Benjamin Scialom
A20430263

Computer Vision
CS-512
Homework 2

I. Noise and Filtering

a.

The signal to noise ratio is given my the following formula :

$$SNR = \frac{E_s}{E_n} = \frac{\sigma_s^2}{\sigma_n^2} = \frac{\frac{1}{n}\sum_{i,j}(I(i,j) - \hat{I})^2}{\sigma_n^2}$$

σ_n^2 is the variance of multiple frames of a static scene. It could be also seen as the variance in a uniform image region.

The signal to noise ratio can be given in decibel by using the logarithm function :

$$SNR(dB) = 10log_{10}(\frac{E_s}{E_n})$$

E_s and E_n stand for, respectively, the energy of the signal and the noise.

b.

Because of imperfections in the imaging and capturing process, the recorded image invariably represents a degraded version of the original scene. There exists a wide range of different degradations. A very important example is the existence of noise.

We focus on two type of noise :

- Gaussian noise is statistical noise having a probability density function (PDF) equal to that of the normal distribution, which is also known as the Gaussian distribution. In other words, the values that the noise can take on are Gaussian-distributed.
- Impulse noise is a category of noise which includes unwanted, almost instantaneous (thus impulse-like) sharp. ... A classic filter used to remove impulse noise is the **median filter**, at the expense of signal degradation.

c.

If we don't take into account the borders the value of the pixels in the image, which has the value 2 in each cells, after applying a 3x3 convolution filter having all 1-s in its entries will be 6.

We can compute the result thanks to this method : I = image, K = filter and R = result matrix.

$$R(x,y) = \begin{matrix} I(x-1,y-1)K(0,0) + I(x,y-1)K(1,0) + I(x+1,y-1)K(2,0) \\ +I(x-1,y)K(0,1) + I(x,y)K(1,1) + I(x+1,y)K(2,1) \\ +I(x-1,y+1)K(0,2) + I(x,y+1)K(1,2) + I(x+1,y+1)K(2,2) \end{matrix} = 18$$

d.

The derivative of an image convolved with a filter can be computed more efficiently than convolve with a 2D Gaussian.
Instead of convoluting with a 2D Gaussian, it should be convolved with **1D Gaussian along rows and then columns**. It works for all 2D filters.

Convolution by a 2D Gaussian

Convolution by 1D Gaussian along rows and columns

e.

Boundaries during convolution has to be dealt with consciousness since it could be complex. There are several methods to handle it easily :

- **zero-padding :** It consist in adding zero
- **minor replicate :** It consist in replicating rows and columns
- **ignore :** It consist in truncate row and line

f.

A basic smoothing filter could be : $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

The sum of all entries in the filter is 1.

Smoothing is often used to reduce noise within an image or to produce a less pixelated image. Most smoothing methods are based on low- pass filters. Smoothing is also usually based on a single value representing the image, such as the average value of the image or the middle (median) value.

g.

According to me, this question is the same one as the d).This was confirm by the professor.

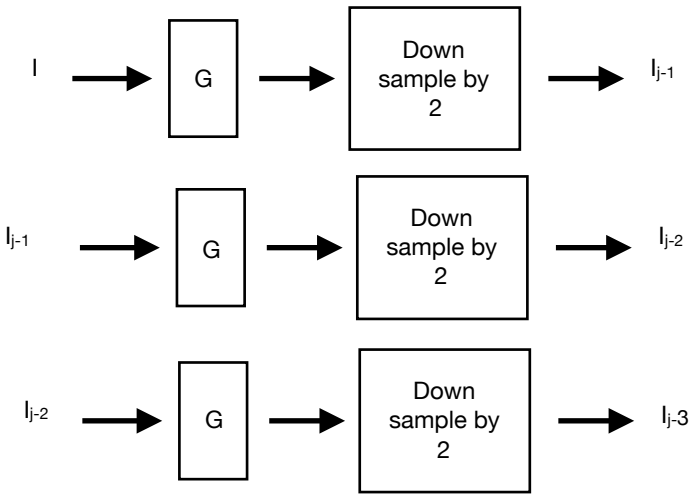
h.

We consider that we can see the whole Gaussian function when more the 99% of it is represented on the graph.
So, if we consider *m* as the size of the filter, we would need **at least** : $m = 3\sigma = 6$.
So, the matrix filter in 1D has a size mxm (1x6).

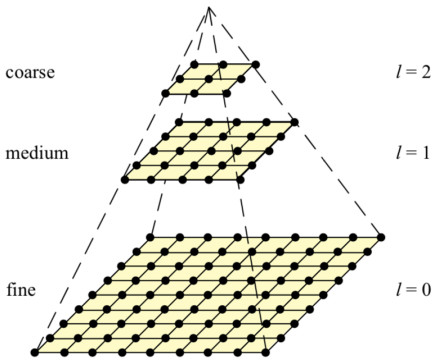
i.

The hardest thing to do when we use filters, which are sliding windows, is to decide the size of the window.
To solve this issue, one possibility is to change the size of the image. So, we study on several levels of the image.

Gaussian pyramid algorithm structure :



Example for 3 levels



A traditional image pyramid: each level has half the resolution

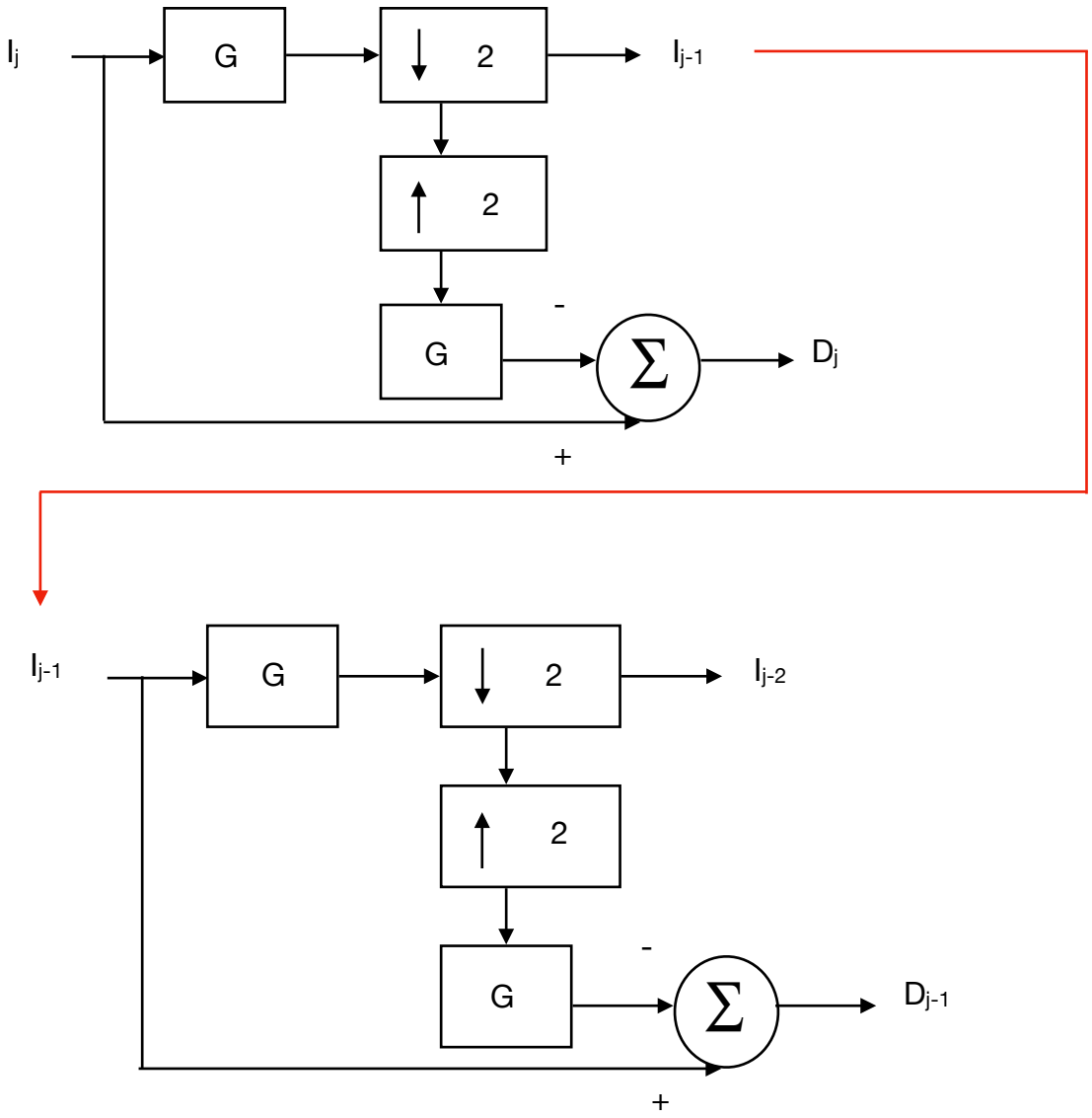
By doing that we add additional processing but it converges to a specific value which depends on the size of the filter *m* :

$$m^2 + \frac{1}{4}m^2(level\ 1) + \frac{1}{16}m^2(level\ 2) + \ldots < \frac{4}{3}m^2$$

j.

The Laplacian pyramid is used for **compression** since it has nice mathematical properties.

<u>Laplacian pyramid algorithm structure :</u>



II. Edge detection

a.

Qualitatively, **edges occur at boundaries between regions of different color, intensity, or texture**.
A reasonable approach is to **define an edge as a location of rapid intensity variation**.
On such a surface, edges occur at locations of steep slopes, or equivalently, in regions of closely packed contour lines.
A mathematical way to define the slope and direction of a surface is through its gradient. The local gradient vector J points in the direction of steepest ascent in the intensity function. Its **magnitude is an indication of the slope or strength of the variation**, while its orientation **points in a direction perpendicular to the local contour**.

b.

Globally, the basics steps of edge detection are **detect features** in all the images under consideration and then **match features** based on their local appearance (localization). It exists several enhancements for these two steps.

In details the following steps are :

Smoothing : (step 1)

Since gradient computation based on intensity values of only two points are susceptible to noise and other vagaries in discrete computations, filtering is commonly used to improve the performance of an edge detector with respect to noise. However, there is a trade-off between edge strength and noise reduction. More filtering to reduce noise results in a loss of edge strength.

Enhancement : (step 2)

In order to facilitate the detection of edges, it is essential to determine changes in intensity in the neighborhood of a point. Enhancement emphasizes pixels where there is a significant change in local intensity values and is usually performed by computing the gradient magnitude.

Detection : (step 3)

We only want points with strong edge content. However, many points in an image have a nonzero value for the gradient, and not all of these points are edges for a particular application. Therefore, some method should be used to determine which points are edge points. Frequently, thresholding provides the criterion used for detection.

Localization : (step 4)

The location of the edge can be estimated with sub-pixel resolution if required for the application. The edge orientation can also be estimated.

c.

There are two basic methods to compute the image gradient : $\nabla I(x,y) = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix}$

- Forward differences :

$$\frac{\partial I(x,y)}{\partial x} = \frac{I(x+h,y) - I(x,y)}{h} = I(x+1,y) - I(x,y)$$

$$\frac{\partial I(x,y)}{\partial y} = \frac{I(x,y+h) - I(x,y)}{h} = I(x,y+1) - I(x,y)$$

so, we have :

$$\Delta_x = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \ \& \ \Delta_x = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

- Central differences :

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

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

So, we have :

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

We have the value at one precise location. Hence, h =1.

The gradient of an image measures how it is changing. It provides two pieces of information. The magnitude of the gradient tells us how quickly the image is changing, while the direction of the gradient tells us the direction in which the image is changing most rapidly.

$$magnitude = |\nabla| = \sqrt{I_x^2 + I_y^2} \ \ \& \ \ angle = \arctan(\frac{I_y}{I_x})$$

d.

A sober filter can be produced by doing the convolution between the smoothing matrix and the derivative matrix.
Sobel's filter derivatives, $\Delta_{x,S}$ and $\Delta_{y,S}$, respectively detect horizontal edges and vertical edges.

$$\Delta_{x,S} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\Delta_{y,S} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ -2 & 0 & 2 \\ -1 & -2 & -1 \end{bmatrix}$$

e.

It is possible to generate more accurate derivatives :

- x-derivative :

$I_x = I * G_x^{'} * G_y$ -> image * derivative * smooth

- y-derivative :

$I_y = I * G_y^{'} * G_x$ -> image * derivative * smooth

where, $G(x) = e^{\frac{-x^2}{2\sigma^2}}$ and $G(y) = e^{\frac{-y^2}{2\sigma^2}}$.

If we take $\sigma = 2$ then we have to take a filter with a size $m = 5$.
Hence, we have the following derivatives along side x and y :

$$x = [-4 \quad -3 \quad -2 \quad -1 \quad 0 \quad 1 \quad 2 \quad 3 \quad 4]$$
$$G_x = \begin{bmatrix} G(-4) & G(-3) & G(-2) & G(-1) & G(0) & G(1) & G(2) & G(3) & G(4) \end{bmatrix}$$
$$y = [-4 \quad -3 \quad -2 \quad -1 \quad 0 \quad 1 \quad 2 \quad 3 \quad 4] \Rightarrow$$
$$G_y = \begin{bmatrix} G(-4) & G(-3) & G(-2) & G(-1) & G(0) & G(1) & G(2) & G(3) & G(4) \end{bmatrix}$$

For the derivative, $G_x^{'} = \frac{1}{\sigma^2} e^{\frac{-x^2}{2\sigma^2}}$ and $G_y^{'} = \frac{1}{\sigma^2} e^{\frac{-y^2}{2\sigma^2}}$. So, it is to get the array which correspond then.

f.

An edge can be localized thanks to the first and the second derivatives.

If we are using the **first derivative**, we should define **threshold** to detect a change (positive to negative slope or reverse) in the curve of the derivative function.Therefore, the **localization is not very precise**.
If we are using the **second derivative,** the edge is very well localized since we can have his location by looking at the **zero crossing** of the function.

g.

Laplacian of Gaussian consists in smooth with a gaussian before applying Laplacian :

$$H(x,y) = \nabla^2(I(x,y) * G) = \nabla^2 G * I(x,y)$$

Where ,

$$\nabla^2 G(x,y) = (\frac{x^2+y^2-2\sigma^2}{\sigma^4}) . \exp(-\frac{(x^2+y^2)}{2\sigma^2})$$

Edge detection using LOG :

1. Compute LOG : *H(x, y)*
2. Threshold : if H(i, j) < 0 then E(i, j) =0 & if H(i, j) ≥ 0 then E(i, j)=1
3. Mark edges at transmission : *0->1 & 1->0 (scan left-to-right and top-to-bottom)*

h.

Canny edge detection detects a**t zero crossing of second order directional derivative taken along the gradient** which differs from a standard edge detection without directional derivatives.

$$n = gradient = \nabla(I * G)$$

The directional derivative is given by : $\frac{\partial(I*G)}{\partial n} = n . \nabla(I*G) = |\nabla(I*G)|^2$

if $|n|$ is strictly superior to the threshold, then it detect edges at zero crossing of $\frac{\partial^2(I*G)}{\partial^2 n}$. The following step is to apply the non-maximum suppression.

i.

Non-maximum suppression :

We need local maximum of gradient magnitude in direction of the gradient. So, non-maximum suppression thins the ridges of gradient magnitude by suppressing all values along the line of the gradient that are not peak values of a ridge. That is done by comparing neighbor after discretization.

Let note : $\nabla(I * G) = (I_x, I_y)$

If $\nabla(I * G)$ is local maximum then *E(i, j)* is equal to 1 and 0 otherwise.

Hysteresis thresholding :

We use τ_H to start tracking and τ_c to continue.
The algorithm follows these steps :

1. Initialize array at visited pixels : *v(i, j)=0*
2. Scan image from top-to-bottom and left-to-right : if | v(i, j)| and $|\nabla I| > \tau_H$, we start tracking an edge
3. Search for additional neighbors in a direction orthogonal to ∇I