find a step change in a plot of brightness vs. position.



Source: Richard Wildes slides

Edge detection

- Simple model
  - Consider a one-dimensional slice through an image in the vicinity of an abrupt change in image brightness.
  - Ideally, we might expect to find a step change in a plot of brightness vs. position.
  - In practice, we find a corrupted version of this ideal
    - Step transition smoothed
    - Noise transitions superimposed.



Source: Richard Wildes slides

Edge detection

General approach

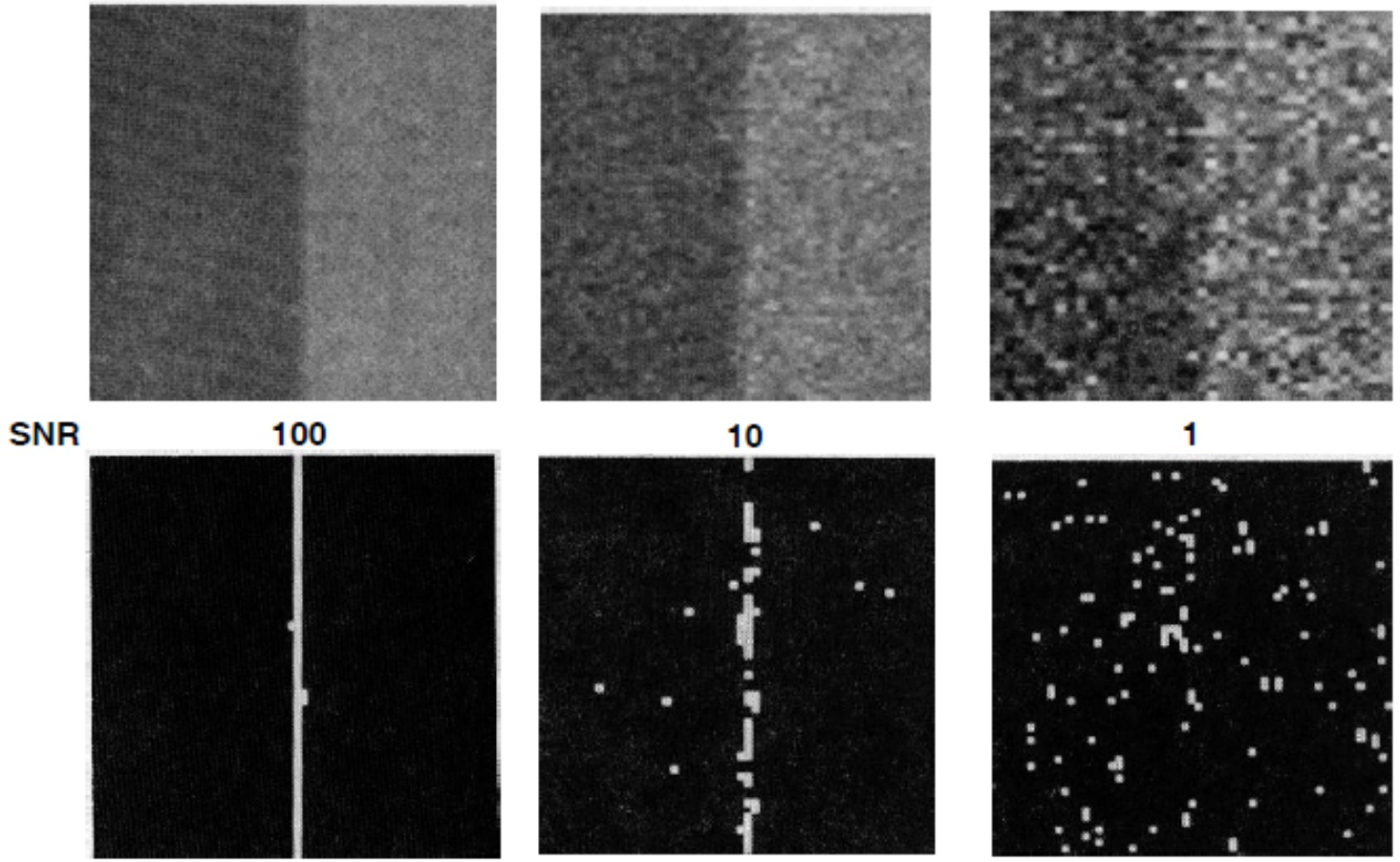
1. Suppress noise

2. Enhance edges

3. Locate edges

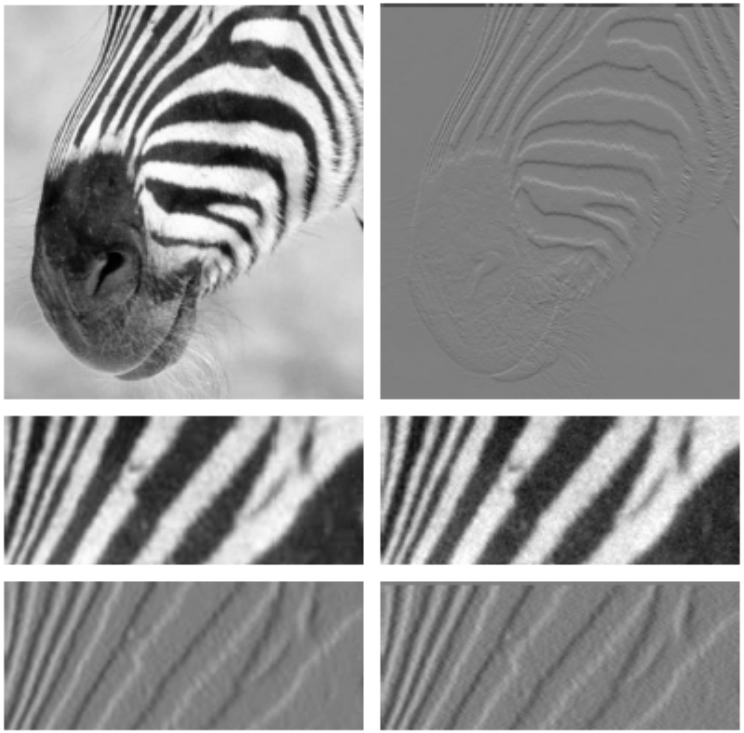
Edge detection: Noise

- Problem for edge detection.



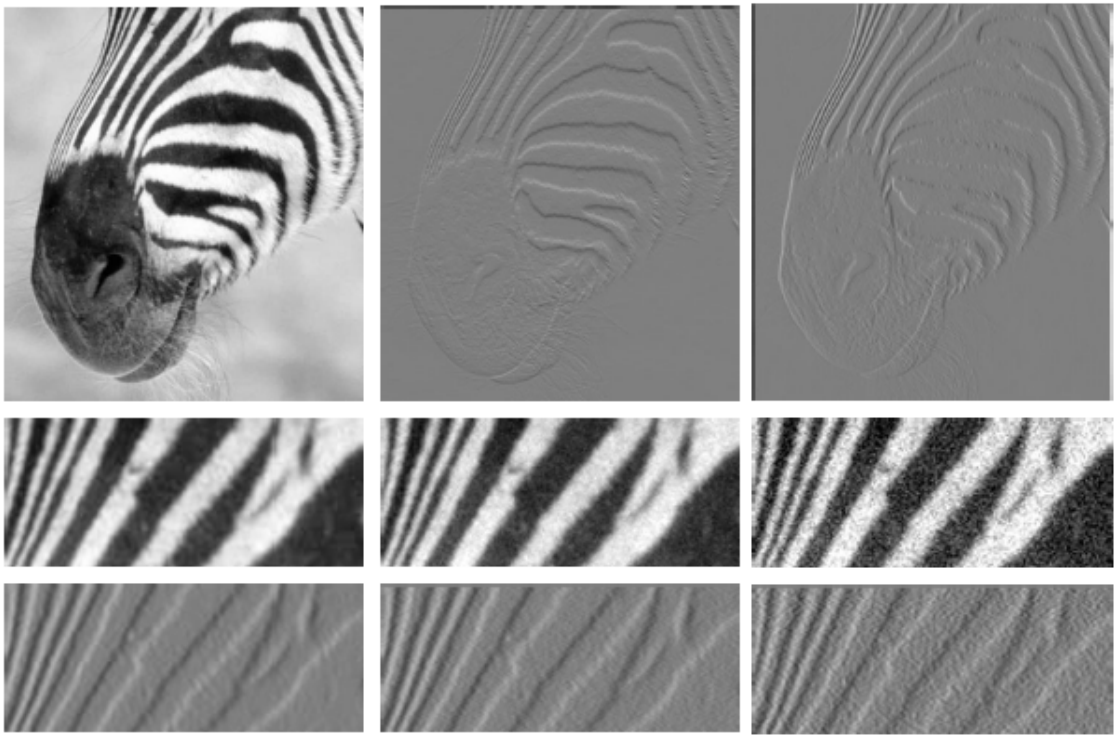
Edge detection: Noise

- Noise suppression
  - Many edge detection filters consist of finite difference computations as we will see next.
  - Differentiation accentuates high frequency components



Edge detection: Noise

- Noise suppression
  - Many edge detection filters consist of finite difference computations as we will see next.
  - Differentiation accentuates high frequency components



Edge detection: Noise

- Noise suppression
  - Many edge detection filters consist of finite difference computations as we will see next.
  - Differentiation accentuates high frequency components
  - We can assume that edges of interest will (typically) have frequency components across a wider range of frequencies (especially) lower frequencies than the noise.

Edge detection: Noise

- Noise suppression
  - In that case, a useful noise suppression is to convolve the image with a Gaussian.

$$h(x, y) = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{1}{2}\left(\frac{x^2 + y^2}{\sigma^2} \right) \right]$$

Edge detection: Noise

- Noise suppression
  - In that case, a useful noise suppression is to convolve the image with a Gaussian.

$$h(x,y) = \frac{1}{2\pi\sigma^2}\exp\left[-\frac{1}{2}\left(\frac{x^2+y^2}{\sigma^2}\right)\right]$$

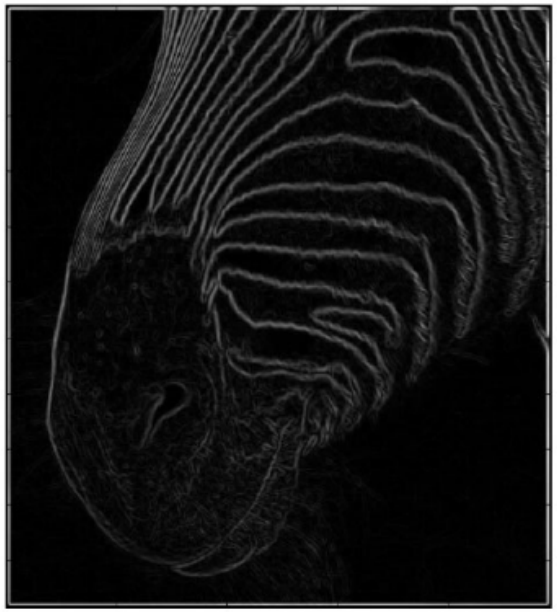
  - Usually noise suppression and enhacement are combined

Edge detection: Noise

- Noise suppression
  - The gradient magnitude can be estimated by smoothing an image and then differentiating it.
  - This is equivalent to convolving with the derivative of a smoothing kernel.

$$h_x(x,y) = -\frac{x}{2\pi\sigma^4}\exp\left[-\frac{1}{2}\left(\frac{x^2+y^2}{\sigma^2}\right)\right]$$

$$h_y(x,y) = -\frac{y}{2\pi\sigma^4}\exp\left[-\frac{1}{2}\left(\frac{x^2+y^2}{\sigma^2}\right)\right]$$



Edge detection

General approach

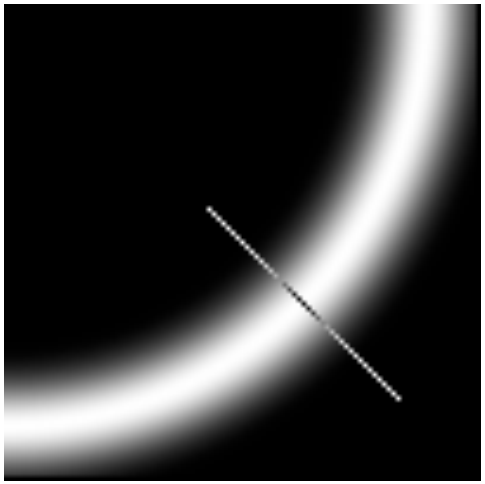
1. Suppress noise
2. Enhance edges
3. Locate edges

Edge detection: Enhancement

- Gradient-based edge detectors
  - We compute an estimate of the gradient magnitude
  - Look for points where the gradient magnitude is a maximum along the direction perpendicular to the edge
  - The direction perpendicular to the edge can be estimated using the direction of the gradient

Edge detection: Enhancement

- Gradient-based edge detectors
  - We compute an estimate of the gradient magnitude
  - Look for points where the gradient magnitude is a maximum along the direction perpendicular to the edge
  - The direction perpendicular to the edge can be estimated using the direction of the gradient



Edge detection: Enhancement

- Gradient-based edge detectors
  - We compute an estimate of the gradient magnitude
  - Look for points where the gradient **magnitude** is a maximum along the direction perpendicular to the edge
  - The direction perpendicular to the edge can be estimated using the direction of the gradient

Images credit: Christopher Rasmussen & Antonio Torralba

Edge detection: Enhancement

- Gradient-based edge detectors
  - We compute an estimate of the gradient magnitude
  - Look for points where the gradient magnitude is a maximum along the direction perpendicular to the edge
  - The **direction** perpendicular to the edge can be estimated using the direction of the gradient

Edge detection: Localization

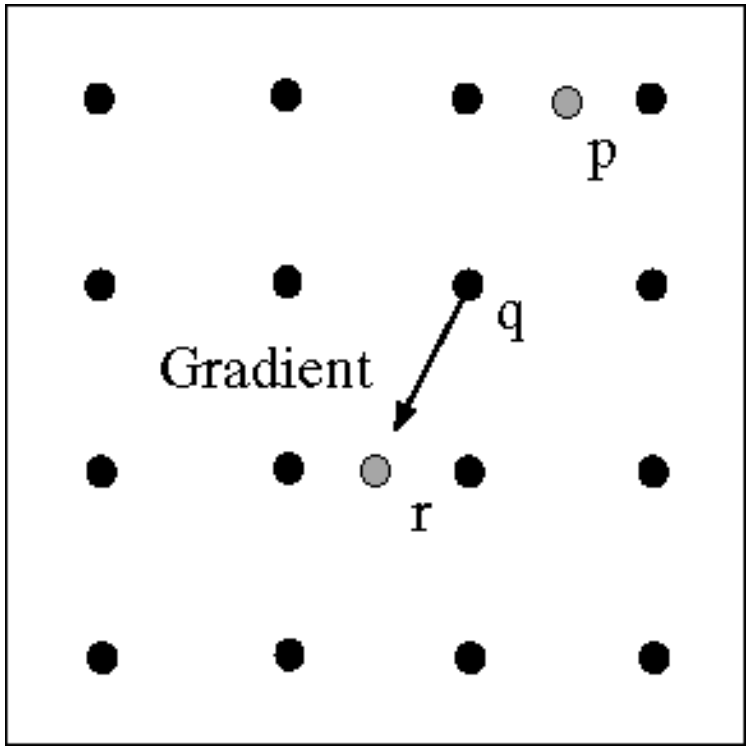
- Gradient-based edge detectors: Non-maximum suppression
  - Points where the gradient magnitude is at a maximum along the direction of the gradient.



Images credit: Christopher Rasmussen & Antonio Torralba

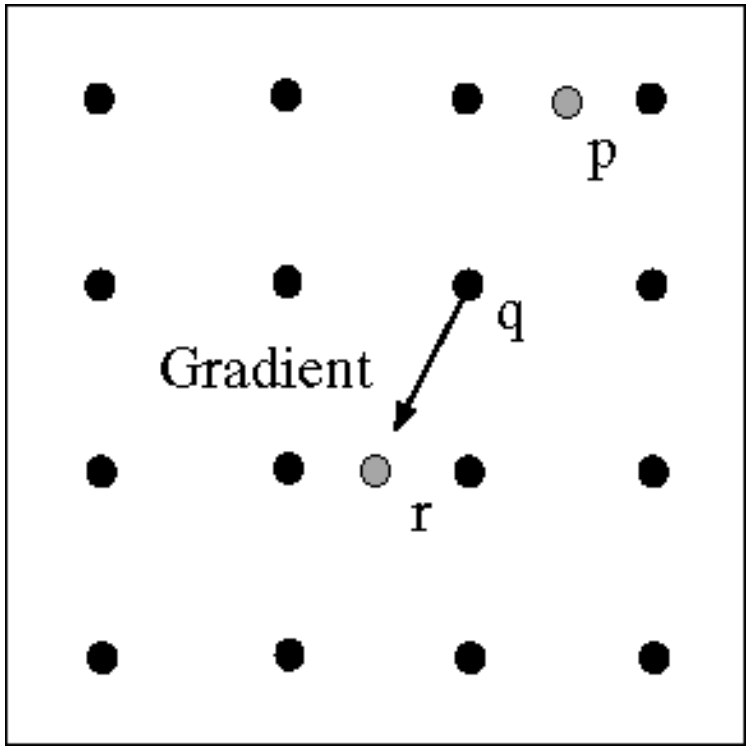
Edge detection: Localization

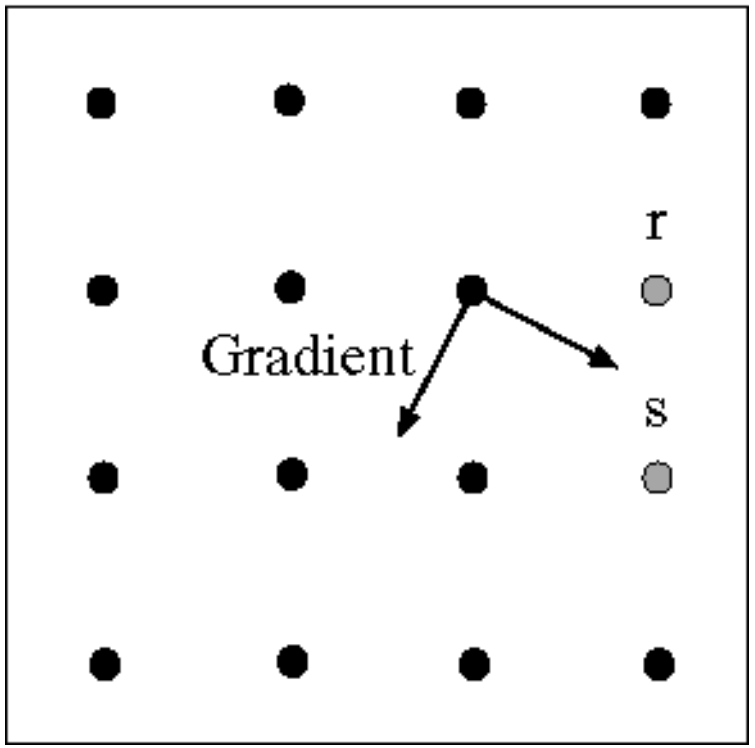
- Gradient-based edge detectors: Non-maximum suppression
  - Points where the gradient magnitude is at a maximum along the direction of the gradient.



Edge detection: Localization

- Gradient-based edge detectors: Non-maximum suppression
  - Points where the gradient magnitude is at a maximum along the direction of the gradient.





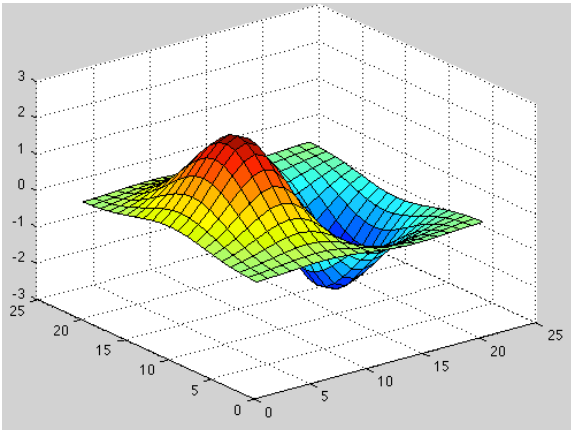
Edge detection

- Gradient-based edge detectors: Canny edge detector (1986)
  1. Noise reduction: Gaussian filter.

Edge detection

- Gradient-based edge detectors: Canny edge detector
  1. Noise reduction: Gaussian filter.
  2. Four filters to detect horizontal, vertical and diagonal edges.

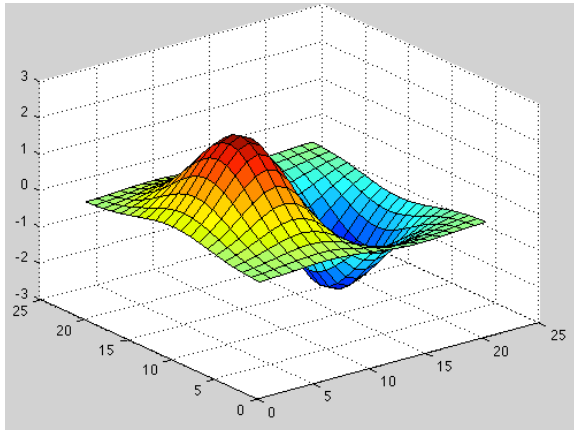
$$h_x(x,y) = \frac{\partial h(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

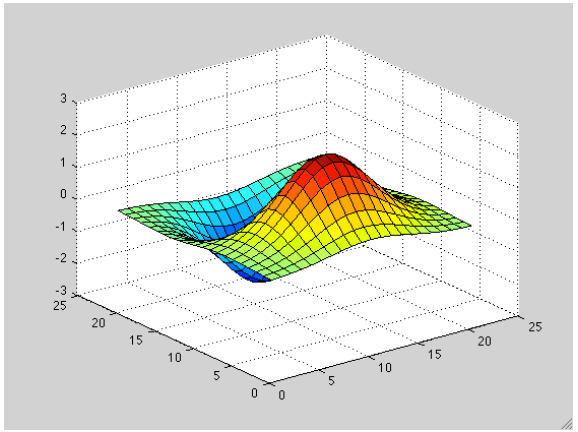


Edge detection

- Gradient-based edge detectors: Canny edge detector
  1. Noise reduction: Gaussian filter.
  2. Four filters to detect horizontal, vertical and diagonal edges.

$$h_x(x,y) = \frac{\partial h(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad h_y(x,y) = \frac{\partial h(x,y)}{\partial x} = \frac{-y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$





Edge detection

- Gradient-based edge detectors: Canny edge detector
  1. Noise reduction: Gaussian filter.
  2. Four filters to detect horizontal, vertical and diagonal edges.

$$h_x(x,y) = \frac{\partial h(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad h_y(x,y) = \frac{\partial h(x,y)}{\partial x} = \frac{-y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Magnitude: $h_x(x,y)^2 + h_y(x,y)^2$ *Edge strength*

Angle: $\arctan\left(\frac{h_y(x,y)}{h_x(x,y)}\right)$ *Edge normal*

Source: Antonio Torralba

Edge detection

- Gradient-based edge detectors: Canny edge detector
  1. Noise reduction: Gaussian filter.
  2. Four filters to detect horizontal, vertical and diagonal edges.
  3. Non-maximum suppression.

Source: Antonio Torralba

Edge detection

- Gradient-based edge detectors: Canny edge detector
  1. Noise reduction: Gaussian filter.
  2. Four filters to detect horizontal, vertical and diagonal edges.
  3. Non-maximum suppression.
  4. Hysteresis.
     - By just applying non-maximum suppression there are too many of these curves.
     - Usually we apply a threshold, but …

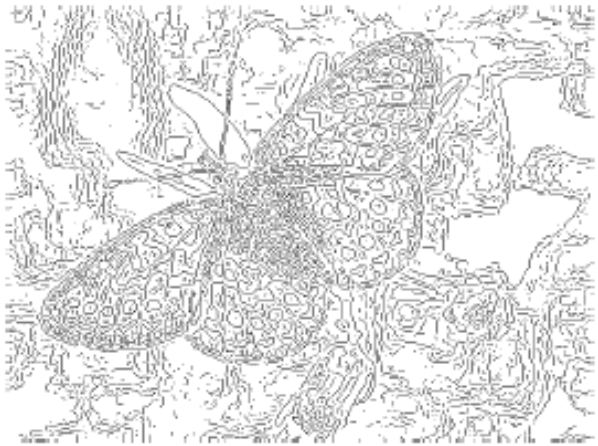
Source: Antonio Torralba

Edge detection

- Gradient-based edge detectors: Canny edge detector
  1. Noise reduction: Gaussian filter.
  2. Four filters to detect horizontal, vertical and diagonal edges.
  3. Non-maximum suppression.
  4. Hysteresis.

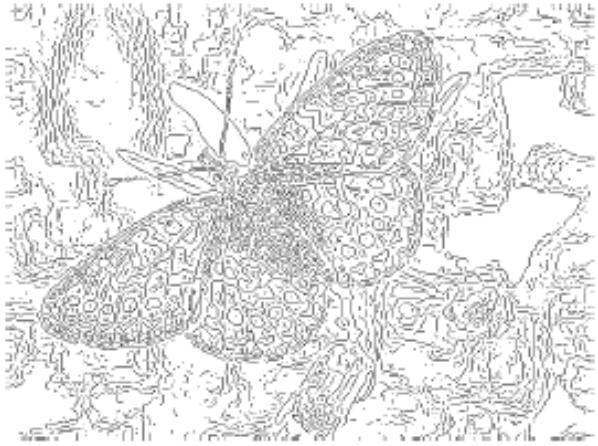
Edge detection

- Gradient-based edge detectors: Canny edge detector
  1. Noise reduction: Gaussian filter.
  2. Four filters to detect horizontal, vertical and diagonal edges.
  3. Non-maximum suppression.
  4. Hysteresis.



Edge detection

- Gradient-based edge detectors: Canny edge detector
  1. Noise reduction: Gaussian filter.
  2. Four filters to detect horizontal, vertical and diagonal edges.
  3. Non-maximum suppression.
  4. Hysteresis.

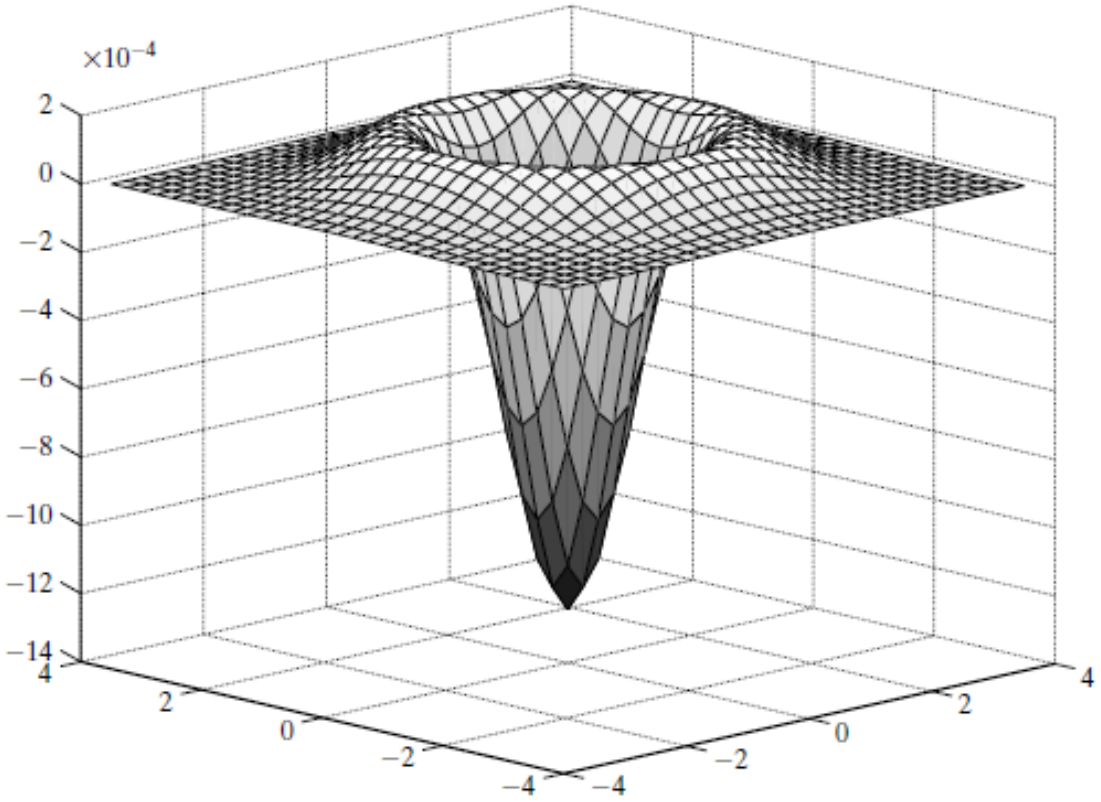


Edge detection

- Gradient-based edge detectors: Canny edge detector
  1. Noise reduction: Gaussian filter.
  2. Four filters to detect horizontal, vertical and diagonal edges.
  3. Non-maximum suppression.
  4. Hysteresis.
     - Solution: Have two thresholds.
       - Use the larger when starting an edge chain.
       - Use the smaller while following it.

Edge detection: Enhancement

- Laplacian
  - Remember: Second partial derivatives



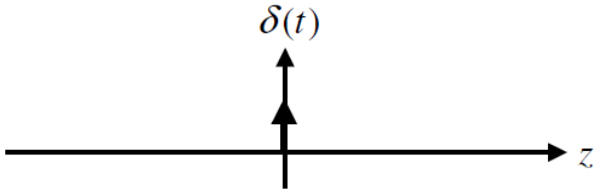
Edge detection: Enhancement

- Laplacian
  - Remember: Second partial derivatives

$$\frac{\partial^2 E}{\partial x^2} = \sin^2\theta(B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

$$\frac{\partial^2 E}{\partial y^2} = \cos^2\theta(B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

  - Note
    - Remember δ is the impulse



Source: Richard Wildes slides

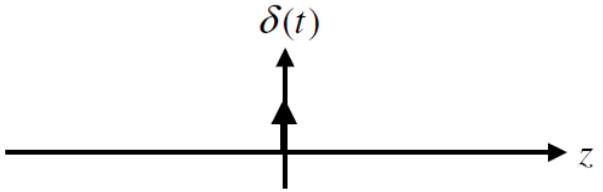
Edge detection: Enhancement

- Laplacian
  - Remember: Second partial derivatives

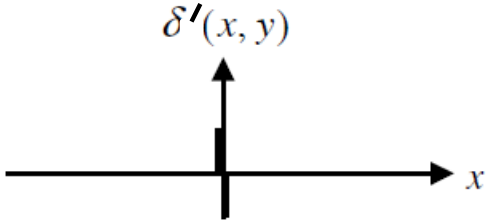
$$\frac{\partial^2 E}{\partial x^2} = \sin^2\theta(B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

$$\frac{\partial^2 E}{\partial y^2} = \cos^2\theta(B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

  - Note
    - Remember δ is the impulse



    - δ' is the impulse derivative



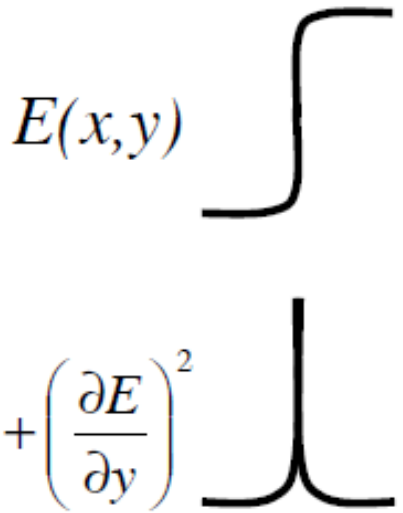
Source: Richard Wildes slides

Edge detection: Enhancement

- Laplacian
  - Remember: Second partial derivatives

$$\frac{\partial^2 E}{\partial x^2} = \sin^2\theta (B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

$$\frac{\partial^2 E}{\partial y^2} = \cos^2\theta (B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$



  - We define the Laplacian as

$$\nabla^2 E = \frac{\partial^2 E}{\partial x^2} + \frac{\partial^2 E}{\partial y^2} = (B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

Source: Richard Wildes slides

Edge detection: Enhancement

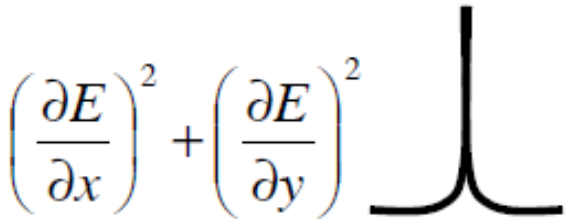
- Laplacian
  - Remember: Second partial derivatives

$$\frac{\partial^2 E}{\partial x^2} = \sin^2\theta (B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

$$\frac{\partial^2 E}{\partial y^2} = \cos^2\theta (B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

  - We define the Laplacian as

$$\nabla^2 E = \frac{\partial^2 E}{\partial x^2} + \frac{\partial^2 E}{\partial y^2} = (B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

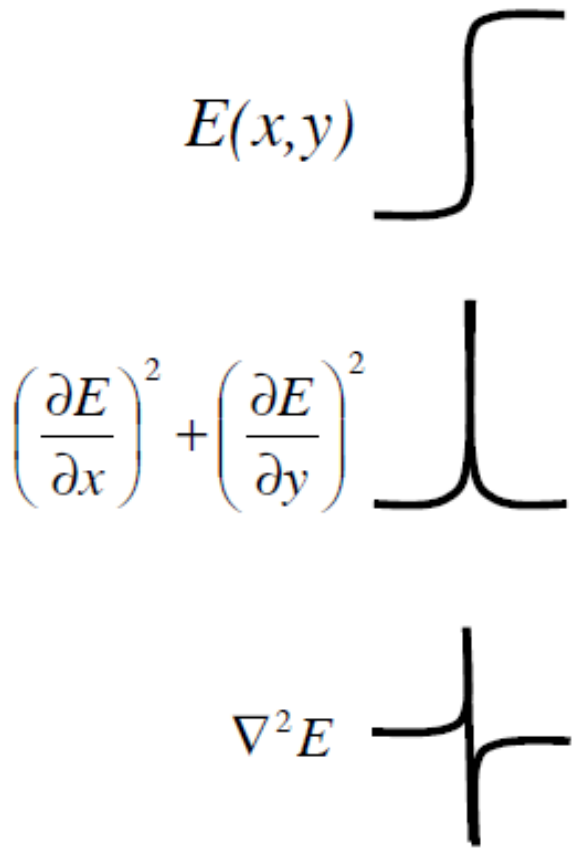


  - This operation will show a "zero-crossing" as we cross the edge.

Source: Richard Wildes slides

Edge detection: Enhancement

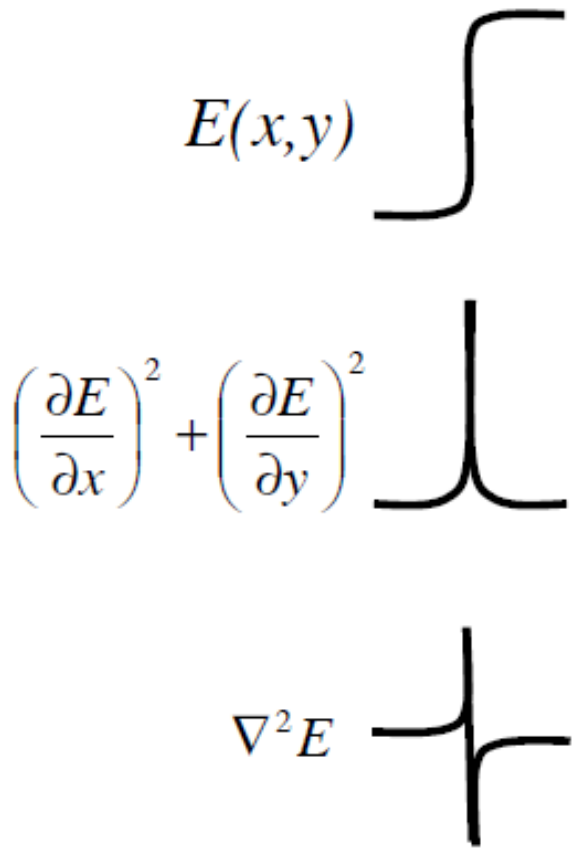
- Laplacian
  - This operation will show a "zero-crossing" as we cross the edge.
  - Similarity with the squared gradient
    - The response of the operator is independent of the edge orientation.



Source: Richard Wildes slides

Edge detection: Enhancement

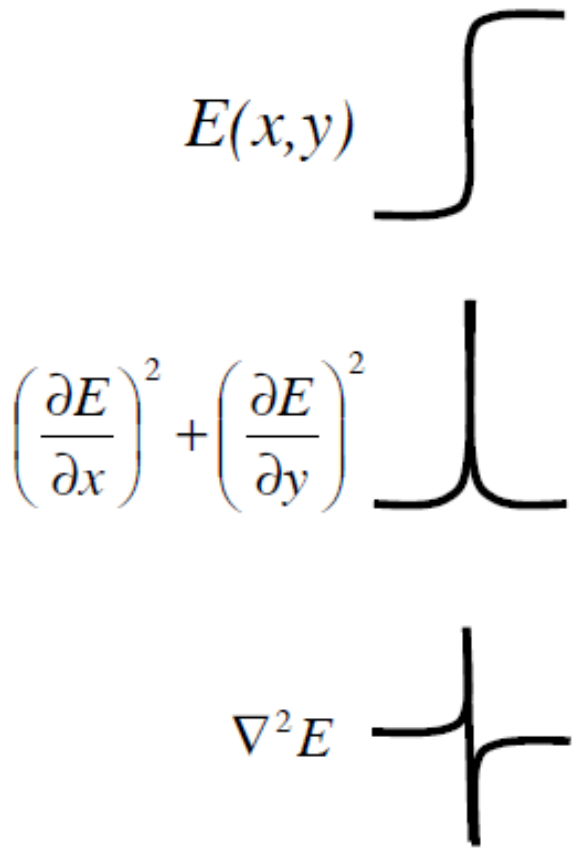
- Laplacian
  - This operation will show a "zero-crossing" as we cross the edge.
  - Similarity with the squared gradient
    - The response of the operator is independent of the edge orientation.
  - But unlike the squared gradient
    - The Laplacian is linear
    - The Laplacian the sign of intensity change across the edge.



Source: Richard Wildes slides

Edge detection: Localization

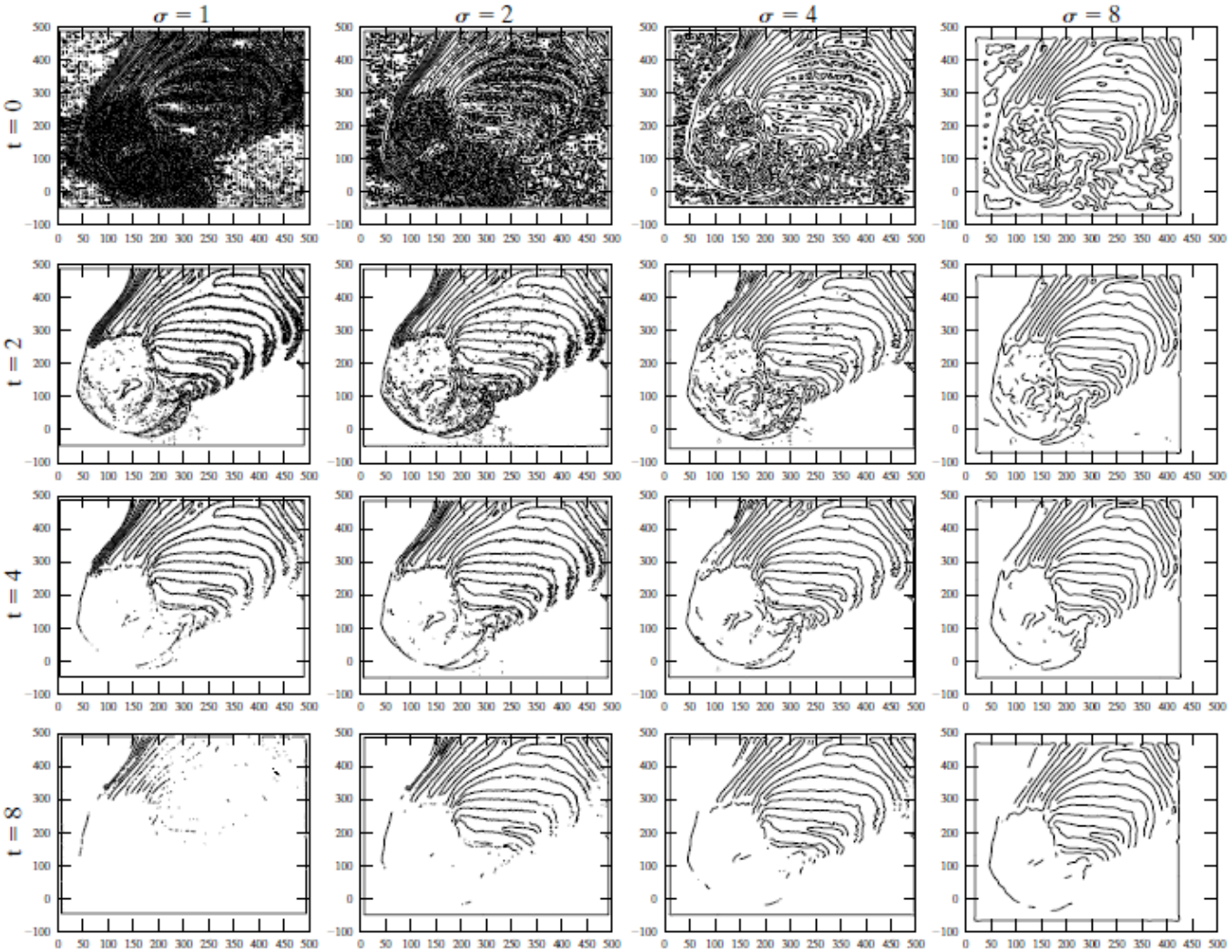
- Laplacian
  - Remember for the gradient we would use a threshold and non-maxima suppression
  - For the Laplacian we seek a transition magnitude across the zero-crossing above which we will declare a value as marking an edge location.



Source: Richard Wildes slides

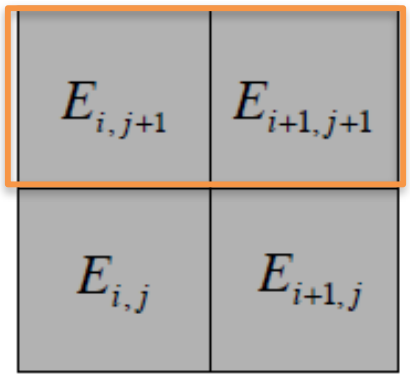
Edge detection: Localization

- Laplacian



Edge detection

- Discrete approximations: The gradient



- We can make the following approximation

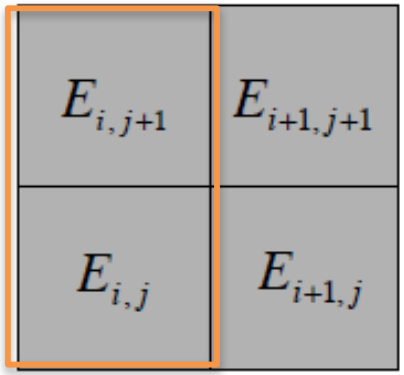
$$\frac{\partial E}{\partial x} \approx \frac{1}{2\varepsilon}[(E_{i+1,j+1} - E_{i,j+1}) + (E_{i+1,j} - E_{i,j})]$$

$$\frac{\partial E}{\partial y} \approx \frac{1}{2\varepsilon}[(E_{i+1,j+1} - E_{i+1,j}) + (E_{i,j+1} - E_{i,j})]$$

Source: Richard Wildes slides

Edge detection

- Discrete approximations: The gradient



- We can make the following approximation

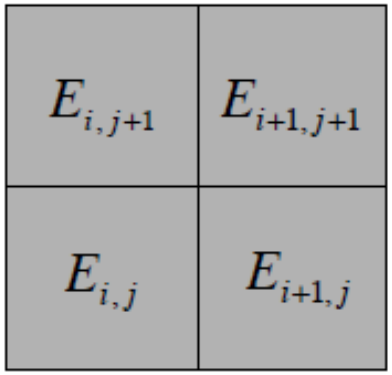
$$\frac{\partial E}{\partial x} \approx \frac{1}{2\varepsilon}[(E_{i+1,j+1} - E_{i,j+1}) + (E_{i+1,j} - E_{i,j})]$$

$$\frac{\partial E}{\partial y} \approx \frac{1}{2\varepsilon}[(E_{i+1,j+1} - E_{i+1,j}) + (E_{i,j+1} - E_{i,j})]$$

Source: Richard Wildes slides

Edge detection

- Discrete approximations: The gradient



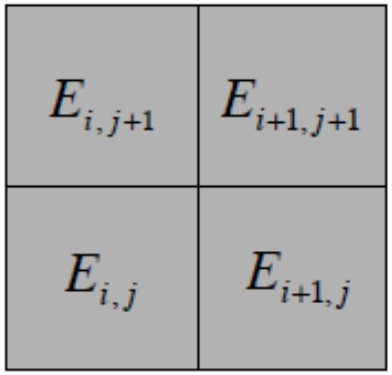
- The Gradient is

$$\left(\frac{\partial E}{\partial x}\right)^2 + \left(\frac{\partial E}{\partial y}\right)^2 \approx \frac{1}{2\varepsilon}[(E_{i+1,j+1} - E_{i,j})^2 + (E_{i,j+1} - E_{i+1,j})^2]$$

Source: Richard Wildes slides

Edge detection

- Discrete approximations: The gradient



- The Gradient is

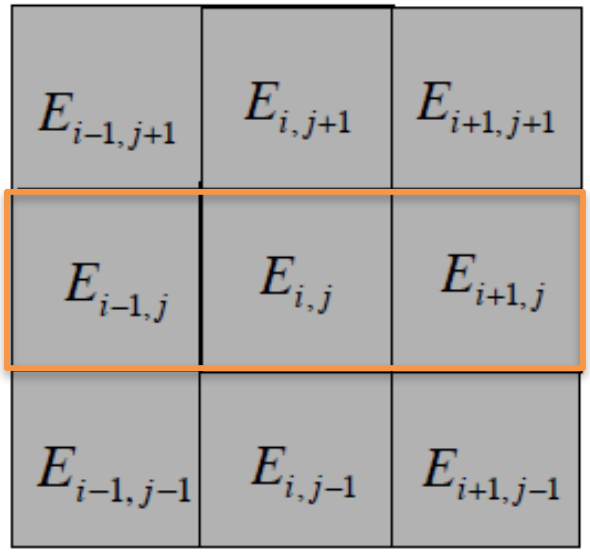
$$\left(\frac{\partial E}{\partial x}\right)^2 + \left(\frac{\partial E}{\partial y}\right)^2 \approx \frac{1}{2\varepsilon}[(E_{i+1,j+1} - E_{i,j})^2 + (E_{i,j+1} - E_{i+1,j})^2]$$

  - We obtain large values where the image brightness is changing rapidly.
  - We obtain a new image with edges strongly emphasized.

Source: Richard Wildes slides

Edge detection

- Discrete approximations: The Laplacian



- The approximation is

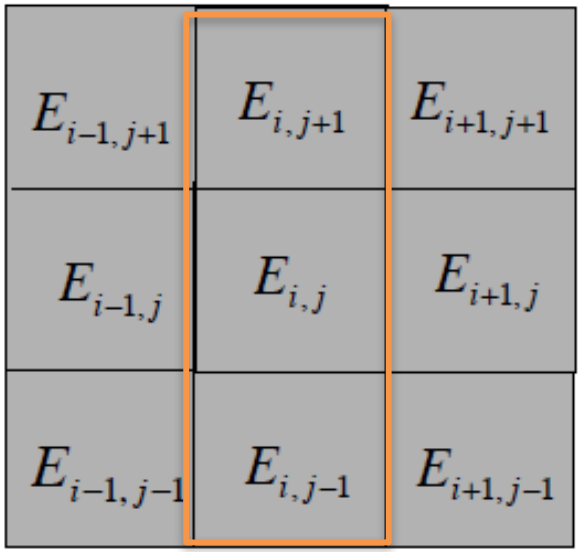
$$\frac{\partial^2 E}{\partial x^2} \approx \frac{1}{\varepsilon^2}(E_{i-1,j} - 2E_{i,j} + E_{i+1,j})$$

$$\frac{\partial^2 E}{\partial y^2} \approx \frac{1}{\varepsilon^2}(E_{i,j-1} - 2E_{i,j} + E_{i,j+1})$$

Source: Richard Wildes slides

Edge detection

- Discrete approximations: The Laplacian



- The approximation is

$$\frac{\partial^2 E}{\partial x^2} \approx \frac{1}{\varepsilon^2}(E_{i-1,j} - 2E_{i,j} + E_{i+1,j})$$

$$\frac{\partial^2 E}{\partial y^2} \approx \frac{1}{\varepsilon^2}(E_{i,j-1} - 2E_{i,j} + E_{i,j+1})$$

Source: Richard Wildes slides

Edge detection

- Discrete approximations: The Laplacian

$E_{i-1,j+1}$	$E_{i,j+1}$	$E_{i+1,j+1}$
$E_{i-1,j}$	$E_{i,j}$	$E_{i+1,j}$
$E_{i-1,j-1}$	$E_{i,j-1}$	$E_{i+1,j-1}$

- The Laplacian is then

$$\frac{\partial^2 E}{\partial x^2}+\frac{\partial^2 E}{\partial y^2}\approx\frac{4}{\varepsilon^2}\left[\frac{1}{4}(E_{i-1,j}+E_{i,j-1}+E_{i+1,j}+Ei,\,j+1)-E_{i,j}\right]$$

Source: Richard Wildes slides

Edge detection

- Discrete approximations: The Laplacian

$E_{i-1,j+1}$	$E_{i,j+1}$	$E_{i+1,j+1}$
$E_{i-1,j}$	$E_{i,j}$	$E_{i+1,j}$
$E_{i-1,j-1}$	$E_{i,j-1}$	$E_{i+1,j-1}$

- The Laplacian is then

$$\frac{\partial^2 E}{\partial x^2}+\frac{\partial^2 E}{\partial y^2}\approx\frac{4}{\varepsilon^2}\left[\frac{1}{4}(E_{i-1,j}+E_{i,j-1}+E_{i+1,j}+Ei,\,j+1)-E_{i,j}\right]$$

  - This is the average of the surrounding values and subtracts that of the center

Source: Richard Wildes slides

Edge detection

- Discrete approximations: The Laplacian
  - It can be mapped onto this mask.

$$\frac{1}{\varepsilon^2}\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

  - Which can be used for convolutions.

Source: Richard Wildes slides

Edge detection

- Seeking human performance

Edge detection

- Seeking human performance
  - Reason

Source: Antonio Torralba slides

Edge detection

- Seeking human performance
  - Reason

Source: Antonio Torralba slides

Edge detection

- Seeking human performance

Source: Antonio Torralba slides

Gaussian gradient boundaries

Human boundaries

Edge detection

- Seeking human performance

Source: Antonio Torralba slides

Gaussian gradient boundaries

Human boundaries

Edge detection

- Seeking human performance

Source: Antonio Torralba slides

Gaussian gradient boundaries

Human boundaries

Edge detection

- Seeking human performance: Example

Source: Jitendra Malik (2004)

Edge detection

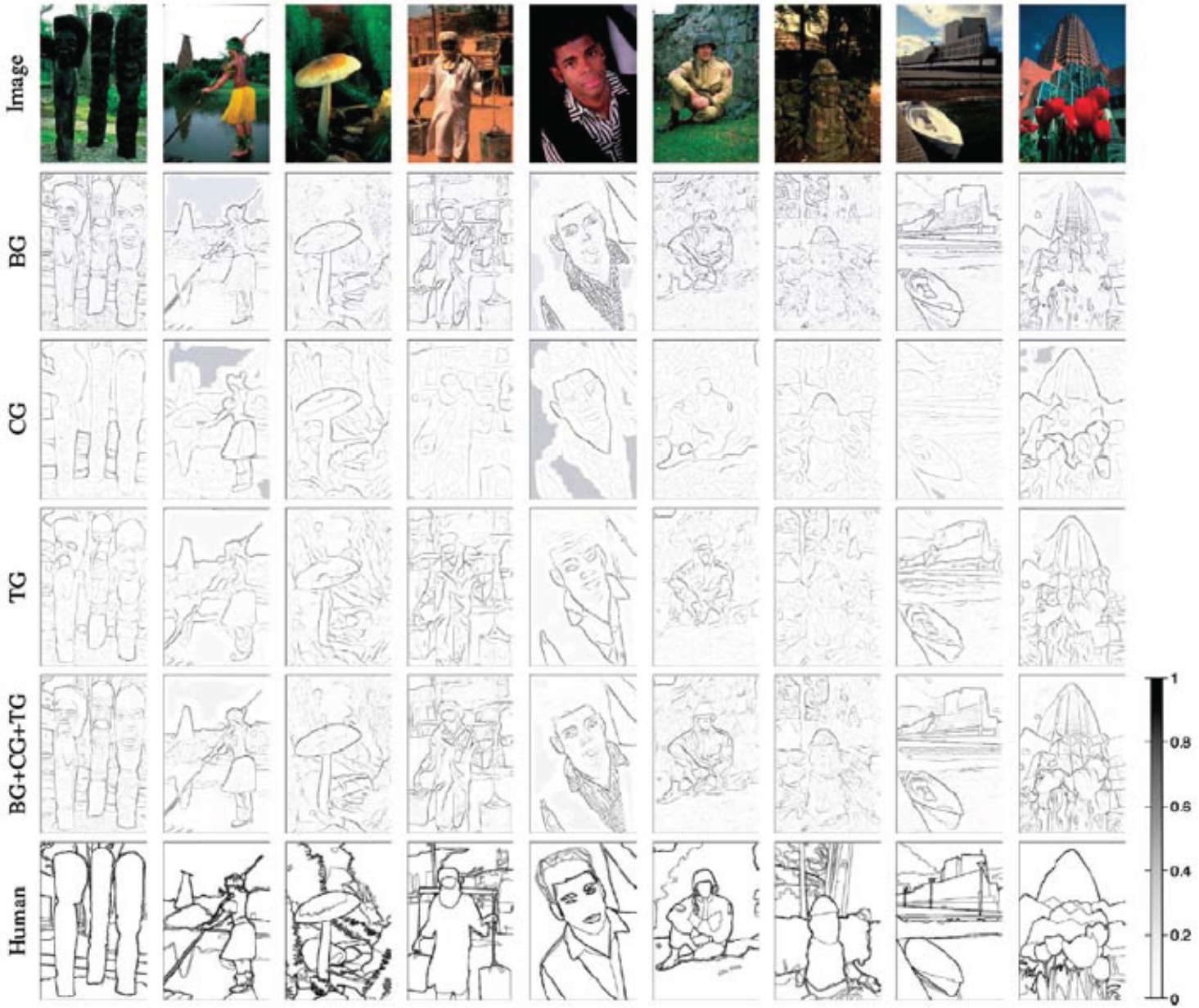


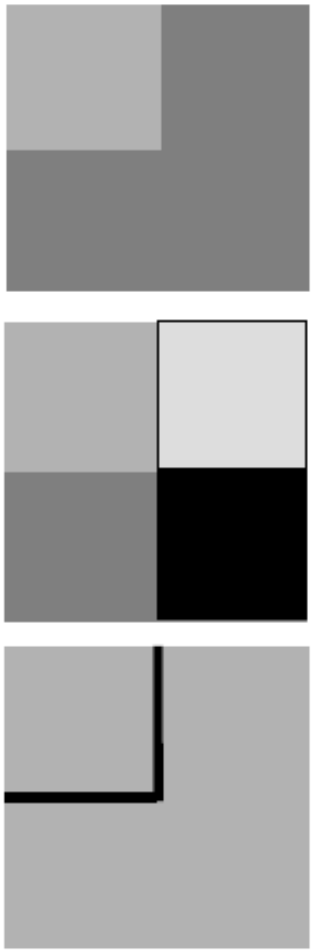
Figure 4.33 Combined brightness, color, texture boundary detector (Martin, Fowlkes, and Malik 2004) © 2004 IEEE. Successive rows show the outputs of the brightness gradient (BG), color gradient (CG), texture gradient (TG), and combined (BG+CG+TG) detectors. The final row shows human-labeled boundaries derived from a database of hand-segmented images (Martin, Fowlkes, Tal *et al.* 2001).

Outline

- Introduction
- Point and patches
- Edges
- Corners

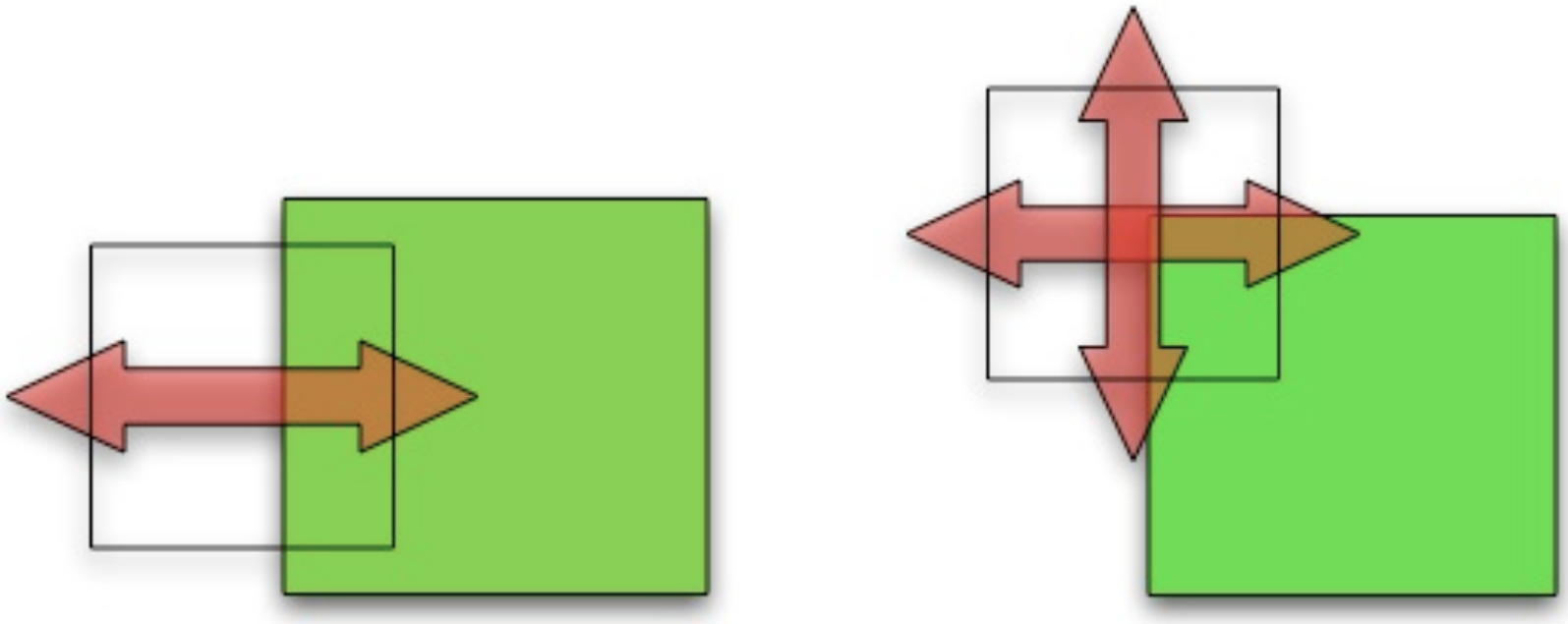
Corners

- What is a corner
  - A corner is an image location where two distinct image orientations occur in a local region.
  - Physically, image corners tend to arise for similar reasons as edges (e.g., changes of reflectance, surface orientation).
  - Corners are of interest for two main reasons
    1. Corners provide constrain 2 degrees of freedom in a pattern's location.
    2. Corners tend to persist across changes in viewpoint.

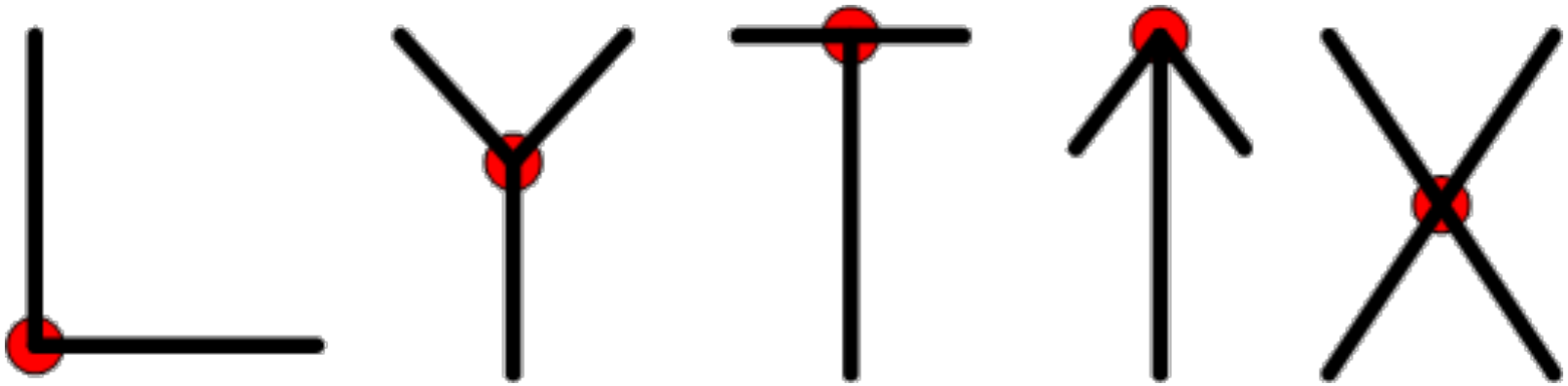


Source: Richard Wildes slides

Corners vs Edges



Corners types



Corners

- We can use also the gradient (Harris & Stephens, 1988)
  - Given that we are concerned with local measures of orientation, one approach is to calculate the local spatial derivatives E_x and E_y, using subscript notation for partial derivatives.
  - We choose to accumulate these measures over a neighborhood via summation and construct the matrix

$$M = \begin{pmatrix} \sum E_x^2 & \sum E_x E_y \\ \sum E_x E_y & \sum E_y^2 \end{pmatrix}$$

Corners

- We can use also the gradient (Harris & Stephens, 1988)
  - We choose to accumulate these measures over a neighborhood via summation and construct the matrix

$$M = G(\sigma) * (\nabla I)(\nabla I)^T = G(\sigma) * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

  - Because this matrix is symmetric, this can be simplified into

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

Corners

- We can use also the gradient (Harris & Stephens, 1988)
  - We choose to accumulate these measures over a neighborhood via summation and construct the matrix

$$M = G(\sigma) * (\nabla I)(\nabla I)^T = G(\sigma) * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

  - Because this matrix is symmetric, this can be simplified into

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

  - To provide three cases
    - If the region of interest is perfectly uniform, then the gradients are identically zero $\lambda_1 = \lambda_2 = 0$

Corners

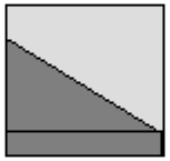
- We can use also the gradient (Harris & Stephens, 1988)
  - We choose to accumulate these measures over a neighborhood via summation and construct the matrix

$$M = G(\sigma) * (\nabla I)(\nabla I)^T = G(\sigma) * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

  - Because this matrix is symmetric, this can be simplified into

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

  - To provide three cases
    - If the region contains an ideal step edge, then there is only one gradient direction $\lambda_1 > 0, \lambda_2 = 0$



Corners

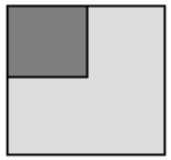
- We can use also the gradient (Harris & Stephens, 1988)
  - We choose to accumulate these measures over a neighborhood via summation and construct the matrix

$$M = G(\sigma) * (\nabla I)(\nabla I)^T = G(\sigma) * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

  - Because this matrix is symmetric, this can be simplified into

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

  - To provide three cases
    - If the region contains two orientations, then there are two gradient directions $\lambda_1 > 0, \lambda_2 > 0$



Corners

- We can use also the gradient (Harris & Stephens, 1988)
  - We choose to accumulate these measures over a neighborhood via summation and construct the matrix

$$M = G(\sigma) * (\nabla I)(\nabla I)^T = G(\sigma) * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

  - Because this matrix is symmetric, this can be simplified into

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

    - Magnitude of λ_1 and λ_2 are called eigenvalues and capture the edge strength
    - The direction is captured by the eigenvectors

Corners

- We can use also the gradient (Harris & Stephens, 1988)
  - Because this matrix is symmetric, this can be simplified into
    - Magnitude of λ_1 and λ_2 are called eigenvalues and capture the edge strength
    - The direction is captured by the eigenvectors

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

Corners

- We can use also the gradient (Harris & Stephens, 1988)
  - Because this matrix is symmetric, this can be simplified into
    - Magnitude of λ_1 and λ_2 are called eigenvalues and capture the edge strength
    - The direction is captured by the eigenvectors

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

$$\lambda_1+\lambda_2 = \mathrm{Tr}(M) = M_{1,1}+M_{2,2}$$

$$\lambda_1\lambda_2 = \mathrm{Det}(M) = M_{1,1}M_{2,2}-M_{1,2}M_{2,1}$$

Corners

- We can use also the gradient (Harris & Stephens, 1988)
  - Because this matrix is symmetric, this can be simplified into
    - Magnitude of λ_1 and λ_2 are called eigenvalues and capture the edge strength
    - The direction is captured by the eigenvectors

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

$$\begin{aligned} \lambda_1+\lambda_2 &= \mathrm{Tr}(M) &&= M_{1,1}+M_{2,2} \\ \lambda_1\lambda_2 &= \mathrm{Det}(M) &&= M_{1,1}M_{2,2}-M_{1,2}M_{2,1} \end{aligned}$$

    - The corner response is $R = \mathrm{Det}(M) - k\mathrm{Tr}^2(M)$

Corners

- We can use also the gradient (Harris & Stephens, 1988)
  - Because this matrix is symmetric, this can be simplified into
    - Magnitude of λ_1 and λ_2 are called eigenvalues and capture the edge strength
    - The direction is captured by the eigenvectors

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

$$\begin{aligned} \lambda_1+\lambda_2 &= \mathrm{Tr}(M) &&= M_{1,1}+M_{2,2} \\ \lambda_1\lambda_2 &= \mathrm{Det}(M) &&= M_{1,1}M_{2,2}-M_{1,2}M_{2,1} \end{aligned}$$

    - The corner response is $R = \mathrm{Det}(M) - k\mathrm{Tr}^2(M)$
    - Corners are all local maxima of R with $R>0$ and $\mathrm{Tr}>t$

Corners

- The discrete case

$\frac{1}{4}$

1	-2	1
-2	4	-2
1	-2	1

$\frac{1}{2}$

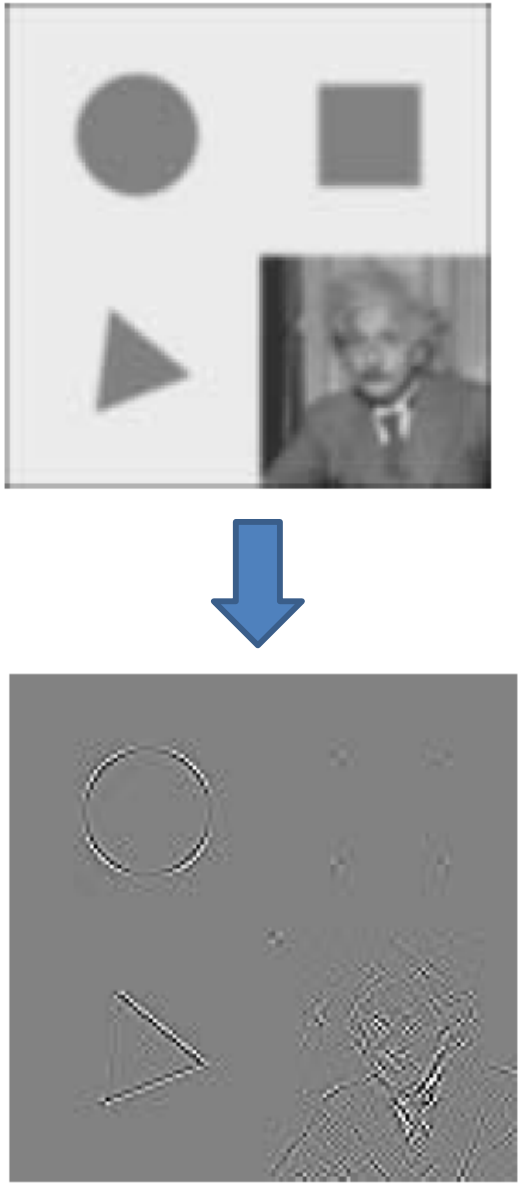
1	-2	1

Corners

- The discrete case

$$\frac{1}{4}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

$$\frac{1}{2}\begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$$



Summary

- Introduction
- Point and patches
- Edges
- Corners

universität innsbruck

OpenCV.
Feature extraction

Assoz.Prof. Antonio Rodríguez-Sánchez, PhD.

Sobel operator

- Remember from steerable filters
  - Consider a 2D circularly symmetric Gaussian function

$$G(x, y) = e^{-(x^2+y^2)}$$

  - The first ***x*** derivative is

$$G_1^{0^\circ} = \frac{\partial}{\partial x} e^{-(x^2+y^2)} = -2xe^{-(x^2+y^2)}$$

  - That same function rotated 90 degrees

$$G_1^{90^\circ} = \frac{\partial}{\partial y} e^{-(x^2+y^2)} = -2ye^{-(x^2+y^2)}$$

Source: Freeman and Adelson, 1991

Sobel operator

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * I$$

Sobel operator

$$G_x = \begin{bmatrix} -3 & 0 & +3 \\ -10 & 0 & +10 \\ -3 & 0 & +3 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ +3 & +10 & +3 \end{bmatrix}$$

Sobel operator

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <stdlib.h>
#include <stdio.h>

using namespace cv;

/** @function main */
int main( int argc, char** argv )
{

  Mat src, src_gray;
  Mat grad;
  char* window_name = "Sobel Demo - Simple Edge Detector";
  int scale = 1;
  int delta = 0;
  int ddepth = CV_16S;
  int c;

  /// Load an image
  src = imread( argv[1] );

  if( !src.data )
  { return -1; }
```

Sobel operator

```
if( !src.data )
{ return -1; }

GaussianBlur( src, src, Size(3,3), 0, 0, BORDER_DEFAULT );

/// Convert it to gray
cvtColor( src, src_gray, CV_BGR2GRAY );

/// Create window
namedWindow( window_name, CV_WINDOW_AUTOSIZE );

/// Generate grad_x and grad_y
Mat grad_x, grad_y;
Mat abs_grad_x, abs_grad_y;

/// Gradient X
//Scharr( src_gray, grad_x, ddepth, 1, 0, scale, delta, BORDER_DEFAULT );
Sobel( src_gray, grad_x, ddepth, 1, 0, 3, scale, delta, BORDER_DEFAULT );
convertScaleAbs( grad_x, abs_grad_x );
```

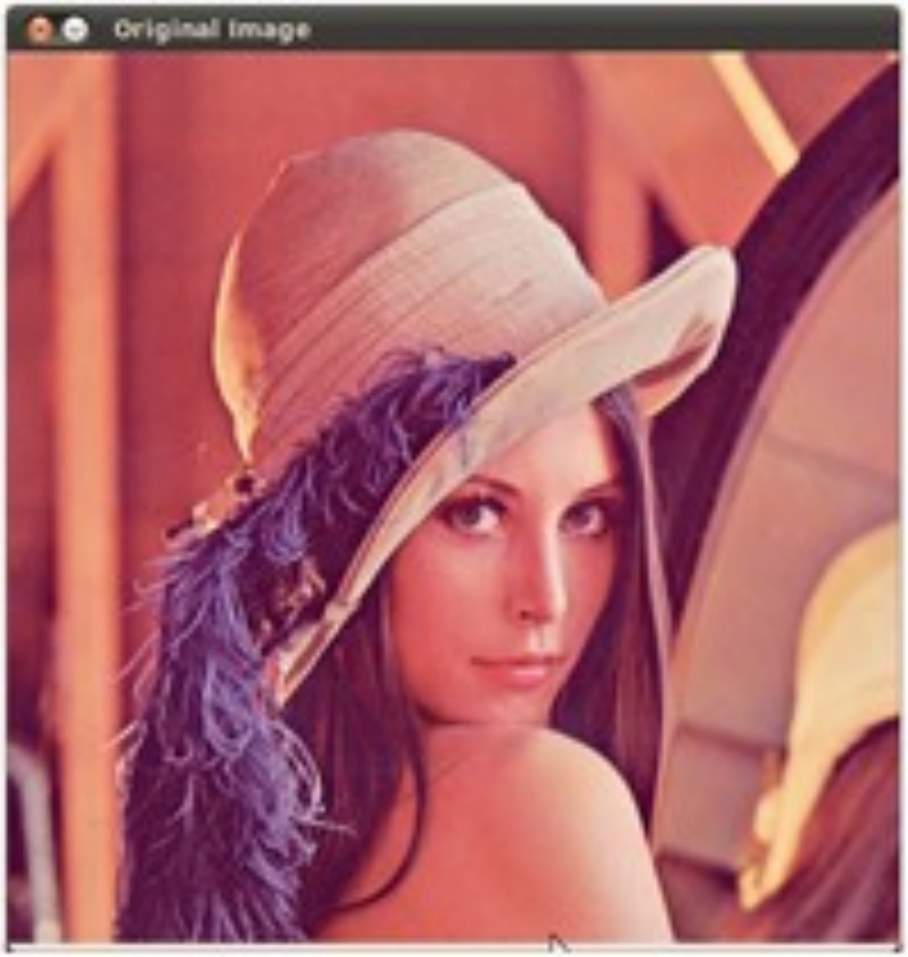
Sobel operator

```
if( !src.data )
{ return -1; }

GaussianBlur( src, src, Size(3,3), 0, 0, BORDER_DEFAULT );

/// Convert it to gray
cvtColor( src, src_gray, CV_BGR2GRAY );

/// Create window
namedWindow( window_name, CV_WINDOW_AUTOSIZE );

/// Generate grad_x and grad_y
Mat grad_x, grad_y;
Mat abs_grad_x, abs_grad_y;

/// Gradient X
//Scharr( src_gray, grad_x, ddepth, 1, 0, scale, delta, BORDER_DEFAULT );
Sobel( src_gray, grad_x, ddepth, 1, 0, 3, scale, delta, BORDER_DEFAULT );
convertScaleAbs( grad_x, abs_grad_x );
```

Sobel operator

```
/// Gradient X
//Scharr( src_gray, grad_x, ddepth, 1, 0, scale, delta, BORDER_DEFAULT );
Sobel( src_gray, grad_x, ddepth, 1, 0, 3, scale, delta, BORDER_DEFAULT );
convertScaleAbs( grad_x, abs_grad_x );

/// Gradient Y
//Scharr( src_gray, grad_y, ddepth, 0, 1, scale, delta, BORDER_DEFAULT );
Sobel( src_gray, grad_y, ddepth, 0, 1, 3, scale, delta, BORDER_DEFAULT );
convertScaleAbs( grad_y, abs_grad_y );

/// Total Gradient (approximate)
addWeighted( abs_grad_x, 0.5, abs_grad_y, 0.5, 0, grad );

imshow( window_name, grad );

waitKey(0);

return 0;
}
```

Sobel operator



Laplace operator

- Band-pass filtering
  - Filter low and high frequencies
  - Remember the Gaussian equation

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}},$$

  - If we take the Laplacian

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

Note: there is a typo in Szeliski book

  - We obtain the Laplacian of Gaussian filter

$$\nabla^2 G(x, y; \sigma) = \left(\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2} \right) G(x, y; \sigma),$$

Laplace operator

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <stdlib.h>
#include <stdio.h>

using namespace cv;

/** @function main */
int main( int argc, char** argv )
{
  Mat src, src_gray, dst;
  int kernel_size = 3;
  int scale = 1;
  int delta = 0;
  int ddepth = CV_16S;
  char* window_name = "Laplace Demo";

  int c;

  /// Load an image
  src = imread( argv[1] );

  if( !src.data )
    { return -1; }
```

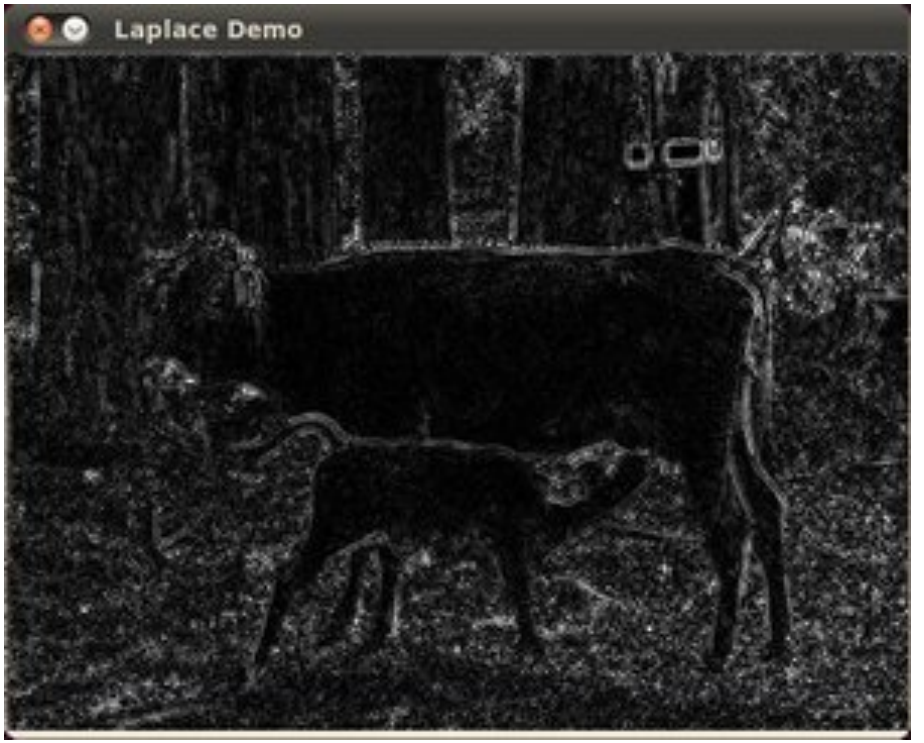
Laplace operator

```
if( !src.data )
  { return -1; }

/// Remove noise by blurring with a Gaussian filter
GaussianBlur( src, src, Size(3,3), 0, 0, BORDER_DEFAULT );

/// Convert the image to grayscale
cvtColor( src, src_gray, CV_BGR2GRAY );

/// Create window
namedWindow( window_name, CV_WINDOW_AUTOSIZE );

/// Apply Laplace function
Mat abs_dst;

Laplacian( src_gray, dst, ddepth, kernel_size, scale, delta, BORDER_DEFAULT );
convertScaleAbs( dst, abs_dst );

/// Show what you got
imshow( window_name, abs_dst );

waitKey(0);

return 0;
```

Laplace operator

```
if( !src.data )
  { return -1; }

/// Remove noise by blurring with a Gaussian filter
GaussianBlur( src, src, Size(3,3), 0, 0, BORDER_DEFAULT );

/// Convert the image to grayscale
cvtColor( src, src_gray, CV_BGR2GRAY );

/// Create window
namedWindow( window_name, CV_WINDOW_AUTOSIZE );

/// Apply Laplace function
Mat abs_dst;

Laplacian( src_gray, dst, ddepth, kernel_size, scale, delta, BORDER_DEFAULT );
convertScaleAbs( dst, abs_dst );

/// Show what you got
imshow( window_name, abs_dst );

waitKey(0);

return 0;
```

Laplace operator



Harris corner detector

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>
#include <stdio.h>
#include <stdlib.h>

using namespace cv;
using namespace std;
```

```
/// Global variables
Mat src, src_gray;
int thresh = 200;
int max_thresh = 255;

char* source_window = "Source image";
char* corners_window = "Corners detected";

/// Function header
void cornerHarris_demo( int, void* );
```

Harris corner detector

```
/// Function header
void cornerHarris_demo( int, void* );

/** @function main */
int main( int argc, char** argv )
{
  /// Load source image and convert it to gray
  src = imread( argv[1], 1 );
  cvtColor( src, src_gray, CV_BGR2GRAY );

  /// Create a window and a trackbar
  namedWindow( source_window, CV_WINDOW_AUTOSIZE );
  createTrackbar( "Threshold: ", source_window, &thresh, max_thresh, cornerHarris_demo );
  imshow( source_window, src );

  cornerHarris_demo( 0, 0 );

  waitKey(0);
  return(0);
}
```

Harris corner detector

```
/** @function cornerHarris_demo */
void cornerHarris_demo( int, void* )
{

  Mat dst, dst_norm, dst_norm_scaled;
  dst = Mat::zeros( src.size(), CV_32FC1 );

  /// Detector parameters
  int blockSize = 2;
  int apertureSize = 3;
  double k = 0.04;

  /// Detecting corners
  cornerHarris( src_gray, dst, blockSize, apertureSize, k, BORDER_DEFAULT );

  /// Normalizing
  normalize( dst, dst_norm, 0, 255, NORM_MINMAX, CV_32FC1, Mat() );
  convertScaleAbs( dst_norm, dst_norm_scaled );

  /// Drawing a circle around corners
  for( int j = 0; j < dst_norm.rows ; j++ )
     { for( int i = 0; i < dst_norm.cols; i++ )
          {
            if( (int) dst_norm.at<float>(j,i) > thresh )
              {
               circle( dst_norm_scaled, Point( i, j ), 5,  Scalar(0), 2, 8, 0 );
              }
          }
     }
```

Harris corner detector

```
/** @function cornerHarris_demo */
void cornerHarris_demo( int, void* )
{

  Mat dst, dst_norm, dst_norm_scaled;
  dst = Mat::zeros( src.size(), CV_32FC1 );

  /// Detector parameters
  int blockSize = 2;
  int apertureSize = 3;
  double k = 0.04;

  /// Detecting corners
  cornerHarris( src_gray, dst, blockSize, apertureSize, k, BORDER_DEFAULT );

  /// Normalizing
  normalize( dst, dst_norm, 0, 255, NORM_MINMAX, CV_32FC1, Mat() );
  convertScaleAbs( dst_norm, dst_norm_scaled );

  /// Drawing a circle around corners
  for( int j = 0; j < dst_norm.rows ; j++ )
     { for( int i = 0; i < dst_norm.cols; i++ )
          {
            if( (int) dst_norm.at<float>(j,i) > thresh )
              {
               circle( dst_norm_scaled, Point( i, j ), 5,  Scalar(0), 2, 8, 0 );
              }
          }
     }
```

Harris corner detector

```
/** @function cornerHarris_demo */
void cornerHarris_demo( int, void* )
{

  Mat dst, dst_norm, dst_norm_scaled;
  dst = Mat::zeros( src.size(), CV_32FC1 );

  /// Detector parameters
  int blockSize = 2;
  int apertureSize = 3;
  double k = 0.04;

  /// Detecting corners
  cornerHarris( src_gray, dst, blockSize, apertureSize, k, BORDER_DEFAULT );

  /// Normalizing
  normalize( dst, dst_norm, 0, 255, NORM_MINMAX, CV_32FC1, Mat() );
  convertScaleAbs( dst_norm, dst_norm_scaled );

  /// Drawing a circle around corners
  for( int j = 0; j < dst_norm.rows ; j++ )
     { for( int i = 0; i < dst_norm.cols; i++ )
          {
            if( (int) dst_norm.at<float>(j,i) > thresh )
              {
               circle( dst_norm_scaled, Point( i, j ), 5,  Scalar(0), 2, 8, 0 );
              }
          }
     }
```

Harris corner detector