# TPs de Vision

***Elio Genson***

Table des matières

[TPs de Vision 1](#_Toc6410)

[TP1 - Caractérisation fréquentielle de la texture 2](#_Toc8300)

[I.Filtrage fréquentiel 2](#_Toc16947)

[II.Caractérisation 5](#_Toc17165)

[TP2 - Points d'intérêt 7](#_Toc9248)

[I. SIFTs 7](#_Toc21365)

[II. Points de Moravec : 8](#_Toc5500)

[III. LBP 10](#_Toc18503)

[TP3 - Transformée de Hough 11](#_Toc19062)

[I. Modélisation de segments de droite. 11](#_Toc9380)

[II. Reconnaissance de segments de droite 12](#_Toc8854)

[III. Rétroprojection de la transformée de Radon 15](#_Toc14140)

[TP4 - Transformée en distance. 15](#_Toc29193)

[I. Modélisation 15](#_Toc10966)

[II.Reconnaissance 16](#_Toc2687)

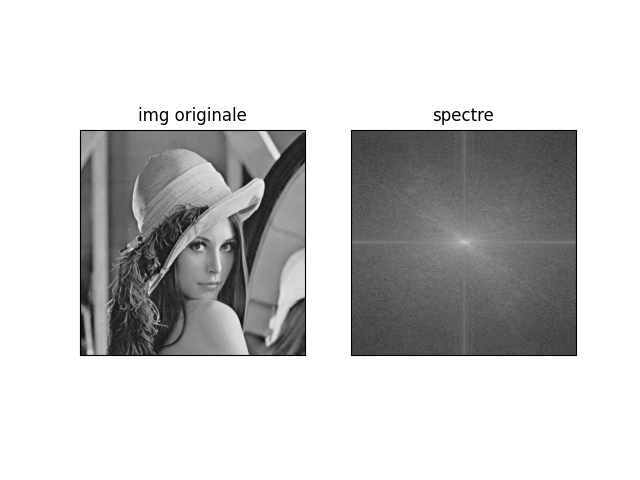
[TP5 - Forêt aléatoire 17](#_Toc15432)

[TP6 - Cascades de Haar et Yolo 18](#_Toc10193)

# TP1 - **Caractérisation fréquentielle de la texture**

### I.Filtrage fréquentiel

1. Image du spectre de Fourier :

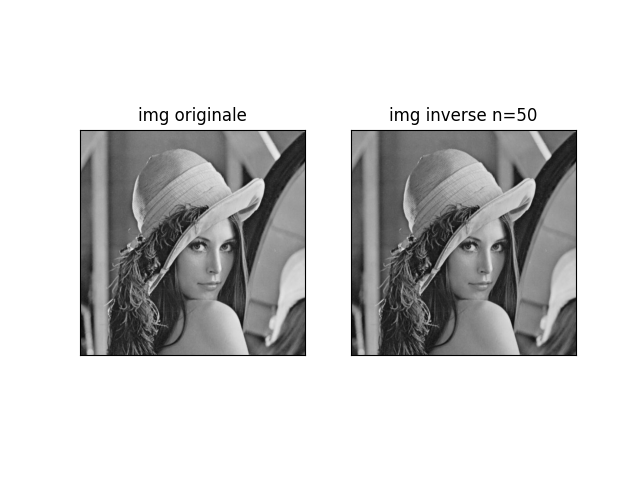


1. Image reconstruite :



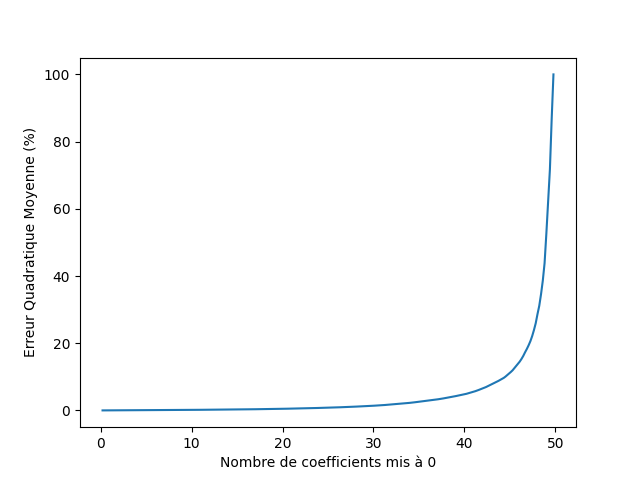
Le MSE calculé entre les deux images est de 17553

1. Images avec coefficient à 0 :



On remarque que l’image n=200 est bien détérioré, il y a des «vagues» sur les contours des formes.

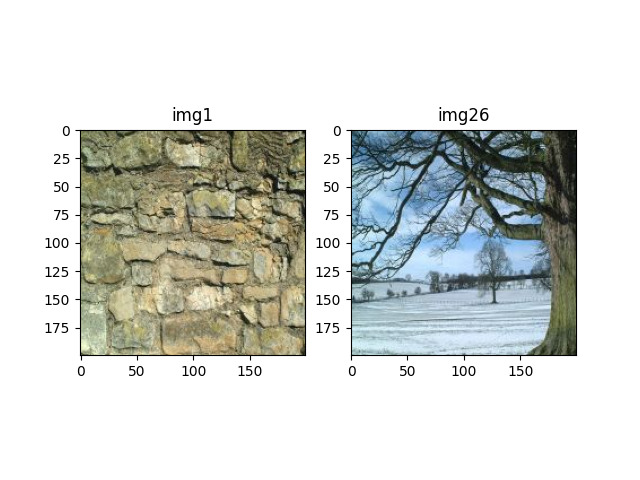
1. Graphe de la MSE en fonction du nombre de coef mit à zero :



### II.Caractérisation

Pour l’image 1 :

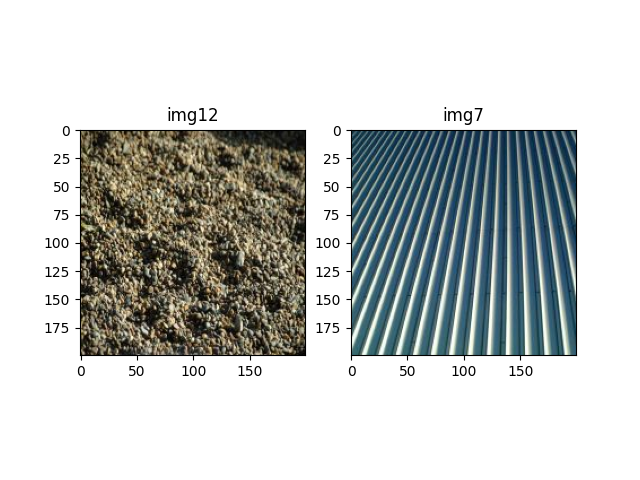
Distance de manhantan min = 249



Autres testes :

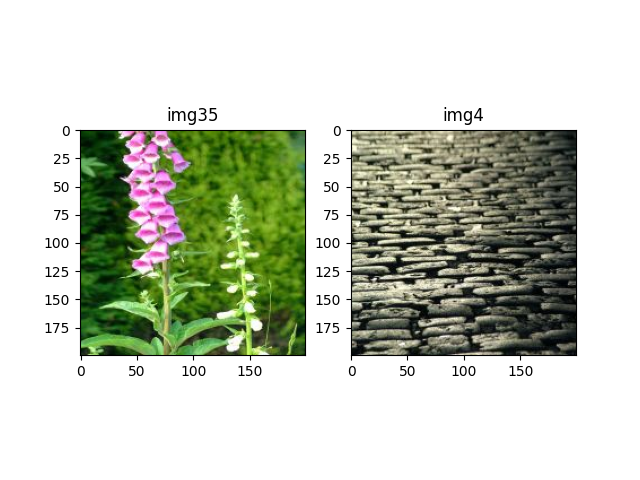
Img 12

Distance min = 35913



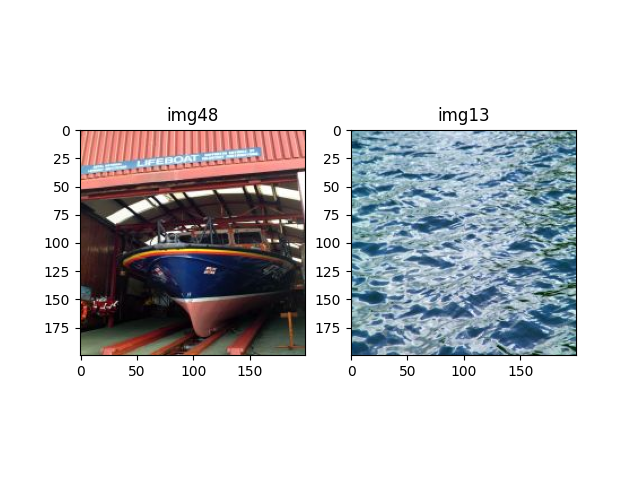
Img 35

Distance min = 9658



Img 48

Distance min = 8385



Je ne retrouve pas beaucoup de corrélation au niveaux des textures des images testés.

les seuls images qui on un rapport son la 35 et la 4 ou les feuilles on le même style de texture que les pavets.

# TP2 - **Points d'intérêt**

### SIFTs





Description du code :

cv.SIFT\_create() crée un détecteur SIFT.

detectAndCompute(img, None) détecte les points d’intérêt (keypoints) et calcule leurs descripteurs.

k\_1, k\_2 : listes des points clés détectés.

des\_1, des\_2 : matrices contenant les descripteurs SIFT des points clés.

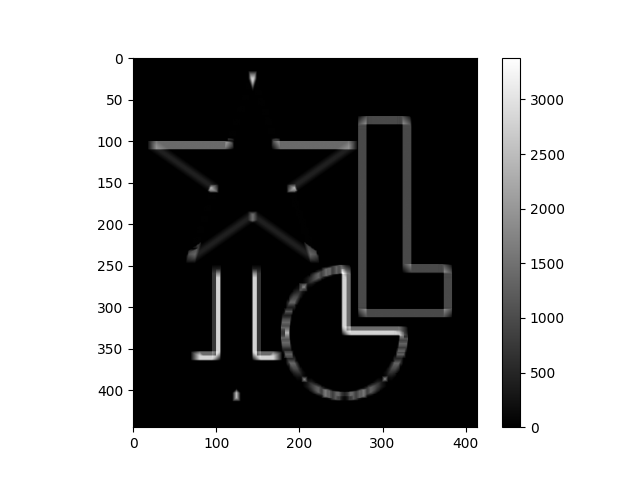
cv.BFMatcher (Brute-Force Matcher) est un algorithme d'appariement descripteur par descripteur.

bf.match(des\_1, des\_2) apparie les descripteurs des deux images.

sorted(matches, key=lambda x: x.distance) trie les correspondances par distance croissante (les meilleures correspondances en premier).

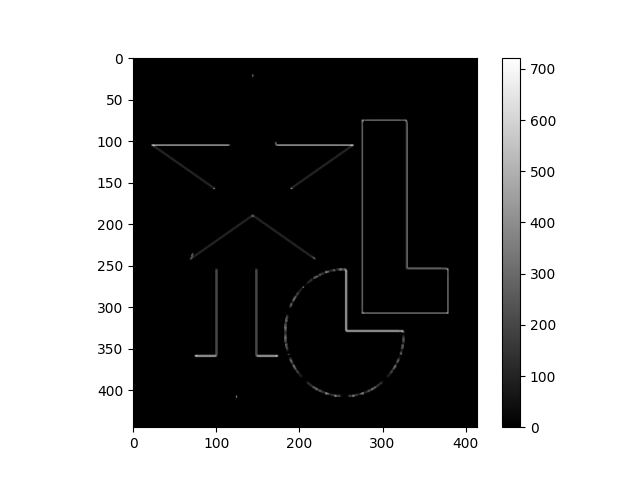
## Points de Moravec :

Fenetre de taille 11



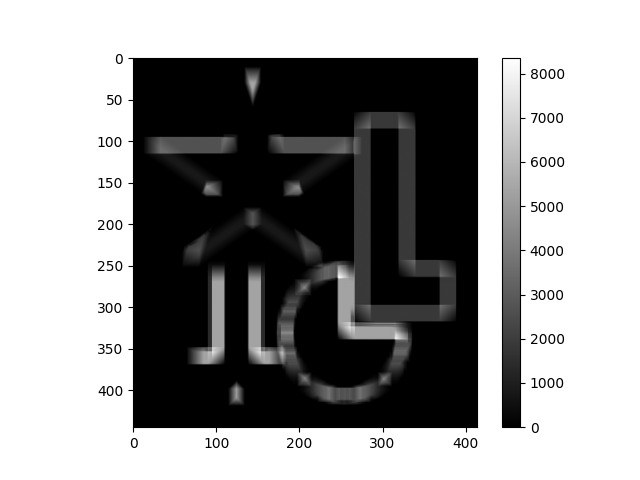
43 points d’intéret

fenêtre de taille 3

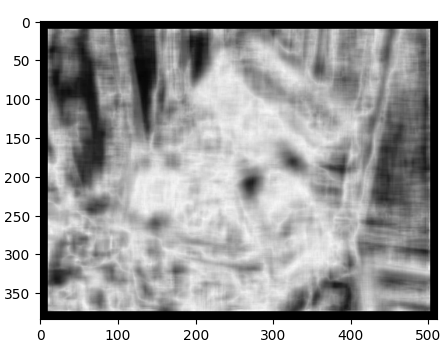


233 points d’intéret

fenêtre de taille 21

233 points d’intéret

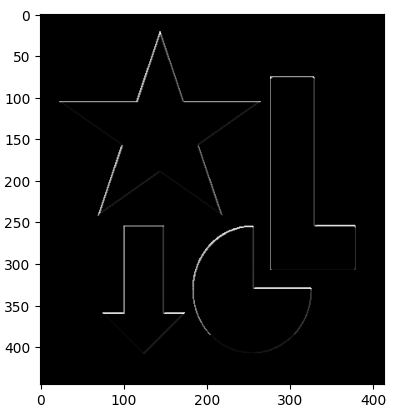
sur l’image dela partie 1 en 11 pixels :



8603 points de moravec

plus le nombre de pixel de voisinage est grand, plus nombreux sont les point de moravec détecté

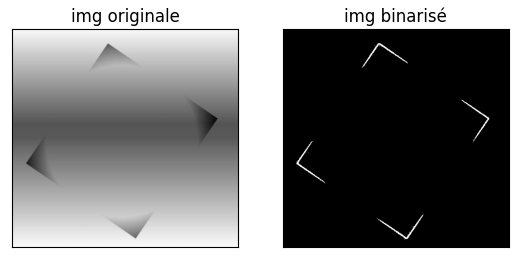
## LBP



11 point d’intéret

# TP3 **- Transformée de Hough**

## I. Modélisation de segments de droite.



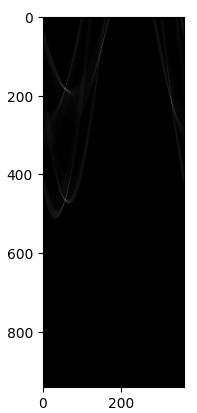
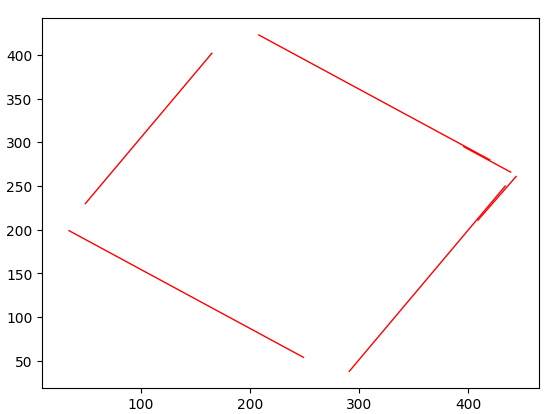


image de Vote :

## II. Reconnaissance de segments de droite



k=10

Point 10 : 467 56 ( 208 , 423 ) ( 420 , 280 )

Point 9 : 183 56 ( 265 , 42 ) ( 265 , 42 )

Point 8 : 93 145 ( 30 , 205 ) ( 30 , 205 )

Point 7 : 184 56 ( 34 , 199 ) ( 249 , 54 )

Point 6 : 219 326 ( 308 , 65 ) ( 308 , 65 )

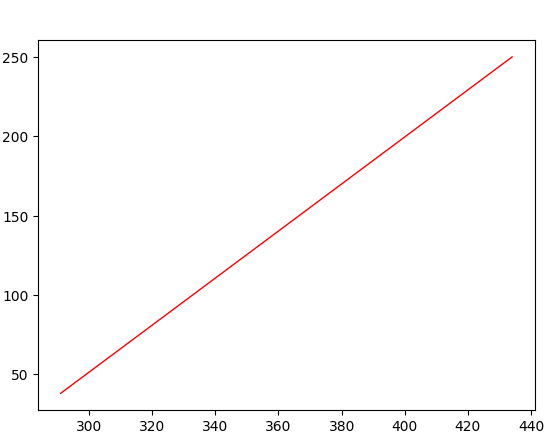
Point 5 : 466 56 ( 396 , 295 ) ( 439 , 266 )

Point 4 : 214 325 ( 409 , 211 ) ( 444 , 261 )

Point 3 : 88 146 ( 49 , 230 ) ( 165 , 402 )

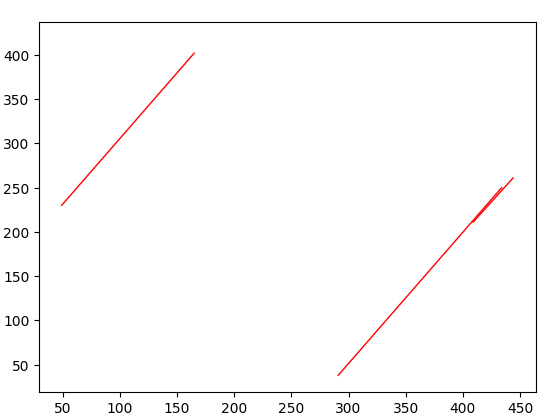
Point 2 : 87 146 ( 177 , 418 ) ( 177 , 418 )

Point 1 : 220 326 ( 291 , 38 ) ( 434 , 250 )



K=1

Point 1 : 220 326 ( 291 , 38 ) ( 434 , 250 )



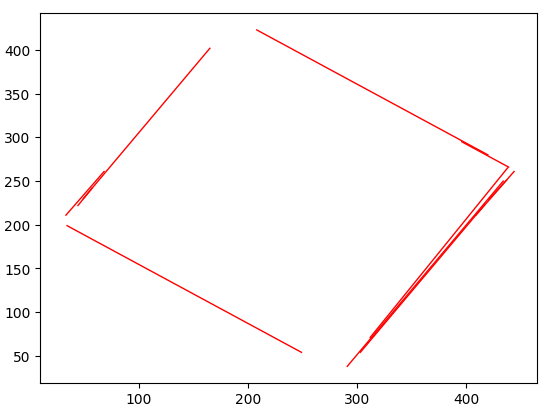
K=4

Point 4 : 214 325 ( 409 , 211 ) ( 444 , 261 )

Point 3 : 88 146 ( 49 , 230 ) ( 165 , 402 )

Point 2 : 87 146 ( 177 , 418 ) ( 177 , 418 )

Point 1 : 220 326 ( 291 , 38 ) ( 434 , 250 )



K= 20

Point 20 : 228 327 ( 442 , 262 ) ( 442 , 262 )

Point 19 : 206 324 ( 421 , 229 ) ( 437 , 251 )

Point 17 : 84 147 ( 44 , 222 ) ( 57 , 242 )

Point 16 : 223 327 ( 312 , 71 ) ( 438 , 265 )

Point 15 : 94 145 ( 33 , 211 ) ( 68 , 261 )

Point 13 : 215 325 ( 413 , 215 ) ( 413 , 215 )

Point 11 : 221 326 ( 303 , 54 ) ( 417 , 223 )

Point 10 : 467 56 ( 208 , 423 ) ( 420 , 280 )

Point 9 : 183 56 ( 265 , 42 ) ( 265 , 42 )

Point 8 : 93 145 ( 30 , 205 ) ( 30 , 205 )

Point 7 : 184 56 ( 34 , 199 ) ( 249 , 54 )

Point 6 : 219 326 ( 308 , 65 ) ( 308 , 65 )

Point 5 : 466 56 ( 396 , 295 ) ( 439 , 266 )

Point 4 : 214 325 ( 409 , 211 ) ( 444 , 261 )

Point 3 : 88 146 ( 49 , 230 ) ( 165 , 402 )

Point 2 : 87 146 ( 177 , 418 ) ( 177 , 418 )

Point 1 : 220 326 ( 291 , 38 ) ( 434 , 250 )

## III. Rétroprojection de la transformée de Radon

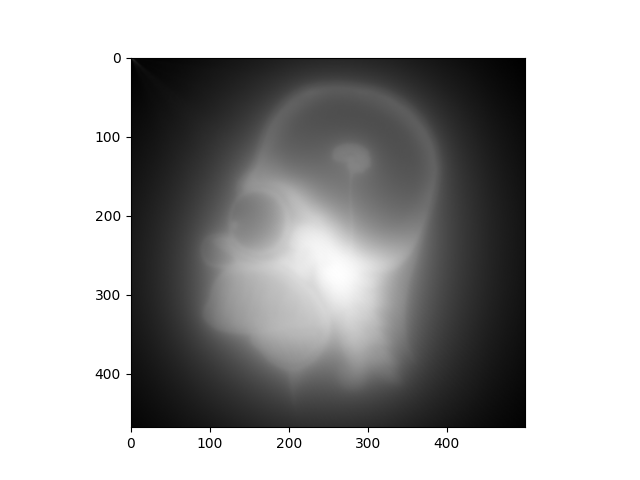
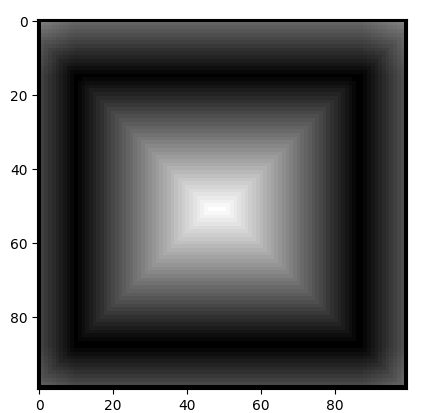
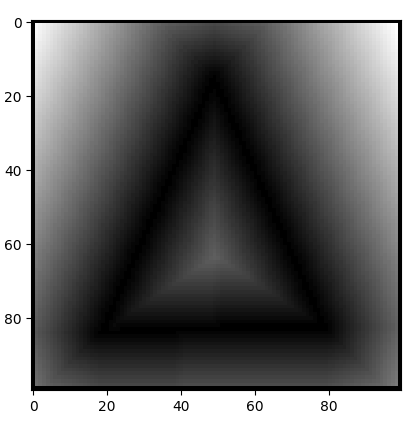


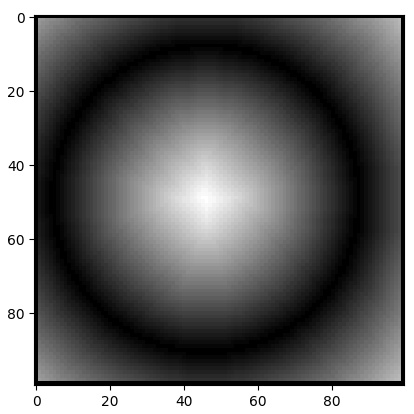
image de radon reconstruite

l’organe est un cerveau !

# TP4 - Transformée en distance.

## Modélisation



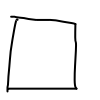
Transformées en distance calculées avec ce masque :

[[4. 3. 4.]

[3. 0. 3.]

[4. 3. 4.]]

## II.Reconnaissance



Reco en cercle



Reco en cercle

reco en Triangle

Une erreur avec le carré reco en cercle.

carre: {'triangle2': 66577497.0, 'carre2': 69139059.0, 'cercle2': 69553092.0}

triangle: {'cercle2': 101068827.0, 'carre2': 101641816.0, 'triangle2': 105029518.0}

cercle: {'triangle2': 76482353.0, 'carre2': 78693061.0, 'cercle2': 79128188.0}

# TP5 **- Forêt aléatoire**

Je calcule 3301 haar features sur les images avec un bloc 10x10 pixels

# TP6 **- Cascades de Haar et Yolo**

Teste sur corps entier :

Teste sur un visage :



Teste visage + yeux :



On remarque beaucoup d’erreur de détection sur un fond chargé.