Università di Genova | MaLGa

Feature Detection

Digital Signal and Image Processing

Francesca Odone

Outline

Enhancement filters and derivatives

Edge detection

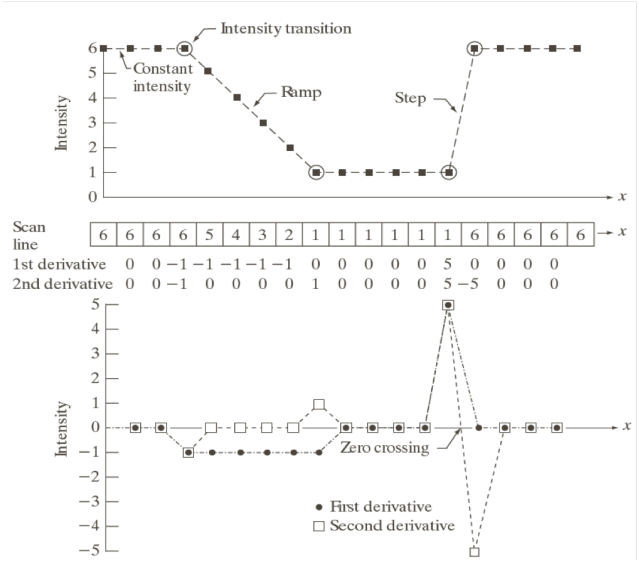
Corner detection

Closing remarks

Computing image derivatives

- Enhancement filters highlight abrupt transitions in intensity, they correspond to high pass filtering in Fourier
- We will use them to estimate finite differences (discrete derivatives), in preparation to *feature detection* in images
- We will need to cope with the fact differentiation enhances interesting discontinuities, but also noise

Computing image derivatives: example



Computing image derivatives

1st derivative:

- Zero in constant areas
- not zero on ramps

2nd derivative:

- Zero in constant areas
- Zero on ramps with constant slope
- not zero at the beginning and the end of ramps

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

First derivatives

It may be useful to compute

- Image gradient $\nabla f = grad(f) = [g_x, g_y] = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- Magnitude of the gradient $M(x, y) = \sqrt{g_x^2 + g_y^2}$

  Notice: it's not linear (we cannot implement it with a kernel)

Examples of 1st derivative spatial filters:

$$g_x = K * f$$
$$g_y = K^\top * f$$

Roberts filter

$$g_x = R_x * f$$
$$g_y = R_y * f$$

Second derivatives

- Zero-crossings are good indicators of sharp variations
- The Laplacian is a 2D isotropic measure of the 2nd spatial derivative of an image

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

- The Laplacian, as a sum of derivatives (which are linear) is linear and can be represented through an appropriate kernel

0	1	0
1	-4	1
0	1	0

Rotation invariant for 90 deg increments

1	1	1
1	-8	1
1	1	1

Rotation invariant for 45 deg increments

What do we do with derivatives?

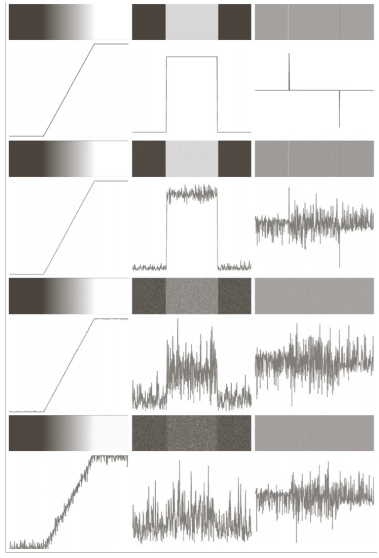
- As we will see later in the course derivatives allow us to highlight signal discontinuities (at the basis of feature detection)
- Gradient: edge detection, corners, image sharpening
- Laplacian: edge detection (zero-crossing algorithms), blob-like features, image sharpening

Derivatives and noise

- So far, we discussed the ideal (noise-free) case
- in the presence of noise, enhancement filters also enhance noise

General scheme:

1. Smoothing the image with a low pass filter
2. Computing the derivative



Derivatives and noise

From the properties of convolution:

$$\frac{\partial^n}{\partial u}(k * f) = \frac{\partial^n k}{\partial u} * f$$

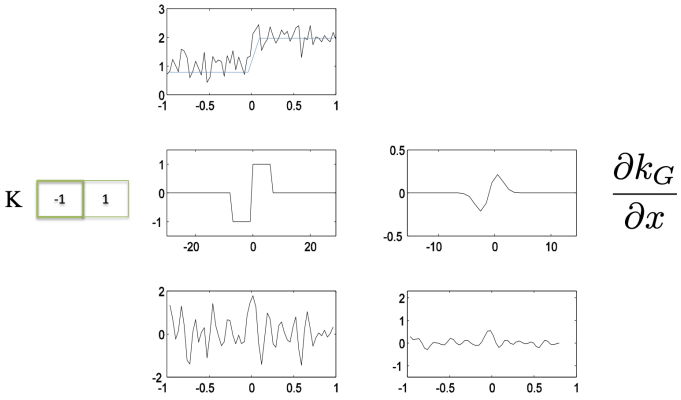
Therefore the previous scheme is equivalent to

1. Computing the derivative of the filter
2. Smooth the image with the derivative of the filter

This alternative has some advantages:

- Since the kernels are smaller than the image, we will have fewer computations
- Often the derivative of kernels can be pre-calculated and this would give us only one convolution

Derivative of the Gaussian



Example: the Sobel operator

Sx

-1	-2	-1
0	0	0
1	2	1

$$g_x = S_x * f$$
$$g_y = S_y * f$$
$$S_y = S_x^\top$$

It can be decomposed as the product of a weighted average and a differentiation filter

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Example: the Sobel operator

Sx

-1	-2	-1
0	0	0
1	2	1

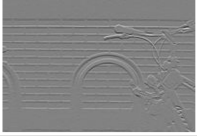
$$g_x = S_x * f$$

$$g_y = S_y * f$$

$$S_y = S_x^\top$$

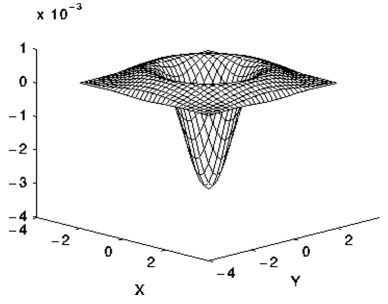
$S_x * f$

$S_y * f$



M

Example: LoG (Laplacian of Gaussian) operator



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

Non linear filters

- In contrast to frequency filtering, in the space domain we could design non-linear filters
- A well-known example is the median filter, used to contrast the salt-and-pepper noise
- another interesting non linear filter is the bilateral filter (smoothing but edge-preserving filter)

Outline

Enhancement filters and derivatives

Edge detection

Corner detection

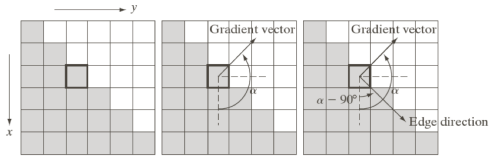
Closing remarks

Edges

Edges

Edges: definition

- Edge position
- Edge normal
- Edge strength



- We refer mainly to step edges
- The *ideal* step edge models the type of structure we are interested in

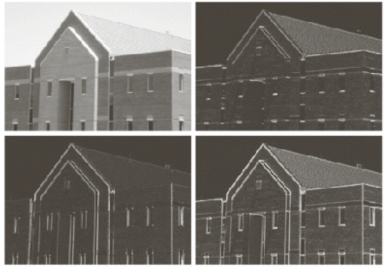
$$I(x) = \begin{cases} 0 & x < 0 \\ A & x \geq 0 \end{cases}$$

Criteria for optimal edge detection

- *Good detection*. The optimal detector must minimise the probability of false positives (spurious edges) as well as that of missing real edges?
- *Good Localization*. The edges detected must be as close as possible to the true edges?
- *Single response constraint*. The detector must return one point only for each true edge point

Edge detection by thresholding: Sobel Algorithm

Edge enhancement
(previous class)



Edge localization,
for instance by
thresholding

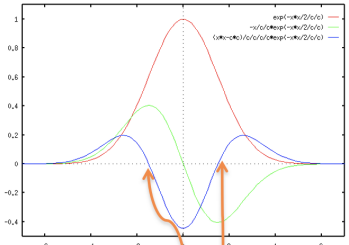
Edge detection by zero crossing

- The simplest way is to compute the gradient magnitude and then threshold it
- A more accurate way to compute the gradient maxima is to look for **zero-crossing** of the second derivative (Marr, Hildreth)

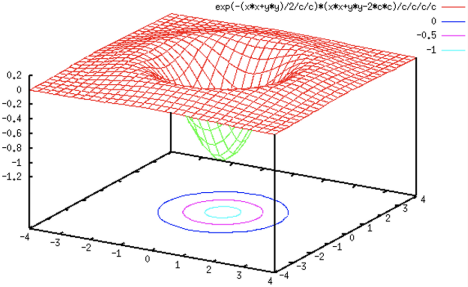
$$\nabla \cdot J_\sigma(\mathbf{x}) = [\nabla^2 G_\sigma](\mathbf{x}) * I(\mathbf{x})$$

- This is the **Laplacian of Gaussian (LoG)**

Edge detection by zero crossing



zero crossing



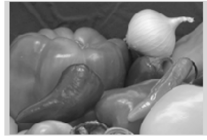
LoG mask =

$$\begin{matrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 2 & -16 & 2 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{matrix}$$

Edge detection by zero crossing

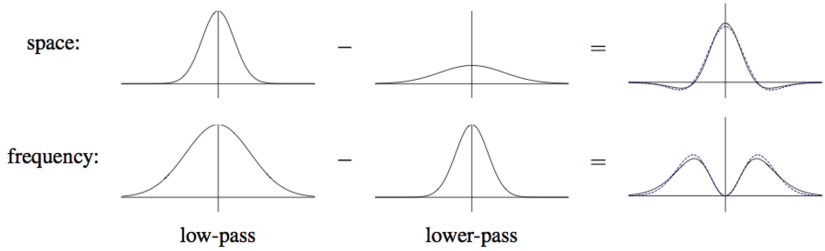
- Filter the image with a LoG mask
- Locate edge elements then the sign between adjacent elements changes
- This approach produces edge chains in closed loops *(why?)*

- It is quite common to use a threshold
  - If adjacent pixels have a different sign and their absolute value exceeds a threshold



Approximating the LoG

- It is common practice to approximate a LoG filter with a Difference of Gaussians (DoG) of different amplitude.
- This is particularly useful in multi-scale models



Edge detection: Canny Algorithm

- The Canny edge detector is still one of the most largely used algorithms.
- It implements an approximation of the optimal step edge detector
- The algorithm
  1. Edge enhancement
  2. Non-maxima suppression
  3. Hysteresis thresholding

Edge detection: Canny Algorithm

- We compute the image gradient J taking into account the presence of noise (previous class)

- We compute the edge intensity

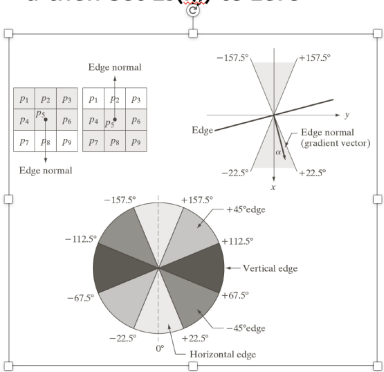
$$E_s(i,j) = \sqrt{J_x^2(i,j) + J_y^2(i,j)}$$

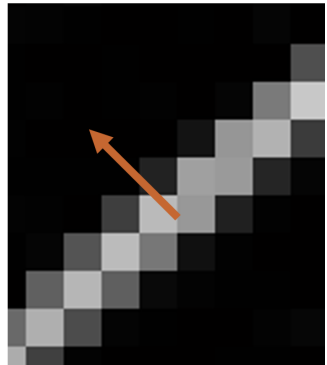
- and estimate the edge normal

$$E_o(i,j) = \arctan \frac{J_y}{J_x}$$

Edge detection: Canny Algorithm

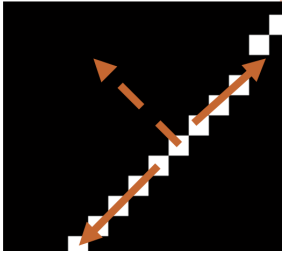
- For each pixel *(i,j)*, given a sampling of directions (eg D={0,45,90,135}):
  - Look for direction *d* in D that best approximates $E_O(i,j)$;
  - If $E_S(i,j)$ is smaller than its neighbours in the direction *d* then set $E_S(i,j)$ to zero



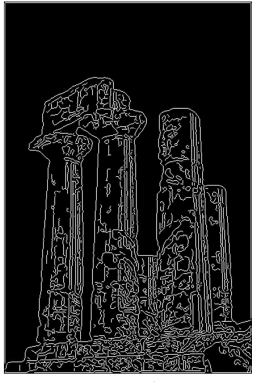


Edge detection: Canny Algorithm

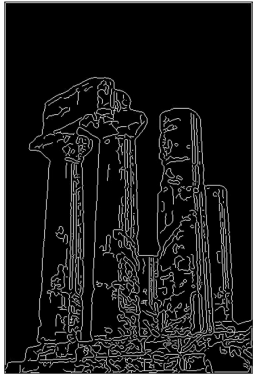
- Given two thresholds $\tau_1 > \tau_2$
- For each pixel (i,j) if Es(i,j)> t_1
  - Starting from Es(i,j) follow the chains of connected local maxima in both directions perpendicular to the edge normal as long as Es(k,h)> t_2
  - Mark all visited points and save a list of the locations of all points in the connected contour found
- The output is a set of edge chains



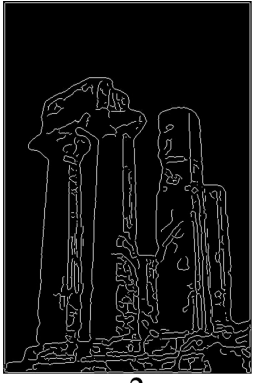
Canny Algorithm: the role of σ



σ=1



σ=1.5



σ=2

Outline

Enhancement filters and derivatives

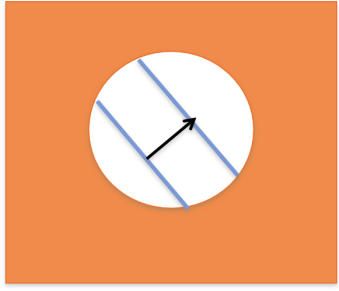
Edge detection

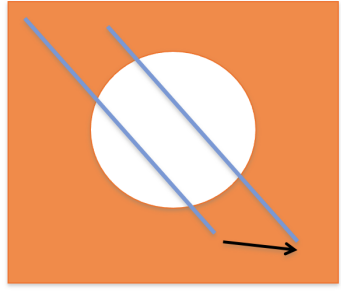
Corner detection

Closing remarks

Good features

- Patches with large contrast changes are easier to localize
- Straight line segments with a single orientation suffer from the **aperture problem**





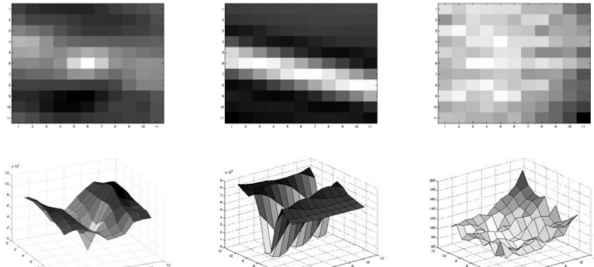
Good features

- We observe how patches with gradients in at least two (significantly) different orientations are the easier to localize
- This can be formalized by analysing a simple matching criterium (Summed Square Difference)
- In particular we use it to check how stable the patch is wrt small variations in position **u** (SSD **autocorrelation function**):

$$E_{AC}(\mathbf{u}) = \sum_i [I(\mathbf{x}_i + \mathbf{u}) - I(\mathbf{x}_i)]^2$$

Good features

(a)



Corner detection: algorithm sketch

- Given an image point q we consider a neighborhood Nq and compute its auto-correlation matrix

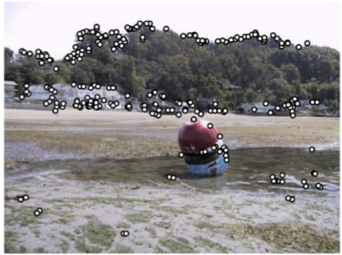
$$A_q = \begin{bmatrix} \sum_{p \in N_q} I_x^2(p) & \sum_{p \in N_q} I_x(p) I_y(p) \\ \sum_{p \in N_q} I_x(p) I_y(p) & \sum_{p \in N_q} I_y^2(p) \end{bmatrix}$$

- We diagonalise it

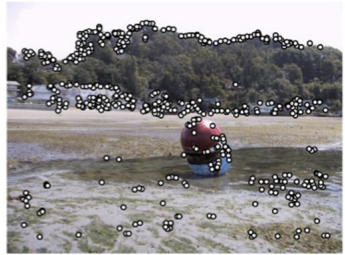
$$\text{eig}(A) = [\lambda_1, \lambda_0] \quad \lambda_1 > \lambda_0$$

$$A = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_0 \end{bmatrix}$$

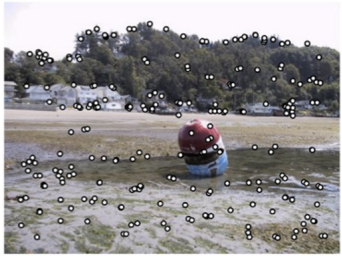
Corner detection results



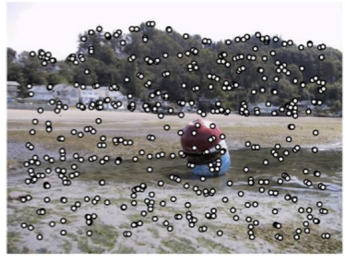
(a) Strongest 250



(b) Strongest 500



(c) ANMS 250, $\epsilon = 24$



(d) ANMS 500, $\epsilon = 16$

Outline

Enhancement filters and derivatives

Edge detection

Corner detection

Closing remarks

Beyond edges and corners

Image features are local, meaningful, detectable part of the image

(a)

(b)

(c)

(d)