UNIVERSITÉ DE GENÈVE
CENTRE UNIVERSITAIRE D'INFORMATIQUE

Université de Genève

Départment d'informatique

Advanced Image Processing

Spatial Filtering

TP Class N° 2

March 14, 2019

Notes: The details of the analytical calculations must be included in the report. If you do not use an Equation Editor (for Word, Latex or similar) you can easily scan a handwritten page (readable!) and put it in the report.

1 Convolution

Convolution of functions f and g is by definition:

$$(f * g)(t) \equiv \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

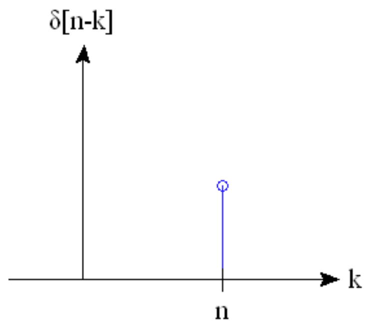
The discrete convolution of f and g is given by:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

By the sifting property of impulses, any signal can be decomposed in terms of an infinite sum of shifted, scaled impulses.

$$f(n) = \sum_{k=-\infty}^{\infty} f(k)\delta(n-k)$$

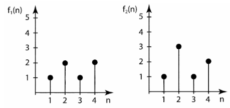
The function $\delta(n-k)$ peaks up where $n = k$.



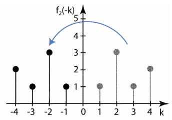
Let there be two functions:

$$\begin{aligned} f_1(n) &= \delta(n-1) + 2\delta(n-2) + \delta(n-3) + 2\delta(n-4) \\ f_2(n) &= \delta(n-1) + 3\delta(n-2) + \delta(n-3) + 2\delta(n-4) \end{aligned}$$

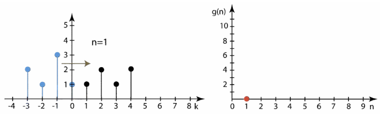
The result of a convolution $(f_1 * f_2)(n)$ between these two functions can be seen in Figure 1.



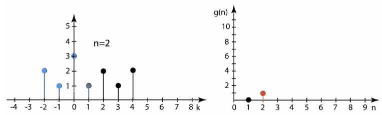
(a) Signals $f_1(n)$ and $f_2(n)$



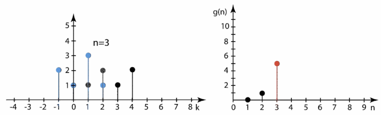
(b) Flip $f_2(n)$



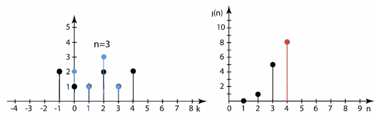
(c) Start



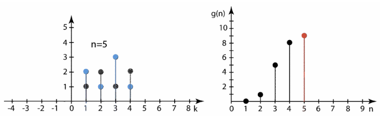
(d) $1 \cdot 1 = 1$



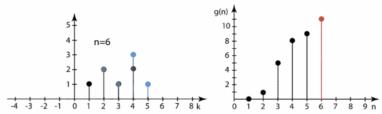
(e) $1 \cdot 3 + 2 \cdot 1 = 5$



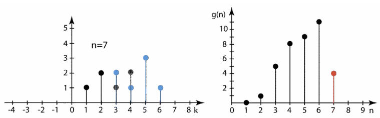
(f) $1 \cdot 1 + 2 \cdot 3 + 1 \cdot 1 = 8$



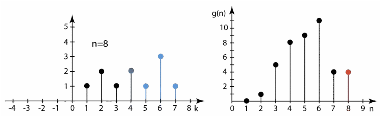
(g) $1 \cdot 2 + 2 \cdot 1 + 1 \cdot 3 + 2 \cdot 1 = 9$



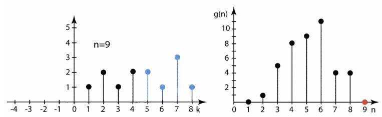
(h) $2 \cdot 2 + 1 \cdot 1 + 2 \cdot 3 = 11$



(i) $1 \cdot 2 + 2 \cdot 1 = 4$



(j) $2 \cdot 2 = 4$



(k) Result

Figure 1: Discrete Convolution Example

Exercise 1. For the f_1 and f_2 given below determine $f_1 * f_2$ analytically ("by had" **without** Matlab) similarly as illustrated in Figure 1.

(a)

$$\begin{aligned} f_1 &= [1, 0, 0] \\ f_2 &= [0, 1, 0] \end{aligned}$$

(b)

$$\begin{aligned} f_1 &= [0, 1, 0, 0, 0, 0] \\ f_2 &= [1, 1, 1, 1, 1, 1] \end{aligned}$$

(c)

$$\begin{aligned} f_1(n) &= \delta(n) + 2\delta(n-1) + 3\delta(n-2) \\ f_2(n) &= \delta(n+1) + \delta(n) + \delta(n-1) + \delta(n-2) \end{aligned}$$

2 Spatial Filtering

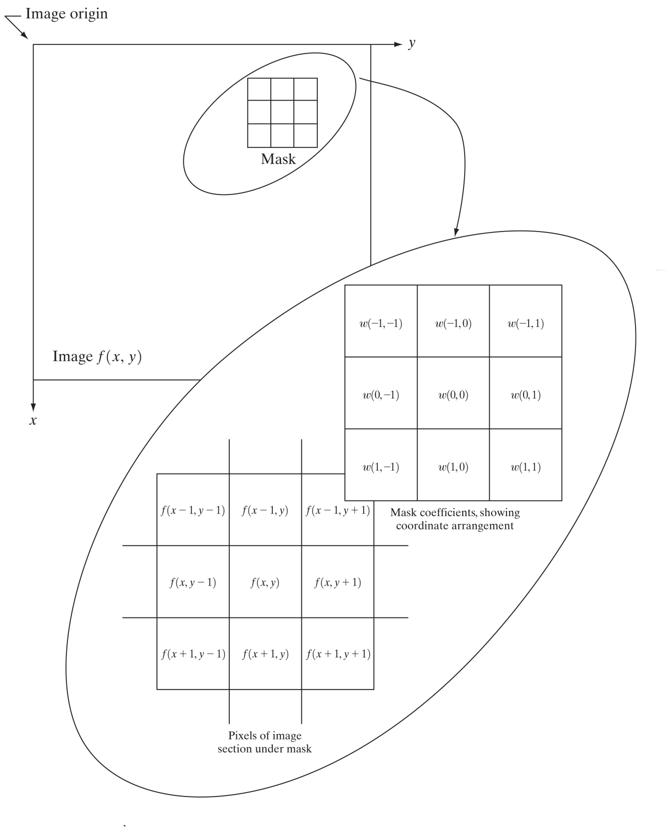


Figure 2: Example of linear filtering with 3×3 Region of Interest and a 3×3 kernel

The basic working of spacial filtering is illustrated in Figure 2. It simply consists of moving the filter mask from point to point in the image. At each point, the *response* of the filter is calculated. For linear spatial filtering, the response is the sum of the products of the filter coefficients and the corresponding image pixels currently under the filter mask. For the 3×3 filter in Figure 2, the response r is thus:

$$\begin{aligned} r =& w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \\ & ... + w(1,1)f(x+1,y+1) \end{aligned}$$

This notation, for a $m \times n$ sized mask is often informally denoted as

$$\begin{aligned} r =& w_1 f_1 + w_2 f_2 + ... + w_{mn} f_{mn} \\ r =& \sum_{i=1}^{mn} w_i f_i \end{aligned}$$

Which is the sum of the products of the filter coefficients against the pixels directly under the mask. Note that in this particular example the filter is centered at (x, y). If the operation is linear, it is in fact identical to *discrete convolution*. An important consideration is what to do at the border of an image, when part of the filter mask falls outside the image:

- One can simply discard the border pixels, in that case, the output image will be smaller than the input.

- If the output image is to be the same size as the input image, one can pad the border either with 1's or 0's.
- One can replicate the image values, so that the image becomes larger. After discarding the border pixels, the output image will be of identical size as the input image. This last method is known as *circular convolution*.

2.1 Low Pass Filtering

An operator that cuts of high frequencies while passing through low-frequencies, is a *low-pass filter*. Examples of low-pass filters are the *averaging* operator or *"box"* filter and the *Gaussian* filter.

Exercise 2.

(a) Implement and test the averaging filter for any image. Visualise the image before and after filtering. What effects can you observe? An averaging filter can be implemented by convolving the following mask:

$$\mathbf{h} = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

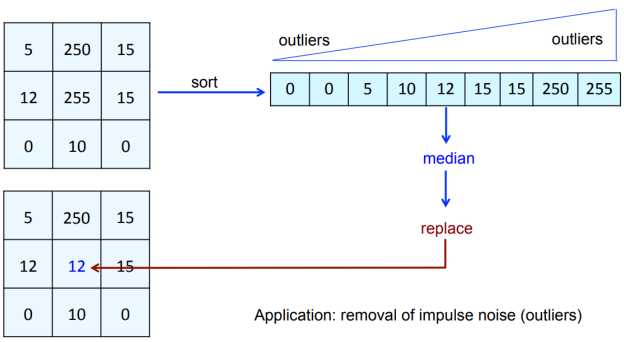
(b) Implement and test the Gaussian filter for any image. Visualise the image before and after filtering. What effects can you observe? The Gaussian filter can be approximated by:

$$\mathbf{h} = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

(c) How could these matrices be modified to increase the blurring effect, i.e. remove more high frequency components.

2.2 Order-Statistic Filtering

Order-statistics filters are non-linear filters whose response is based on the ranking (or ordering) of the pixels currently under the filter mask. The central pixel is then replaced by the result of the ranking operation. The best known example is the so called *median* filter. This filter replaces the central pixel by the median value of the region around it. Median filters are particularly capable in dealing with *salt & pepper* noise.

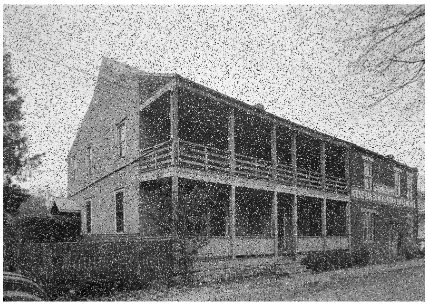


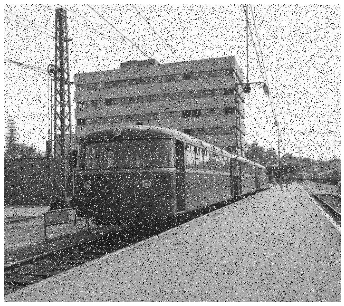
Exercise 3. You are given two noisy images *tp2_001.jpg* and *tp2_002.jpg* corrupted by "impulse" noise. Defining the optimal parameters by yourself try to remove the noise with:

(a) an averaging filter.

(b) a median filter.

Report all results and used parameters of filtering.


(a) tp2_001.jpg


(b) tp2_002.jpg

2.3 High Pass Filtering

High pass filtering cuts of low frequency and only passes on high frequency content. Informally, you have seen applications of such filters in *edge detection* and in *sharpening* algorithms.

First order derivative

Before jumping into application we will have a look at some of the fundamental properties of discrete derivatives and digital imaging.

The basic definition for a discrete 1 dimensional first and second order derivative are:

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

Exercise 4. Given the following discrete signal:

$$x = [5, 7, 6, 5, 4, 3, 2, 1, 0, 5, 4, 3, 2, 1, 0, 0, 0, 6, 0, 0, 0, 0, 1, 3, 1, 0, 0, 0, 0, 7, 7, 7, 7]$$

(a) Determine the first and second order derivative

(b) Plot the original and the derivatives

(c) What properties of the first and second order derivative can you see?

Second order derivative

The second order derivative is known as the *Laplacian* and it is used extensively in image enhancement. Formally it is defined as follows:

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

As derivatives of any kind are linear operations, the Laplacian is also a linear operator. In order to be useful for digital image processing, we need to express the Laplacian in the discrete form. In x-direction:

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

and in y-direction:

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

The final digital form is then obtained by summing these components:

$$\nabla^2 f(x, y) = f(x-1, y) + f(x+1, y) + f(x, y-1) + f(x, y+1) - 4f(x, y) \tag{1}$$

Exercise 5.

(a) Implement by yourself and test (apply to any image) the Laplacian filter based on Equation 1 and a 3×3 neighbourhood. Visualise the image before and after filtering. What effects can you observe?

(b) The Laplacian filter can be implemented by:

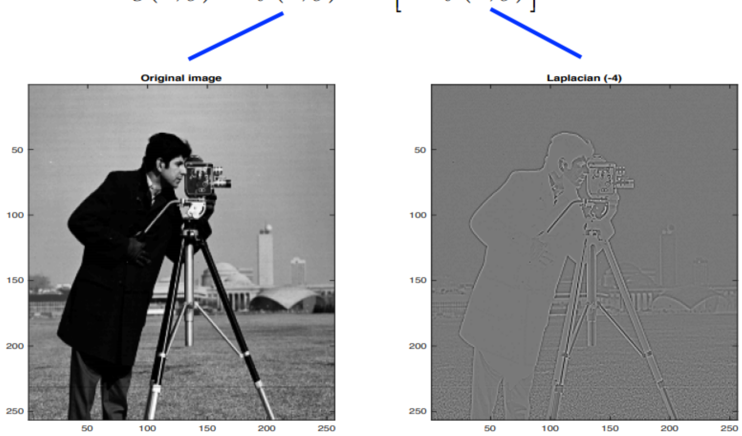
$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

Implement and test (apply to any image) the above Laplacian filter implementations. Visualise the image before and after filtering. What effects can you observe?

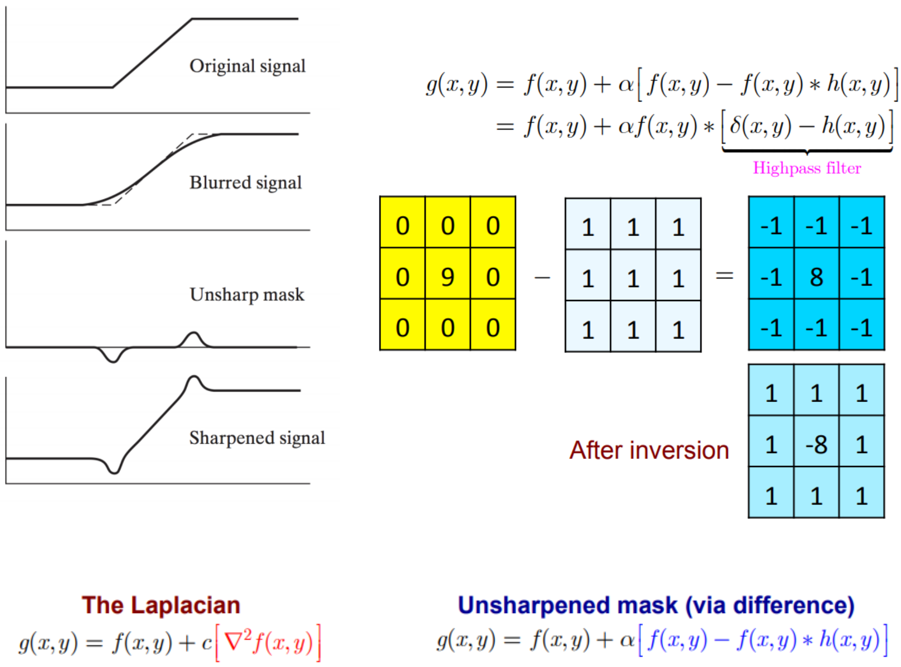
2.4 Image Sharpening

Sharpening: image enhancement by highlighting sharp intensity transitions and deemphasizing regions of slowly varying regions:

$$g(x,y) = f(x,y) + c\left[\nabla^2 f(x,y)\right]$$



Similar way to achieve the same effect is unsharp masking:



Exercise 6. Take any image. Blur this image with the box filter of size 3×3.

(a) Defining the optimal parameters perform image sharpening using the Gaussian filter.

(b) Defining the optimal parameters perform image sharpening using the Laplacian filter.

Display blurred and enhanced images. Comment the quality of the enhanced images based on the visual quality and based on the *mse* between the original and improved images.

3 Covariance Matrix and Eigenvectors

The covariance between two random variables is a measure how much the two numbers covariate (to covariate: if one variable fluctuates, the other variable will fluctuate proportionally, and vice verse).

$$\begin{aligned} cov(X_1, X_2) &= E(X_1 X_2) - E(X_1)E(X_2) \\ &= E(X_1 X_2) - \mu_{X_1}\mu_{X_2} \end{aligned}$$

As the two elements in the vector $\mathbf{x}$ are independently generated their covariance is per definition **zero**:

$$cov(X_1, X_2) = 0$$

Let $Y = (X_1; X_2)$. Then the covariance matrix of Y is defined as:

$$\mathbf{C}_Y = \begin{bmatrix} var[X_1] & cov[X_1, X_2] \\ cov[X_1, X_2] & var[X_2] \end{bmatrix}$$

Associated with a covariance matrix and an expectation vector is an ellipsoid (in two dimensions: an ellipse) that can be regarded as a region of uncertainty around the expectation.
If realisations $\mathbf{y}$ from random vector Y with covariance matrix $\mathbf{C}_Y$ are linearly transformed:

$$\mathbf{y}' = \mathbf{A}\mathbf{y}$$

Then the covariance matrix and expectation for the resulting vector $\mathbf{y}'$ are:

$$\mathbf{C}_{Y'} = \mathbf{A}\mathbf{C}_Y\mathbf{A}^T$$
$$\boldsymbol{\mu}_{Y'} = \mathbf{A}\boldsymbol{\mu}_Y$$

Exercise 7.

1. Generate 2x100 realisations using `randn` with zero mean and unit variance. These values, $\mathbf{x}$, are realisations of two random variables X_1 and X_2 that are Gaussian distributed, i.e $X_i \sim \mathcal{N}(0,1)$.
   (a) Plot X_1 and X_2.
   (b) What is the expectation of X_i, $E[X_i]$ (or $\mu_{\mathbf{X_i}}$).
   (c) What is the value of $var[X_1]$ and $var[X_2]$.
2. Determine $\mathbf{C}_Y$ empirically. Use the function `plotcov` to draw the ellipse associated with $\boldsymbol{\mu}_Y$ and $\mathbf{C}_Y$. When plotting, use `axis equal` to correctly plot circles.
3. Let linear operator $\mathbf{A}$ be defined as:

$$\mathbf{A} = \begin{pmatrix} \sqrt{2} & 0 \\ 0 & 1/\sqrt{3} \end{pmatrix}$$

   Apply this operator to the dataset generated by Y to form realizations $\mathbf{g}$
   (a) Calculate the covariance matrix $\mathbf{C}_G$ and $\boldsymbol{\mu}_G$.
   (b) Plot the resulting new dataset $\mathbf{g}$ together with the associated ellipsoid. Explain its shape.
   (c) Define a new linear operator $\mathbf{A}$:

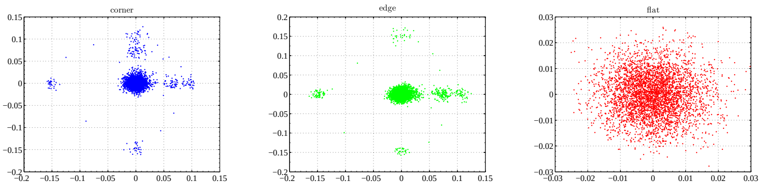
$$\mathbf{A} = \begin{pmatrix} \cos(\pi * 30/180) & -\sin(\pi * 30/180) \\ \sin(\pi * 30/180) & \cos(\pi * 30/180) \end{pmatrix}$$

   And apply $\mathbf{A}$ to the dataset $\mathbf{y}$ generated by Y to form realizations $\mathbf{e}$.
   (d) Calculate the covariance matrix $\mathbf{C}_E$ and $\boldsymbol{\mu}_E$.
   (e) Plot the resulting new dataset $\mathbf{e}$ together with the associated ellipsoid.
   (f) Determine the *eigenvalues* and *eigenvectors* of $\mathbf{C}_E$ and $\mathbf{C}_G$. Explain their values.

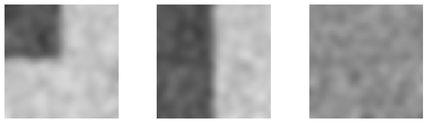
4 Feature detection

In this Section, we will use spatial filtering and eigenvalues to design a so called image feature detector. This particular algorithm can find edges and corners in images. This type of feature has many applications in image domain from identifying image content to automatically stitching panorama images together. See Figure 5.

Note: In this Section, the derivatives of some function f or a matrix $\mathbf{m}$ in x direction will de denoted as: $\frac{\partial f}{\partial x} = f_x$ and $\frac{\partial^2 f}{\partial x^2} = f_{xx}$



(a) Corner (b) Edge (c) Flat



(d) (e) (f)

Figure 3: $\mathbf{I}_x$ and $\mathbf{I}_y$ values for different image patches.

4.1 Feature Detection Algorithm

The basic Feature Detection Algorithm is as follows:

1. Compute the image derivatives in both x and y direction. This can be done by convolving the image with a first order Gaussian derivative. The Gaussian derivatives $G_\sigma^{'x}$ and $G_\sigma^{'y}$ may be estimated with the 3×3 matrix kernel $\mathbf{A}_x$ for both x and y direction:

$$\mathbf{A}_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \qquad \mathbf{A}_y = \mathbf{A}_x^T$$

$$\begin{aligned} \mathbf{I}_x &= G_\sigma^{'x} * \mathbf{I} \\ \mathbf{I}_y &= G_\sigma^{'y} * \mathbf{I} \end{aligned}$$

2. For all pixels, calculate the product of the derivatives:

$$\begin{aligned} \mathbf{I}_{xx} &= \mathbf{I}_x . * \mathbf{I}_x \\ \mathbf{I}_{yy} &= \mathbf{I}_y . * \mathbf{I}_y \\ \mathbf{I}_{xy} &= \mathbf{I}_x . * \mathbf{I}_y \end{aligned}$$

3. Convolve with a Gaussian filter. This will blur the intermediate results we have so far. Generate the Gaussian filter with a size $6 * \sigma$, meaning (+/- $3 * \sigma$). σ is thus a function

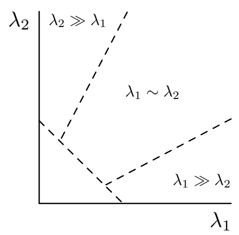


Figure 4: Eigenvalues from matrix $\mathbf{m}$. Corners are characterized by $\lambda_1 \sim \lambda_2$ where both are of significant value. Edges are indicated by $\lambda_1 \gg \lambda_2$ or $\lambda_2 \gg \lambda_1$ for large enough values.

parameter that must be set by the user. Ensure that σ has a minimum value of 1. You may use Matlab `fspecial` to generate the filter. See **Hints**. Then convolve as follows:

$$\begin{aligned} \mathbf{L}_{xx} &= G_{\sigma^w} * \mathbf{I}_{xx} \\ \mathbf{L}_{yy} &= G_{\sigma^w} * \mathbf{I}_{yy} \\ \mathbf{L}_{xy} &= G_{\sigma^w} * \mathbf{I}_{xy} \end{aligned}$$

4. Form matrix $\mathbf{m}$ at each pixel point (x, y):

$$\mathbf{m}(x,y) = \begin{bmatrix} L_{xx}(x,y) & L_{xy}(x,y) \\ L_{xy}(x,y) & L_{yy}(x,y) \end{bmatrix}$$

5. Matrix $\mathbf{m}$(x, y) has a number of interesting properties. The *eigen values* λ_1 and λ_2 of Matrix $\mathbf{m}$(x, y) for each point (x, y) give an indication if that particular point lies on an edge or on a corner. See Figure 4. A point on an edge is characterized by the fact that one of the eigenvalues is significantly larger than the other: $\lambda_1 \gg \lambda_2$ or $\lambda_2 \gg \lambda_1$. A corner is indicated by sufficiently large and similar eigenvalues. The 'corners-ness' measure $\mathbf{r}$ is defined as follows:

$$\mathbf{r} = \det \mathbf{m} - k(\text{trace } \mathbf{m})^2 \quad (2)$$

$$\det \mathbf{m} = \lambda_1 \lambda_2 \quad (3)$$

$$\text{trace } \mathbf{m} = \lambda_1 + \lambda_2 \quad (4)$$

To prevent the explicit calculation of eigenvalues, which is computationally intensive, one can determine the corner response $\mathbf{r}$ for all image points (x, y) as follows:

$$r(x,y) = \frac{L_{xx}(x,y) \cdot L_{yy}(x,y) - (L_{xy}(x,y))^2}{L_{xx}(x,y) + L_{yy}(x,y)} \quad (5)$$

An example of the corner reponse $\mathbf{r}$ can be seen in Figures 5b and 5c.

6. Optionally, you can implement a measure that stipulates that a potential corner $r(x, y)$ is only accepted if it is significantly larger than all other 'corner-ness' scores in a certain region of interest.

7. Find points for which $r(x, y)$ is the most strong, and write out their coordinates.

Exercise 8.

(a) Implement the feature detector and run it on 2-3 different images.

(b) Plot the found corners with a marker over the input images.

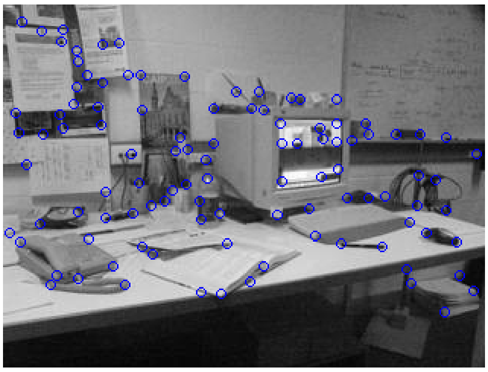
Exercise 9. The corner response **r** is rotation invariant, which naturally, is a direct consequence of the fact that it is based on eigenvalues. The feature points are, however, not invariant to scale. Large corners can also be classified as two big edges pending the scale the detector is working in. This scale can be set by function parameter σ of the Gaussian filter that was used in step x to blur the derivatives.

(a) Experiment with parameter σ

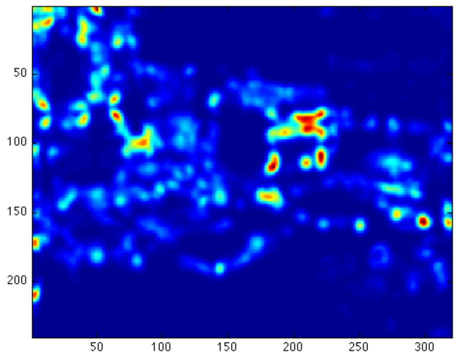
(b) Run the detector on the *tp2_003.jpg* images from `Chamilo` with different values of σ. Show the results.

Hints

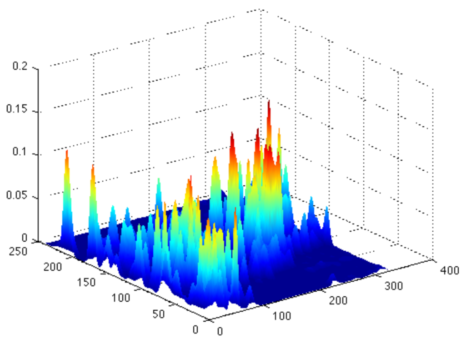
- You may generate the Gaussian filter of step 3 as follows:
  `g = fspecial('gaussian',max(1,fix(6*sigma)), sigma);`
- Do not define a separate matrix **m** for all pixels for $\mathbf{L}_{xx}$, $\mathbf{L}_{xy}$ and $\mathbf{L}_{yy}$. Access all separate **L**-components when needed.
- To find the local maxima in the corner response matrix or image **r**, you can use Matlab `ordfilt2`.



(a) Detected corners



(b) Corner response **r**



(c) Corner response **r**

Figure 5: Detecting corners and Edges

Submission

Please archive your report and codes in "Name_Surname.zip" (replace "Name" and "Surname" with your real name), and upload to "`Assignments/TP2: Spatial Filtering`" on `https://chamilo.unige.ch` before Wednesday, March 27 2019, 23:59 PM. Note, **the assessment is mainly based on your report, which should include your answers to all questions and the experimental results**.