Filtrage linéaire

1 Principe

1.1 Introduction

L'objectif est de se familiariser avec le filtrage linéaire d'une image $I(x, y)$ par un filtre $F(x, y)$, conduisant à une nouvelle image $J(x, y)$.

Mathématiquement, le filtrage linéaire $2D$ est associé au produit de convolution suivant:

$$J(x, y) = I(x, y) * F(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I(x - u, y - v) \times F(u, v) du dv$$

que l'on discrétiser de la manière suivante:

$$J(x, y) = I(x, y) * F(x, y) = \sum_u \sum_v I(x - u, y - v) \times F(u, v)$$

En pratique, par symétrie des filtres, on considère le produit de corrélation:

$$J(x, y) = I(x, y) \otimes F(x, y) = \sum_u \sum_v I(x + u, y + v) \times F(u, v)$$

En pratique, le calcul en un point donné (i.e. en (x, y)) consiste à "superposer" le filtre à l'image sur ce point, puis à faire la somme des produits "terme à terme" entre le coefficients du filtre et les intensités correspondantes de l'image (i.e. au point, ou autour du point considéré).

English: The purpose is to get familiar with the linear filtering of an image $I(x, y)$ by a filter $F(x, y)$, leading to a new output image $J(x, y)$. Mathematically, $2D$ linear filtering is based an convolution:

$$J(x, y) = I(x, y) * F(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I(x - u, y - v) \times F(u, v) du dv$$

50	50	50	50	51	50	50
50	50	170	171	170	50	49
49	51	170	100	170	49	43
47	51	169	170	169	48	48
50	50	50	50	50	49	44
51	102	51	50	50	48	47
49	51	50	50	50	49	46

Table 1: Image

Table 2: Image

that can be discretize :

$$J(x,y) = I(x,y) * F(x,y) = \sum_u \sum_v I(x-u, y-v) \times F(u,v)$$

In practice, due to the symmetry of filters, on considers correlation:

$$J(x,y) = I(x,y) \otimes F(x,y) = \sum_u \sum_v I(x+u, y+v) \times F(u,v)$$

In practice, the computation of the result at one pixel (i.e. in (x,y)) is based on the 'superimposition' of the filter on the image at this point, and then on the computation of the sum of the pointwise product between filter coefficient and image points.

1.2 Calcul manuel en un point - Manual calculation

Soit l'image I des tableaux 2 et 1, et les filtres des tableaux 3 et 4.

English:

Let us consider image I (tables 2 and 1), and filters associated to tables (3 and 4).

En vous inspirant de l'exemple donné dans lors du cours, calculer (manuellement) les valeurs:

$$J_G(4,1) = (I \otimes F_G)(4,1)$$
$$J_G(4,0) = (I \otimes F_G)(4,0)$$
$$J_S(4,1) = (I \otimes F_S)(4,1)$$

1	2	1
2	4	2
1	2	1

Table 3: Filtre gaussien/Gaussian filter F_G

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

Table 4: Filtre de Sobel/Sobel filter F_S

$$J_S(4,0) = (I \otimes F_S)(4,0)$$

Remarque: on considère que la première coordonnée correspond à l'indice de la ligne, commençant à 0.

Remarque: le calcul en bordure d'image (e.g. indice x ou y à 0) nécessite d'élargir (i.e. ajout de lignes et colonnes) celle-ci le temps du calcul et d'effectuer ensuite les opérations sur la région interne (i.e. sans la bordure). En anglais et dans ce contexte, ceci correspond à l'opération de *padding*. Penser à procéder ainsi pour les calculs en $(4,0)$ (en chaque point de la bordure, on affectera la valeur du pixel interne le plus proche).

English:
Let us consider image I (tables 2 and 1), and filters associated to tables (3 and 4). Manual calculate :

$$J_G(4,1) = (I \otimes F_G)(4,1)$$
$$J_G(4,0) = (I \otimes F_G)(4,0)$$
$$J_S(4,1) = (I \otimes F_S)(4,1)$$
$$J_S(4,0) = (I \otimes F_S)(4,0)$$

Note : One considère that the first coordinate corresponds to rows, starting at 0.

Note: Computations at image boundaries (e.g. x=0 or y=0) require padding (adding news rows/columns), while calculations remains constrained within to inner part of image (e.g. without padding). In this exercice, pixels of the newly added rows/columns will have the same value as the closest pixel belonging to the original image (without padding).

1.3 Implémentation numérique

En utilisant la fonction `scipy.ndimage.correlate()`, filtrez l'ensemble de l'image:

$$J_G = (I \otimes F_G)$$

$$J_S = (I \otimes F_S)$$

Afficher les images filtrées (`print`) et vérifier la conformité avec le calcul manuel aux points $(4, 1)$ et $(4, 0)$.

En pratique, un filtre du type de F_G est normalisé (par la somme de ses coefficients (`filtre=filtre.astype(np.float)/np.sum(filtre)`)): calculer $J_G = I \otimes F_G$ dans ce cas.

English:
Using the function `scipy.ndimage.correlate()`, filter images :

$$J_G = (I \otimes F_G)$$

$$J_S = (I \otimes F_S)$$

Print filtered images (`print`) and check the conformity of results with manual calculations at pixels $(4, 1)$ and $(4, 0)$.

In practice, a filter of type F_G is normalized by the sum of its coefficients (`filtre=filtre.astype(np.float)/np.sum(filtre)`): compute $J_G = I \otimes F_G$ is this case.

2 Application à la segmentation: image synthétique

On souhaite, par seuillage, extraire de l'image synthétique 2 le carré.

- Appliquer un simple seuillage à l'image initiale, le seuil étant choisi en observant les valeurs des pixels: on constate que, quelques soit le seuil choisi, on ne parvient pas à extraire le carré parfaitement.
- Appliquer préalablement un filtrage avec le filtre F_G **normalisé**. On normalise le filtre en divisant les coefficients par leur somme.

English:
On wants, by thresholding, to extract the square object from the synthetic image2.

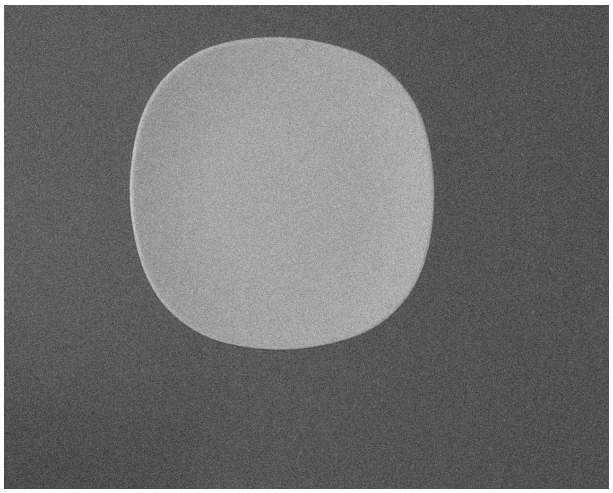

Figure 1: Image

- Apply a simple thresholding to the initial image, the threshold being chosen by observing pixels values : one sees that, whatever the chosen threshold, one cannot correctly extract the square.
- Filter the image first with the F_G filter (**normalized**), and then threshold the image : one observes that the result is improved.

3 Application à la segmentation: image réelle et approche par région (region-based segmentation)

En utilisant l'image fournie, on souhaite mesurer la taille de l'assiette (voir image 1), qui fait 353394 pixels.
Implémentez les traitements suivants:

- Appliquer un seuillage directement à l'image initiale (utiliser otsu pour calculer le seuil optimal): afficher l'histogramme (en superposant le seuil) et l'image finale. On constate que, à cause du bruit, l'image seuillée est incorrecte, et la surface mal estimée.

- Reprendre la procédure précédente mais en appliquant au préalable un filtrage. On considérera un filtre moyenneur: matrice (normalisée) de taille 3x3 et composée de 1. Afficher également l'histogramme et constater que les 2 lobes sont mieux séparés.

English:
Using the provided image, on wants to measure the area of plate (see Image 1), which is 353394 pixels.
Test the following procedures :

- Directly apply a thresholding to the initial image (Ostu method for determining the optimal threshold): display the histogram (superimposing the threshold value) and the resulting image. One observes that, due to the noise, the result is altered and the area not correctly estimated.

- Apply the same procedure, but with a preliminary filtering (normalized mean filter of size 3x3 pixels) : the result is better (also display the histogram : both lobes are better separated).

4 Application à la segmentation: image réelle et approche par contour (contour-based segmentation)

Considérons maintenant l'image de la figure 5. En appliquant l'approche précédente, on observe que le résultat n'est pas correct (voir 6). Pour contourner ce problème, on considérera la procédure suivante:

1. Réduction du bruit par filtrage "moyenneur" de grande taille (e.g. filtre de taille 13x13),

2. Extraction des contours (approche par contour - voir support de cours): deux filtrages linéaires avec des filtres de Sobel (vertical et horizontal), et combinaison des résultats (racine carrée de la somme des carrés des images « contours »). On pourra soit définir soit-même les 2 filtres et utiliser `scipy.ndimage.correlate`, soit utiliser la fonction `ndimage.filters.sobel` qui prédéfinit ces filtres.

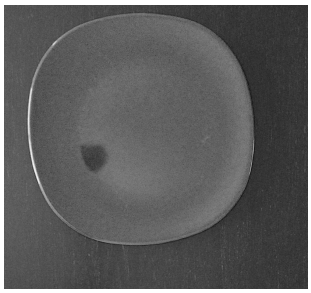

Table 5: Image

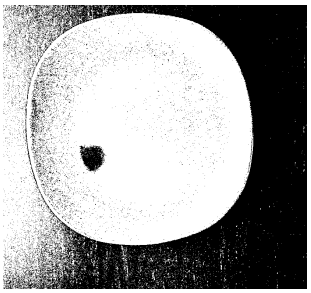

Table 6: Image seuillée / thresholded image.

3. Seuillage du résultat (seuil obtenu par la méthode d'otsu),

4. Remplissage de la zone délimitée par le contour de l'objet (avec une 4-connexité et non un 8-connexité), avec la fonction `scipy.ndimage.morphology.binary_fill_holes`

English:
Let us now consider the image reported by Figure 5. By considering the previous region-based approach, the result is not correct (see Figure 6). To overcome this issue, let us consider the following approach :

1. Remove noise by mean filtering (large filter of size 13x13 pixels),

2. Extract contours using 2 Sobel filterings (vertical and horizontal) and combined both filtered images (square root of the sum of squared filtered images). One can either manually define both filters or use the function `ndimage.filters.sobel`.

3. Threshold the result (threshold determined by Otsu),

4. Fill the area surrounded by the contour using the function `scipy.ndimage.morphology.binary_fill_holes` (with a 4-connexity and not a 8-connexity)