Transformata Hadamard pentru reprezentarea imaginilor și filtrare

-PROIECT PRELUCRAREA NUMERICĂ A IMAGINILOR-

Abstract

Această lucrare își propune prezentarea utilității transformatei Hadamard în ceea ce privește prelucrarea de imagini și descrierea matematică a acesteia.

Totodată, documentul expune pe scurt logica de programare pe care am folosit-o pentru a scrie un program în Python care să urmărească algoritmul specific transformării precum și calculul anumitor parametri specifici acestui mod de prelucrare.

1.Cerințe

Se urmărește generarea matricei transformării și aplicarea formei matriceale pe imagine precum și filtrarea informației obținute în domeniul frecvență.

Date de intrare:

- o imagine pe nivele de gri;
- dimensiunea blocului;
- parametrii de filtare.

Date de ieșire:

- imaginea originală și histograma ei;
- imaginea în domeniul frecvență cu histograma corespunzătoare;
- imaginea refăcută și histograma ei;
- parametri privind eroarea la refacere, conservarea și compactarea energiei.

2.Aspecte teoretice

2.1. Introducere

Teoria transformărilor joacă un rol deosebit de important în procesarea de imagini. Transformările sunt de preferat imaginii originale întrucât este mult mai ușor de scos în evidență anumite proprietăți ale imaginii și folosirea lor în funcție de o anumită aplicație. Transformările 2D sunt des utilizat în algoritmii de îmbunătățire a imaginilor, refacerea și codarea acestora. [1]

Transformata Hadamard (cunoscută și sub numele de Transformata Walsh-Hadamard sau Transformata Hadamard-Rademarcher-Walsh) este un exemplu de generalizare a clasei de transformate Fourier. Este mai rapidă decât transformările sinusoidale deoarece nu necesită înmulțiri. Această transformată se aplică în compresia de imagini, filtrare și proiectarea de codoare. Alte utilizări sunt în criptarea datelor, în algoritmi de compresie de date și în procesare

de semnal (JPEG XR și MPEG-4 AVC), în aplicațiile video, și de asemenea, joacă un rol foarte important în ceea ce privește algoritmii cuantici si prezintă o bună compactare de energie.

2.1. Considerente generale

Transformata Hadamard are la bază o matrice rectangulară cunoscută sub denumirea de matrice Hadamard. Transformările rectangulare se caracterizează prin faptul că folosesc funcții de bază ce reprezintă variații ale unor unde dreptungiulare. În general, ele se caracterizează prin rapiditate de calcul, având avantajul utilizării a mai puține operații de înmulțire (față de cazul transformărilor în care variază unde sinusoidale, de exemplu).

O matrice Hadamard, $H = (h_{ij})$ este definită ca o matrice pătratică de dimensiune $N \times N$ în care:

a) Toate elementele sunt ± 1;
b) Oricare două coloane distincte sunt ortogonale: $\forall\, i, j,\ \ i \neq j,\ \ \sum_k h_{ik} h_{jk} = 0$.

Funcțiile de bază ale transformării Hadamard sunt, de fapt, funcții Walsh. Din această cauză ea este denumită uneori și transformare Walsh. În unele referințe se întâlnește și denumirea combinată de transformare Walsh-Hadamard.

Matricea Hadamard cu ordinul cel mai scăzut se obține pentru N=2:

$$H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}; \qquad (1)$$

Matricele de ordin superior ale transformării se definesc foarte ușor utilizând o relație recursivă simplă. Astfel, dacă H_N reprezintă matricea de ordin N, atunci relația recursivă are următoarea formă:

$$H_N = \frac{1}{\sqrt{N/2}} \begin{bmatrix} H_{N/2} & H_{N/2} \\ H_{N/2} & -H_{N/2} \end{bmatrix}; \qquad (2)$$

Expresia (2) reprezintă matricea transformării Hadamard în forma sa neordonată. În formă ordonată se obține matricea transformării $H_{N,ord}$ prin rearanjarea liniilor matricii H_N în ordine crescătoare a numărului de schimbări de semn pe linie, în mod analog cu creșterea frecvenței la transformata Fourier (rezultă o transformare mai ușor de implementat).

După acest principiu, H_8 are următoarea formă:

$$H_8 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{matrix} 0 \\ 7 \\ 3 \\ 4 \\ 1 \\ 6 \\ 2 \\ 5 \end{matrix} \qquad (3)$$

$$H_{8,ord} = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix} \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} \qquad (4)$$

O transformare unitară a imaginii U[MxN] reprezintă o rotație a spațiului MN-dimensional, fiind definită de o matrice unitară de rotație A(de dimensiune MNxMN). Se aplică relația:

$$V = A \times U \times A^T \qquad (5)$$

Particularizând, în cazul transformării Hadamard se obține:

$$V = H \times U \times H \qquad (6)$$

2.2. Conservarea energiei

Pentru transformări bidimensionale, așa cum este transformata exemplificată, conservarea energiei înseamnă verificarea respectării relației:

$$E_V = E_U; \qquad (7)$$

,unde:

$$E_U = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} u^2(m,n) \; ; \qquad (8)$$

$$E_V = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} |v(k,l)|^2 \; . \qquad (9)$$

E_U=Energia imaginii de intrare, E_V =Energia imaginii în domeniul transformat.

2.3. Compactarea energiei

Compactarea energiei reprezintă o proprietate importantă în ceea ce privește compresia de imagini. Ea se poate calcula atât pentru o imagine individuală cât și pentru un set de imagini.

Compactarea energiei blocului în domeniul transformat, în comparație cu distribuția energiei blocului în domeniul original, este dată de numărul de coeficienți nuli sau foarte mici în

comparație cu coeficienții din blocul original. Astfel, capacitatea de compactare se poate obține sub forma unui procent:

$$p = \frac{v(k,l)\neq 0}{nr.total\ de\ coeficienți} \qquad (10)$$

,unde numărul total de coeficienți este reprezentat de $M \times N$. Particularizând, în cazul transformării Hadamard avem:

$$p = \frac{v(k,l)\neq 0}{N^2} \qquad (11)$$

2.4. Etapa de filtrare

Prin transformarea directă se determină, de fapt, coeficienții de ponderare [2], iar în transformarea inversă se face reasamblarea imaginii din imaginile de bază.

Filtrarea în domeniul transformat presupune o modificare a coeficienților de ponderare (prin înmulțire sau prin alte metode), înaintea reconstrucției imaginii prin transformarea inversă. Pentru ușurința implementării, în etapa următoare am ales o mască de filtrare dreptunghiulară. Astfel, după transformarea inversă rezultă imaginea filtrată.

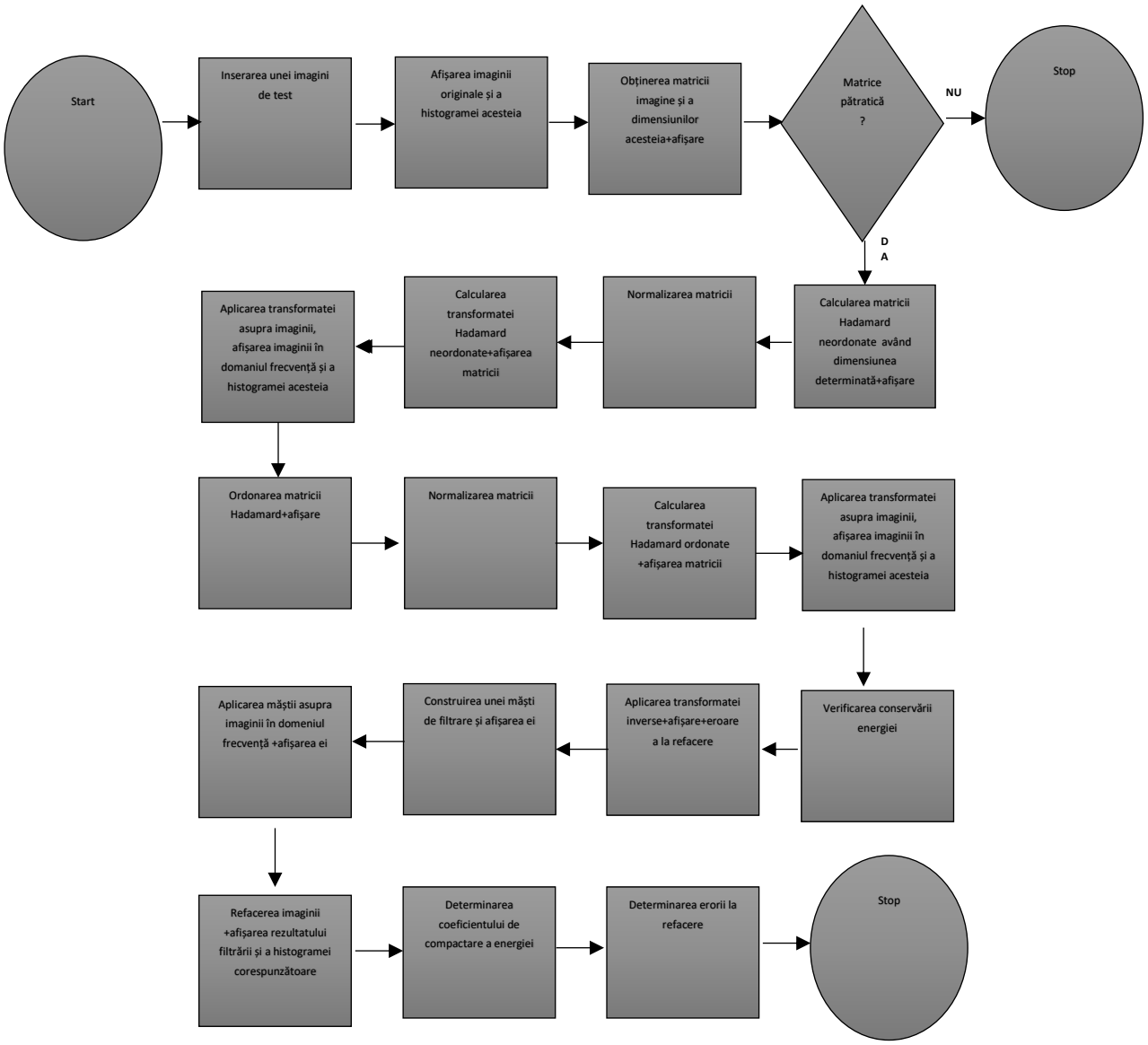
$$U' = (A^{-1})^T \times V' \times A^T \qquad (12)$$

Datorită faptului că transformata Hadamard bidimensională este o transformare unitară și separabilă, formula de calcul a transformării inverse devine identică cu cea a transformării directe.

$$U' = H \times V' \times H \qquad (13)$$

2.5. Soluția propusă

Ne-am gândit la un algoritm care să se execute în mod secvențial după următoarea structură:


Start
Inserarea unei imagini de test
Afișarea imaginii originale și a histogramei acesteia
Obținerea matricii imagine și a dimensiunilor acesteia+afișare
Matrice pătratică ?
NU
Stop
D
A
Calcularea matricii Hadamard neordonate având dimensiunea determinată+afișare
Normalizarea matricii
Calcularea transformatei Hadamard neordonate+afișarea matricii
Aplicarea transformatei asupra imaginii, afișarea imaginii în domaniul frecvență și a histogramei acesteia
Ordonarea matricii Hadamard+afișare
Normalizarea matricii
Calcularea transformatei Hadamard ordonate +afișarea matricii
Aplicarea transformatei asupra imaginii, afișarea imaginii în domaniul frecvență și a histogramei acesteia
Verificarea conservării energiei
Aplicarea transformatei inverse+afișare+eroare a la refacere
Construirea unei măști de filtrare și afișarea ei
Aplicarea măștii asupra imaginii în domeniul frecvență +afișarea ei
Refacerea imaginii +afișarea rezultatului filtrării și a histogramei corespunzătoare
Determinarea coeficientului de compactare a energiei
Determinarea erorii la refacere
Stop

3.Implementare

Pentru exemplificarea modului de aplicare a transformatei Hadamard asupra unei imagini am folosit utilitarul Spyder din suita Anaconda, produsul final fiind o arhivă conținând un fișier cu extensia .py și, respectiv, imagini de test.

Librăriile utilizate:

- numpy → este utilizată pentru lucrul cu array-uri și are funcții care se folosesc în domeniul algebrei liniare, transformatelor Fourier și în operațiile cu matrici;
- matplotlib.pyplot → este utilizată pentru a crea reprezentări grafice;
- math → este utilizată pentru funcții matematice.

Programul va fi modularizat, urmărind următoarele etape:

- Aducem imaginea de intrare pe nivele de gri și obținem matricea conținând valorile luminanțelor pixelilor;

```
"""Import imagine de test"""
Imagine_de_intrare="PeppersGri.bmp"
U=cv2.imread(Imagine_de_intrare,0).astype(float)
```

- Facem afișarea imaginii originale cu histograma corespunzătoare;

```
print("Imaginea originala: \n")
plt.imshow(U,cmap='gray')
plt.show()

print("\nHistograma imaginii originale: ")
plt.figure()
(n,bins,patches)=plt.hist(U.ravel(),256,[0,255],density=True,cumulative=True)
plt.show()
```

- Afișăm matricea imagine și, respectiv, dimensiunea imagini;

```
print("\nMatricea imagine: ")
print(U,"\n")
nr_linii=len(U)
print("\nNumarul de linii: ",nr_linii)
nr_coloane=len(U[0])
print("\nNumarul de coloane: ",nr_coloane)
```

- Apelăm o funcție prin care se generează matricea Hadamard de ordinul egal cu dimensiunea imaginii (dacă numărul de linii din matricea imagine este egal cu numărul de coloane);

APEL:

```
if nr_linii==nr_coloane:
    """Generarea matricii Hadamard"""
    N=nr_linii
    H=Hadamard(N)
```

DEFINIRE FUNCȚIE:

```
"""calculam in mod recursiv matricae Hadamard sub forma nenormalizata"""
def Hadamard(N):
    if N>2:
        Matrice1=np.concatenate((Hadamard(N/2),Hadamard(N/2)),axis=1,out=None)
        Matrice2=np.concatenate((Hadamard(N/2),-Hadamard(N/2)),axis=1,out=None)
        Matrice3=np.concatenate((Matrice1,Matrice2),axis=0,out=None)
        H=Matrice3
    else:
        aux2=1
        A=[[1,1],[1,-1]]
        H=np.dot(aux2,A)
    return H
```

- Normalizarea matricii:

```
"""Normalizare"""
var=1/math.sqrt(N)
H=np.dot(var,H)
```

- Se calculează transformata.

```
"""Transformata Hadamard
V=H*U*H
"""
aux=np.dot(H,U) #H*U
V=np.dot(aux,H) #(H*U)*H
print("\n\n\n")
```

- Facem afișarea imaginii neordonate în domeniul transformat cu histograma corespunzătoare;

```
print("\nMatricea transformarii in forma neordonata: ")
print(V)
print("\nImaginea transformata(in forma neordonata): ")
plt.figure()
plt.imshow(V.astype(np.uint8),cmap = 'gray')
plt.show()

print("\nHistograma imaginii in domeniul frecventa: \n")
plt.figure()
(n,bins,patches)=plt.hist(V.ravel(),256,[0,255],density=True,cumulative=True)
plt.show()
```

- Se face apelul unei funcții care să ordoneze matricea Hadamard generată anterior și se repetă ultimii 3 pași și pentru matricea ordonată.

APEL:

```
H_ord=ordonare(H)
print("\nMatricea Hadamard de ordinul ",N," ordonata: ")
print(H_ord,"\n")
```

```
aux=np.dot(H_ord,U)
V2=np.dot(aux,H_ord)
print("\nMatricea transformarii in forma ordonata: ")
print(V2,"\n")

print("\nImaginea transformata(in forma ordonata): ")
plt.figure()
plt.imshow(V2.astype(np.uint8),cmap = 'gray')
plt.show()

print("\nHistograma imaginii in domeniul frecventa: ")
plt.figure()
(n,bins,patches)=plt.hist(V2.ravel(),256,[0,255],density=True,cumulative=True)
plt.show()
```

DEFINIRE FUNCȚIE:

```
"""functie ce ordoneaza o matrice primita ca si parametru
in functie de numarul de schimbari de semn de pe fiecare linie"""
def ordonare(H):
    nr_linii=len(H)
    nr_coloane=len(H[0])
    v=np.zeros(nr_linii,dtype=int)
    k=0
    for i in range(nr_linii):
        nr=0;
        for j in range(nr_coloane-1):
            if H[i][j] != H[i][j+1]:
                nr+=1
        v[k]=nr
        k=k+1
    #print(v)
    """ in vectorul am obtinut valorea numarului schimbarilor de semn pentru fiecare linie in parte,
    in ordinea liniilor
    """
    H_nou=np.zeros((nr_linii,nr_coloane))
    i=0
    for k in range(nr_linii):
        for j in range(nr_coloane):
            H_nou[v[k]][j]=H[i][j]
        i+=1
    return H_nou
```

- Matricea imagine și matricea transformării se trimit ca și parametri unei funcții care să calculeze energia în vederea verificării teoremei de conservare a energiei.

APEL+VERIFICARE:

```
"""Conservarea energiei"""
membru_stang=energie(V,N)
print("\nEnergia in domeniul imaginii: ",membru_stang,"\n")
membru_drept=energie(U,N)
print("\nEnergia in domeniul transformat: ",membru_drept,"\n")
if membru_stang==membru_drept:
    print("\nENERGIA SE CONSERVA!!!")
else:
    print("\nENERGIA NU SE CONSERVA!!!")
```

DEFINIRE FUNCȚIE:

```
"""suma patratelor modulelor coeficientilor"""
def energie(M,N):
    suma=0
    for m in range(0,N):
        for n in range(0,N):
            suma+=math.pow(np.abs(M[m][n]),2)
    return suma
```

- Refacem imaginea și analizăm parametrii privind eroarea la refacere;

```
"""Imaginea refacuta"""
aux=np.dot(np.transpose(np.linalg.inv(H_ord)),V2)
V_out=np.dot(aux,np.transpose(H_ord))
print("\nImaginea refacuta: ")
plt.figure()
plt.imshow(V_out,cmap='gray')
plt.show()
print("\nHistograma imaginii refacute")
plt.figure()
(n,bins,patches)=plt.hist(V_out.ravel(),256,[0,255],density=True,cumulative=True)
plt.show()

"""Eroarea la refacere"""
print('\nEroarea la refacere: ')
mse = (np.square(U.astype(float) - V_out.astype(float))).mean()
print('MSE = ',mse)
```

- Aplicăm o mască de filtrare dreptunghiulară matricei transformării prin înmulțirea dintre mască și matricea transformării și apoi refacem imaginea;

```
"""Filtrare in domeniul frecventa"""
PercentKeptCoeffs=0.2
H,W=V2.shape
CoeffsMask=np.zeros(V2.shape)
CoeffsMask[:np.int(PercentKeptCoeffs*H),:np.int(PercentKeptCoeffs*W)]=1
print("\nMasca de filtrare: ")
plt.figure()
plt.imshow((CoeffsMask*255).astype(np.uint8),cmap = 'gray')
plt.show()
print("\nMasca de filtrare sub forma matriceala: ")
print(CoeffsMask)

"""Se inmulteste matricea imaginii in domeniul transformat cu masca de filtrare definita anterior"""
FiltImg=np.multiply(V2,CoeffsMask)
print("\nMatricea dupa filtrare: ")
print(FiltImg)
print("\nImaginea in domeniul frecventa: ")
plt.figure()
plt.imshow(FiltImg,cmap='gray')
plt.show()
```

- Afișăm imaginea refăcută împreună cu histograma acesteia;

```
"""Aplicam transfromata inversa"""
X=np.linalg.inv(H_ord)
OutImgFilt=(np.dot(np.dot(X,FiltImg),H_ord)).astype(np.uint8)

print("\nRezultatul filtrarii: ")
plt.figure()
plt.imshow(OutImgFilt,cmap='gray')
plt.show()
print("\nHistograma imaginii filtrate: ")
plt.figure()
(n,bins,patches)=plt.hist(OutImgFilt.ravel(),256,[0,255],density=True,cumulative=True)
plt.show()
```

- Calculăm capacitatea de compactare a energiei blocului transformat și transformăm valoarea rezultată în procente.

```
    """Compactarea energiei"""
    nr=0
    for i in range(nr_linii):
        for j in range(nr_coloane):
            if np.abs(OutImgFilt[i][j])!=0:
                nr+=1
    p=nr/(nr_linii*nr_coloane)
    print("\nCoeficientul de compactare este de",p*100,"%. ")
```

- Analizăm parametrii privind eroarea la refacere.

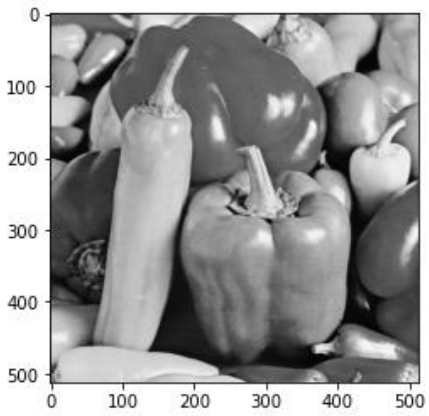
```
"""Eroarea la refacere"""
print('\nComparam imaginea filtrata cu imaginea originala.')
mse = (np.square(U.astype(float) - OutImgFilt.astype(float))).mean()
print('MSE = ',mse)
```

4.Rezultate experimentale

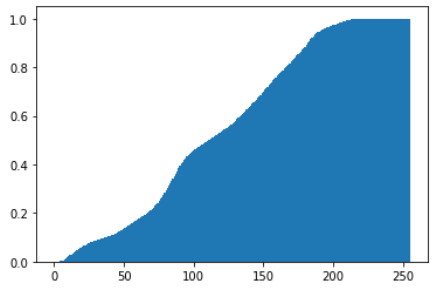
În continuare, pentru a evita inserarea de capturi de ecran, am salvat rezultatele din consolă într-un fișier html, urmând a le importa de acolo în prezentul document. Am rulat programul pentru un set de 5 imagini de dimensiuni pătratice, obținând următoarele rezultate:

- Pentru imaginea nr. 1:

```
Python 3.8.3 (default, Jul 2 2020, 17:30:36) [MSC
v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more
information.

IPython 7.16.1 -- An enhanced Interactive Python.

In [28]:
runfile('C:/Users/CRIS/Desktop/proiect_final/proiect
_v2/proiect1.py',
wdir='C:/Users/CRIS/Desktop/proiect_final/proiect_v2
')
Test functionare algoritm:

Matricea Hadamard de ordinul 8:
[[ 1 1 1 1 1 1 1 1]
 [ 1 -1 1 -1 1 -1 1 -1]
 [ 1 1 -1 -1 1 1 -1 -1]
 [ 1 -1 -1 1 1 -1 -1 1]
 [ 1 1 1 1 -1 -1 -1 -1]
 [ 1 -1 1 -1 -1 1 -1 1]
 [ 1 1 -1 -1 -1 -1 1 1]
 [ 1 -1 -1 1 -1 1 1 -1]]

Matricea Hadamard de ordinul 8 ordonata:
[[ 1. 1. 1. 1. 1. 1. 1. 1.]
 [ 1. 1. 1. 1. -1. -1. -1. -1.]
 [ 1. 1. -1. -1. -1. -1. 1. 1.]
 [ 1. 1. -1. -1. 1. 1. -1. -1.]
 [ 1. -1. -1. 1. 1. -1. -1. 1.]
 [ 1. -1. -1. 1. -1. 1. 1. -1.]
 [ 1. -1. 1. -1. -1. 1. -1. 1.]
 [ 1. -1. 1. -1. 1. -1. 1. -1.]]

Imaginea originala:
```



Histograma imaginii originale:



Matricea imagine:
[[27. 54. 57. ... 72. 78. 52.]
[34. 109. 104. ... 167. 153. 148.]
[35. 112. 106. ... 163. 163. 150.]
...
[24. 114. 122. ... 193. 189. 180.]
[24. 111. 100. ... 189. 187. 161.]
[25. 76. 118. ... 176. 185. 190.]]

Numarul de linii: 512

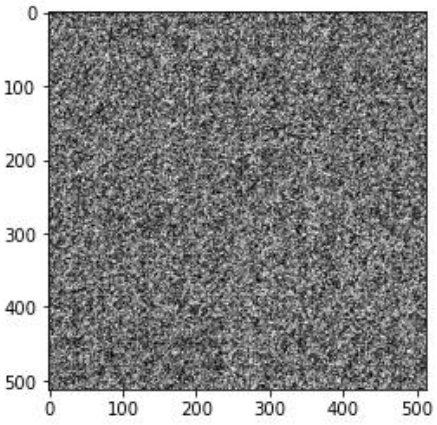
Numarul de coloane: 512

Matricea Hadamard de ordinul 512 :
[[0.04419417 0.04419417 0.04419417 ... 0.04419417
0.04419417
0.04419417]
[0.04419417 -0.04419417 0.04419417 ... -0.04419417
0.04419417
-0.04419417]
[0.04419417 0.04419417 -0.04419417 ... 0.04419417 -
0.04419417
-0.04419417]
...
[0.04419417 -0.04419417 0.04419417 ... 0.04419417 -
0.04419417
0.04419417]
[0.04419417 0.04419417 -0.04419417 ... -0.04419417
0.04419417
0.04419417]
[0.04419417 -0.04419417 -0.04419417 ... 0.04419417
0.04419417
-0.04419417]]

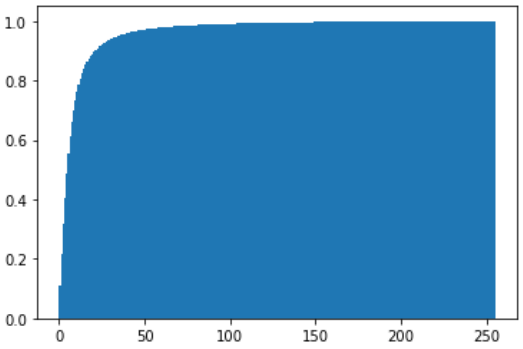
Matricea Hadamard de ordinul 512 ordonata:
[[0.04419417 0.04419417 0.04419417 ... 0.04419417
0.04419417
0.04419417]
[0.04419417 0.04419417 0.04419417 ... -0.04419417 -
0.04419417
-0.04419417]
[0.04419417 0.04419417 0.04419417 ... 0.04419417
0.04419417
0.04419417]
...
[0.04419417 -0.04419417 0.04419417 ... -0.04419417
0.04419417
-0.04419417]
[0.04419417 -0.04419417 0.04419417 ... 0.04419417 -
0.04419417
0.04419417]
[0.04419417 -0.04419417 0.04419417 ... -0.04419417
0.04419417
-0.04419417]]

Matricea transformarii in forma neordonata:
[[5.76768105e+04 -8.00917969e+01 -7.46582031e+01
... -9.98027344e+01
-1.25697266e+02 -8.04824219e+01]
[-7.90097656e+01 -1.40429688e+00 -2.98828125e-01 ...
4.53320312e+00
3.87304688e+00 3.33007812e+00]
[-9.47949219e+01 -3.53320313e+00 4.34570312e+00 ...
1.24804687e+00
1.02832031e+01 6.92773438e+00]
...
[-5.97402344e+01 4.84179688e+00 -1.29492188e+00 ...
-2.95507813e+00
2.61132812e+00 -7.25195312e+00]
[-2.54472656e+01 9.00390625e-01 -3.36914063e+00 ...
-4.24023438e+00
-1.57226563e+00 1.66738281e+01]
[-7.02988281e+01 -8.96484375e-01 -4.10351563e+00 ...
5.43945312e+00
1.52949219e+01 -1.20117187e+00]]

Imaginea transformata(in forma neordonata):

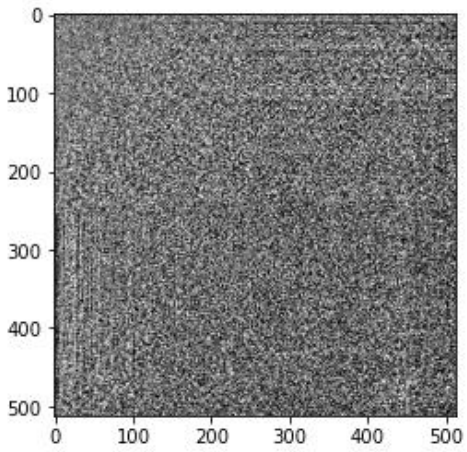


Histograma imaginii in domeniul frecventa:

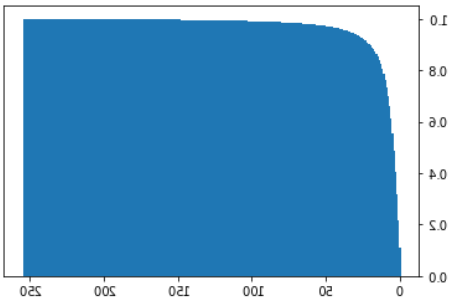


Matricea transformarii in forma ordonata:
[[5.76768105e+04 -1.83669922e+02 -3.68708984e+02
... -1.12869141e+02
-7.82753906e+01 -8.00917969e+01]
[3.55868945e+03 -3.98208008e+03 1.37248242e+03 ...
-1.26152344e+01
5.86132813e+00 -2.30410156e+01]
[-9.84826172e+02 1.11662305e+03 5.50013086e+03 ...
1.70507813e+00
-1.15429687e+00 -7.33789062e+00]
...
[-1.17427734e+02 1.31152344e+01 -9.83789062e+00 ...
1.18164063e+00
9.55078125e-01 -3.83789063e+00]
[-4.88808594e+01 2.85058594e+01 2.15136719e+01 ...
4.48632813e+00
7.51953125e-01 -2.69726563e+00]
[-7.90097656e+01 2.35351563e+00 1.36347656e+01 ...
5.76171875e-01
3.08398437e+00 -1.40429688e+00]]

Imaginea transformata(in forma ordonata):



Histograma imaginii in domeniul frecventa:

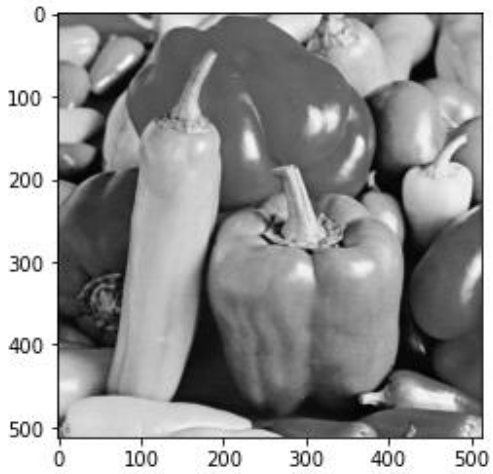


Energia in domeniul imaginii: 4078827247.0

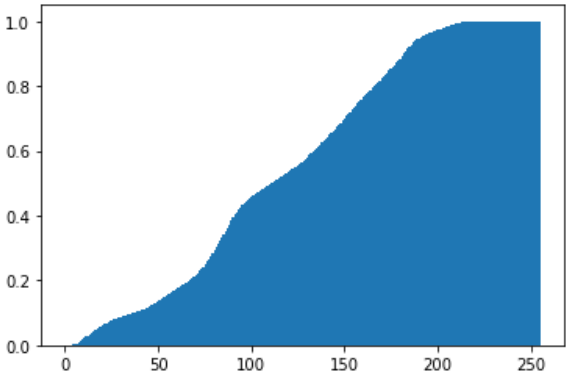
Energia in domeniul transformat: 4078827247.0

ENERGIA SE CONSERVA!!!

Imaginea refacuta:

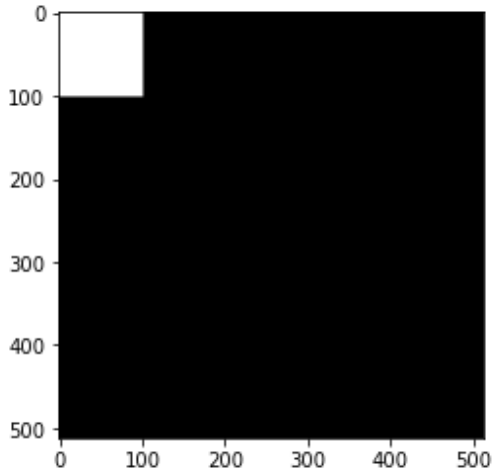


Histograma imaginii refacute



Eroarea la refacere:
MSE = 2.6851480103143044e-24

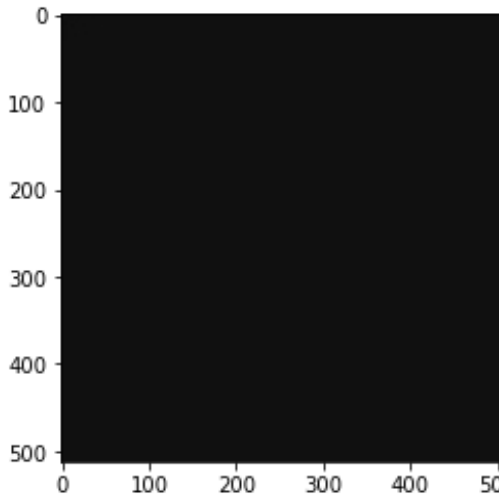
Masca de filtrare:



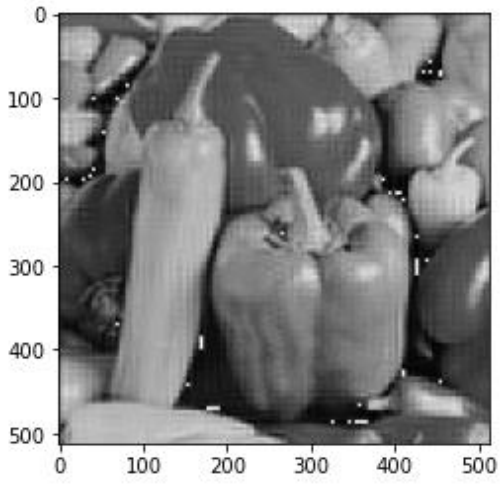
Masca de filtrare sub forma matriceala:
[[1. 1. 1. ... 0. 0. 0.]
[1. 1. 1. ... 0. 0. 0.]
[1. 1. 1. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]

Matricea dupa filtrare:
[[57676.81054688 -183.66992188 -368.70898438 ... -0.
-0. -0.]
[3558.68945312 -3982.08007813 1372.48242188 ... -0.
0. -0.]
[-984.82617188 1116.62304687 5500.13085938 ... 0.
-0. -0.]
...
[-0. 0. -0. ... 0.
0. -0.]
[-0. 0. 0. ... 0.
0. -0.]
[-0. 0. 0. ... 0.
0. -0.]]

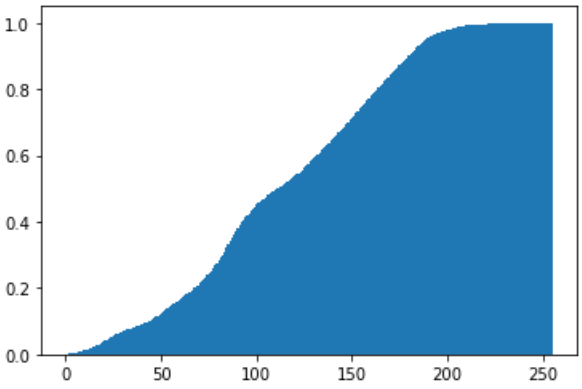
Imaginea in domeniul frecventa:




```
Coeficientul de compactare este de 3.96881103515625
%.
Rezultatul filtrarii:
```



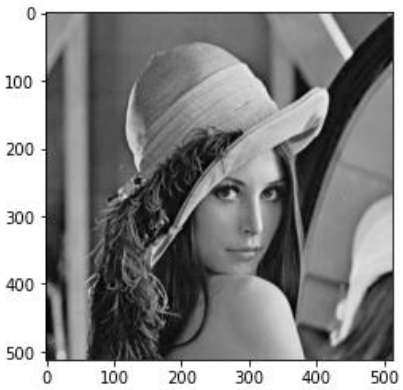

```
Histograma imaginii filtrate:
```




```
Comparam imaginea filtrata cu imaginea originala.
MSE = 386.5731544494629

In [29]:
```

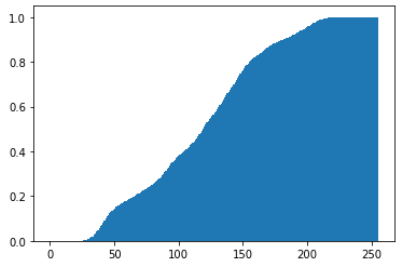
- Pentru imaginea nr. 2:

```
In [33]: runfile('C:/Users/CRIS/Desktop/proiect_final/proiect_v2/proiect1.py',
wdir='C:/Users/CRIS/Desktop/proiect_final/proiect_v2')
Test functionare algoritm:

Matricea Hadamard de ordinul 8:
[[ 1 1 1 1 1 1 1 1]
 [ 1 -1 1 -1 1 -1 1 -1]
 [ 1 1 -1 -1 1 1 -1 -1]
 [ 1 -1 -1 1 1 -1 -1 1]
 [ 1 1 1 1 -1 -1 -1 -1]
 [ 1 -1 1 -1 -1 1 -1 1]
 [ 1 1 -1 -1 -1 -1 1 1]
 [ 1 -1 -1 1 -1 1 1 -1]]

Matricea Hadamard de ordinul 8 ordonata:
[[ 1. 1. 1. 1. 1. 1. 1. 1.]
 [ 1. 1. 1. 1. -1. -1. -1. -1.]
 [ 1. 1. -1. -1. -1. -1. 1. 1.]
 [ 1. 1. -1. -1. 1. 1. -1. -1.]
 [ 1. -1. -1. 1. 1. -1. -1. 1.]
 [ 1. -1. -1. 1. -1. 1. 1. -1.]
 [ 1. -1. 1. -1. -1. 1. -1. 1.]
 [ 1. -1. 1. -1. 1. -1. 1. -1.]]
```




```
Histograma imaginii originale:
```

```
Imaginea originala:
```



Matricea imagine:
[[154. 154. 154. ... 162. 146. 119.]
[154. 154. 154. ... 162. 146. 119.]
[154. 154. 154. ... 162. 146. 119.]
...
[35. 35. 41. ... 94. 91. 89.]
[36. 36. 46. ... 94. 96. 98.]
[36. 36. 46. ... 94. 96. 98.]]

Numarul de linii: 512

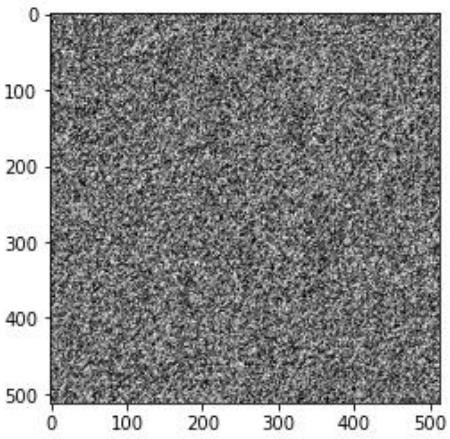
Numarul de coloane: 512

Matricea Hadamard de ordinul 512 :
[[0.04419417 0.04419417 0.04419417 ... 0.04419417
0.04419417
0.04419417]
[0.04419417 -0.04419417 0.04419417 ... -0.04419417
0.04419417
-0.04419417]
[0.04419417 0.04419417 -0.04419417 ... 0.04419417 -
0.04419417
-0.04419417]
...
[0.04419417 -0.04419417 0.04419417 ... 0.04419417 -
0.04419417
0.04419417]
[0.04419417 0.04419417 -0.04419417 ... -0.04419417
0.04419417
0.04419417]
[0.04419417 -0.04419417 -0.04419417 ... 0.04419417
0.04419417
-0.04419417]]

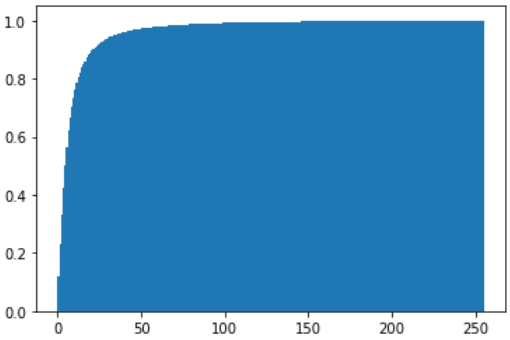
Matricea Hadamard de ordinul 512 ordonata:
[[0.04419417 0.04419417 0.04419417 ... 0.04419417
0.04419417
0.04419417]
[0.04419417 0.04419417 0.04419417 ... -0.04419417 -
0.04419417
-0.04419417]
[0.04419417 0.04419417 0.04419417 ... 0.04419417
0.04419417
0.04419417]
...
[0.04419417 -0.04419417 0.04419417 ... -0.04419417
0.04419417
-0.04419417]
[0.04419417 -0.04419417 0.04419417 ... 0.04419417 -
0.04419417
0.04419417]
[0.04419417 -0.04419417 0.04419417 ... -0.04419417
0.04419417
-0.04419417]]

Matricea transformarii in forma neordonata:
[[5.91558887e+04 -3.04511719e+01 -4.19355469e+01
... 1.27011719e+01
2.49511719e+01 7.94726563e+00]
[8.64648437e+00 1.79882812e+00 2.35351562e+00 ...
1.88867187e+00
3.03710937e+00 -2.75585938e+00]
[1.68574219e+01 3.68945313e+00 7.23632813e+00 ... -
9.86328125e-01
2.01367188e+00 -8.58398437e+00]
...
[-4.90039063e+00 -3.18554688e+00 -5.19335937e+00 ...
-5.72265625e-01
2.48046875e-01 -9.19921875e-01]
[-1.20566406e+01 3.49414063e+00 -8.22460937e+00 ...
5.31054688e+00
5.99804687e+00 -8.41210938e+00]
[-1.07050781e+01 -1.34960938e+00 -6.54296875e-01 ...
-8.00781250e-02
3.47460938e+00 -2.53710937e+00]]

Imaginea transformata(in forma neordonata):

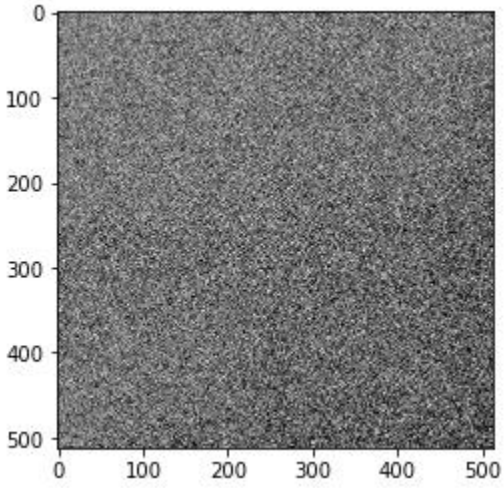


Histograma imaginii in domeniul frecventa:

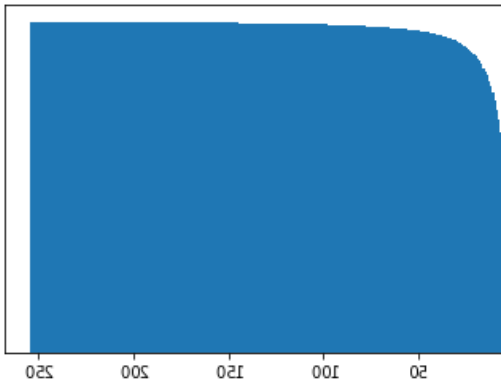


Matricea transformarii in forma ordonata:
[[5.91558887e+04 -6.96876367e+03 -1.25409961e+03
... -7.86132812e+00
-3.86269531e+01 -3.04511719e+01]
[3.38709570e+03 3.55932617e+03 -3.66425195e+03 ...
-4.98046875e-01
5.22851562e+00 2.50996094e+01]
[-8.04080078e+02 1.32634570e+03 -2.88827930e+03 ...
5.01074219e+01
-3.34472656e+01 2.15097656e+01]
...
[-2.40722656e+01 -1.09980469e+01 5.32519531e+01 ...
-7.12890625e-01
1.90429687e+00 9.78515625e-01]
[1.39082031e+01 -5.57226562e+00 -1.82128906e+01 ...
-4.23242187e+00
2.59765625e-01 3.13867188e+00]
[8.64648437e+00 5.79882813e+00 9.79101562e+00 ...
2.71484375e-01
5.84960937e+00 1.79882812e+00]]

Imaginea transformata(in forma ordonata):



Histograma imaginii in domeniul frecventa:

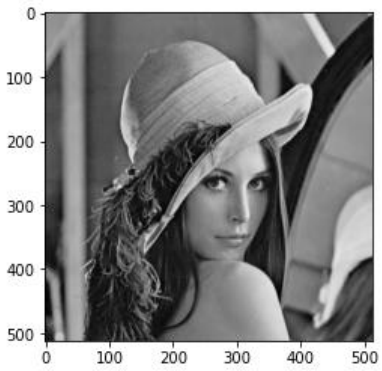


Energia in domeniul imaginii: 4117535069.0

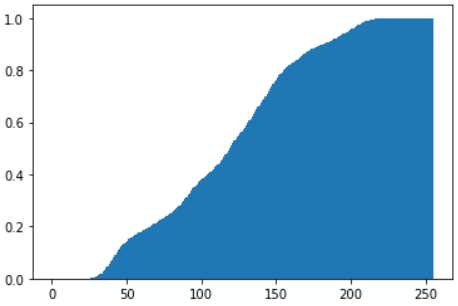
Energia in domeniul transformat: 4117535069.0

ENERGIA SE CONSERVA!!!

Imaginea refacuta:

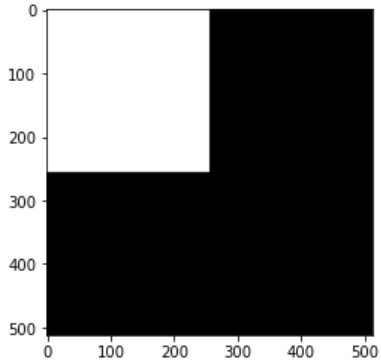


Histograma imaginii refacute



Eroarea la refacere:
MSE = 2.762693109297264e-24

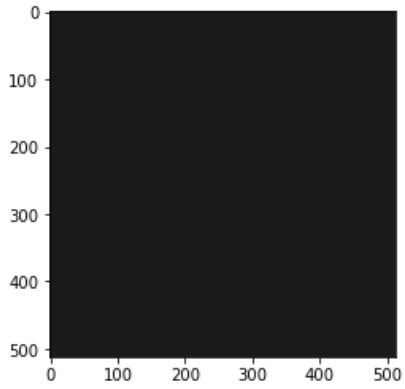
Masca de filtrare:



Masca de filtrare sub forma matriceala:
[[1. 1. 1. ... 0. 0. 0.]
[1. 1. 1. ... 0. 0. 0.]
[1. 1. 1. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]

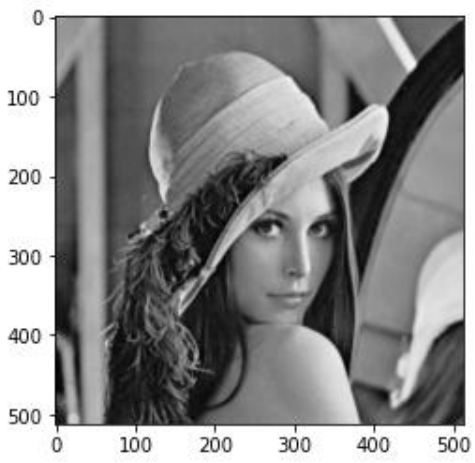
Matricea dupa filtrare:
[[59155.88867188 -6968.76367188 -1254.09960938 ... -0.
-0. -0.]
[3387.09570312 3559.32617187 -3664.25195312 ... -0.
0. 0.]
[-804.08007812 1326.34570313 -2888.27929687 ... 0.
-0. 0.]
...
[-0. -0. 0. ... -0.
0. 0.]
[0. -0. -0. ... -0.
0. 0.]
[0. 0. 0. ... 0.
0. 0.]]

Imaginea in domeniul frecventa:

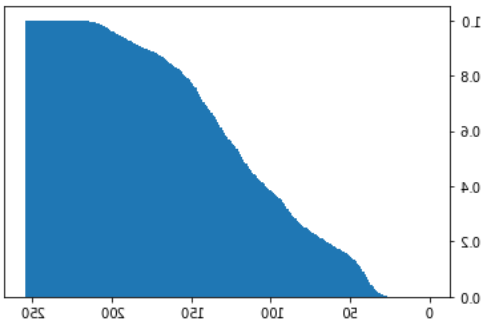


Coeficientul de compactare este de 25.0 %

Rezultatul filtrarii:



Histograma imaginii filtrate:



Comparam imaginea filtrata cu imaginea originala.
MSE = 45.57278060913086

In [34]:

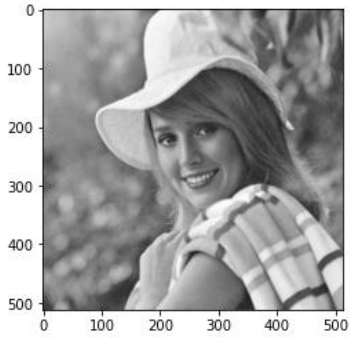
- Pentru imaginea nr. 3:

In [10]:
runfile('C:/Users/CRIS/Desktop/proiect_final/proiect_v2/proiect
1.py', wdir='C:/Users/CRIS/Desktop/proiect_final/proiect_v2')
Test functionare algoritm:

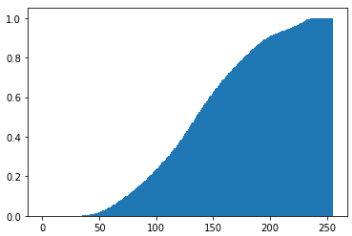
Matricea Hadamard de ordinul 8:
[[1 1 1 1 1 1 1 1]
[1 -1 1 -1 1 -1 1 -1]
[1 1 -1 -1 1 1 -1 -1]
[1 -1 -1 1 1 -1 -1 1]
[1 1 1 1 -1 -1 -1 -1]
[1 -1 1 -1 -1 1 -1 1]
[1 1 -1 -1 -1 -1 1 1]
[1 -1 -1 1 -1 1 1 -1]]

Matricea Hadamard de ordinul 8 ordonata:
[[1. 1. 1. 1. 1. 1. 1. 1.]
[1. 1. 1. 1. -1. -1. -1. -1.]
[1. 1. -1. -1. -1. -1. 1. 1.]
[1. 1. -1. -1. 1. 1. -1. -1.]
[1. -1. -1. 1. 1. -1. -1. 1.]
[1. -1. -1. 1. -1. 1. 1. -1.]
[1. -1. 1. -1. -1. 1. -1. 1.]
[1. -1. 1. -1. 1. -1. 1. -1.]]

Imaginea originala:



Histograma imaginii originale:



Matricea imagine:
[[190. 190. 190. ... 109. 87. 90.]
[190. 190. 190. ... 101. 99. 76.]
[190. 190. 190. ... 103. 92. 92.]
...
[71. 71. 67. ... 172. 176. 175.]
[72. 72. 69. ... 169. 167. 167.]
[0. 72. 69. ... 169. 167. 167.]]

Numarul de linii: 512

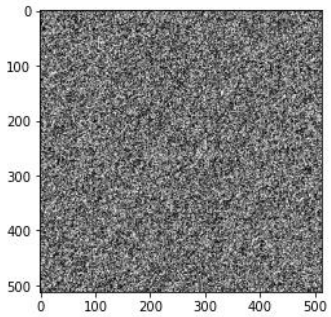
Numarul de coloane: 512

Matricea Hadamard de ordinul 512 :
[[0.04419417 0.04419417 0.04419417 ... 0.04419417 0.04419417
0.04419417]
[0.04419417 -0.04419417 0.04419417 ... -0.04419417 0.04419417
-0.04419417]
[0.04419417 0.04419417 -0.04419417 ... 0.04419417 -0.04419417
-0.04419417]
...
[0.04419417 -0.04419417 0.04419417 ... 0.04419417 -0.04419417
0.04419417]
[0.04419417 0.04419417 -0.04419417 ... -0.04419417 0.04419417
0.04419417]
[0.04419417 -0.04419417 -0.04419417 ... 0.04419417 0.04419417
-0.04419417]]

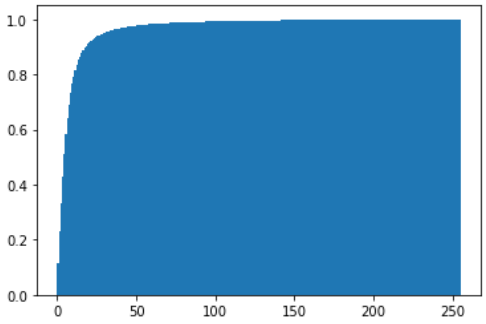
Matricea Hadamard de ordinul 512 ordonata:
[[0.04419417 0.04419417 0.04419417 ... 0.04419417 0.04419417
0.04419417]
[0.04419417 0.04419417 0.04419417 ... -0.04419417 -0.04419417
-0.04419417]
[0.04419417 0.04419417 0.04419417 ... 0.04419417 0.04419417
0.04419417]
...
[0.04419417 -0.04419417 0.04419417 ... -0.04419417 0.04419417
-0.04419417]
[0.04419417 -0.04419417 0.04419417 ... 0.04419417 -0.04419417
0.04419417]
[0.04419417 -0.04419417 0.04419417 ... -0.04419417 0.04419417
-0.04419417]]

Matricea transformarii in forma neordonata:
[[6.98145352e+04 8.17968750e+00 1.18085937e+01 ... -
2.86054688e+01
-3.50937500e+01 7.38281250e-01]
[5.39843750e+00 -2.05078125e+00 4.28125000e+00 ... -
2.09375000e+00
-8.28515625e+00 1.12500000e+00]
[1.69179687e+01 -1.82734375e+01 2.92968750e-01 ... -
6.44531250e-01
-1.36718750e+00 8.12890625e+00]
...
[1.84765625e+00 7.81250000e-03 -3.26171875e+00 ... -
1.14101562e+01
-8.12500000e+00 -1.83593750e-01]
[-3.06250000e+00 -3.45703125e+00 1.57031250e+00 ...
2.85937500e+00
-1.32304687e+01 -9.04687500e+00]
[-2.53906250e-01 3.72656250e+00 2.16015625e+00 ...
2.76953125e+00
5.30468750e+00 -5.89843750e-01]]

Imaginea transformata(in forma neordonata):

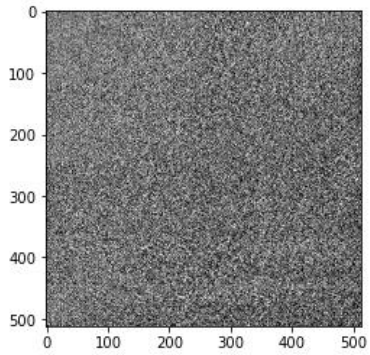


Histograma imaginii in domeniul frecventa:

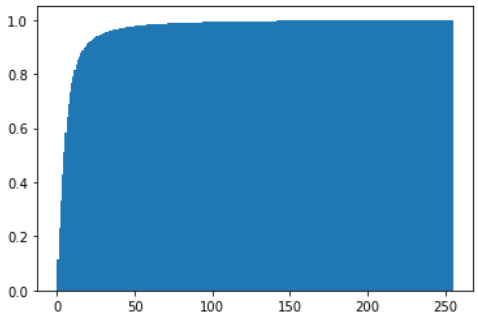


Matricea transformarii in forma ordonata:
[[6.98145352e+04 5.01582812e+03 -2.30639062e+03 ... -
9.55078125e+00
-1.49023438e+01 8.17968750e+00]
[-1.78026953e+03 8.17374219e+03 -1.24415625e+03 ... -
3.09648438e+01
1.01953125e+00 2.35078125e+01]
[6.00401562e+03 -1.87795703e+03 -2.16108984e+03 ...
1.86718750e+01
-1.58593750e+01 -2.57421875e+00]
...
[3.30820313e+01 -3.35937500e+01 -9.15625000e+00 ... -
7.38281250e-01
-2.43359375e+00 -7.81250000e-03]
[2.98828125e+01 -1.62304687e+01 2.97265625e+00 ... -
6.17187500e-01
4.73437500e+00 -2.18359375e+00]
[5.39843750e+00 1.18164062e+01 1.18359375e+00 ... 8.35937500e-
01
1.89843750e+00 -2.05078125e+00]]

Imaginea transformata(in forma ordonata):



Histograma imaginii in domeniul frecventa:

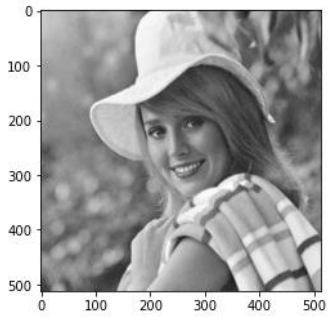


Energia in domeniul imaginii: 5430111494.0

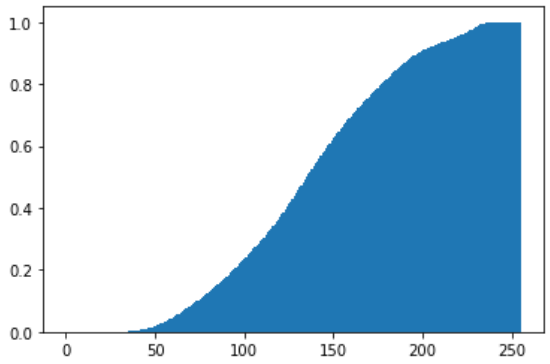
Energia in domeniul transformat: 5430111494.0

ENERGIA SE CONSERVA!!!

Imaginea refacuta:

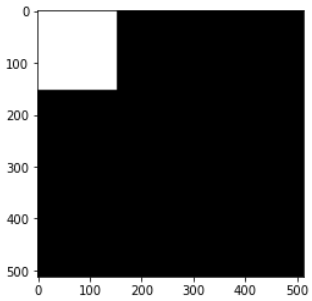


Histograma imaginii refacute



Eroarea la refacere:
MSE = 3.564926935576 49e-24

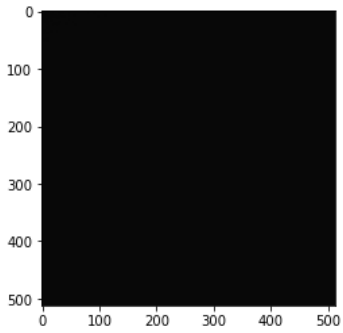
Masca de filtrare:



Masca de filtrare sub forma matriceala:
```
[[1. 1. 1. ... 0. 0. 0.]
[1. 1. 1. ... 0. 0. 0.]
[1. 1. 1. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]
```

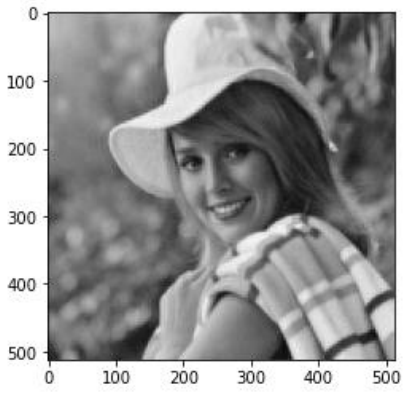
Matricea dupa filtrare:
```
[[69814.53515625 5015.828125 -2306.390625 ... -0.
-0. 0. ]
[-1780.26953125 8173.7421875 -1244.15625 ... -0.
0. 0. ]
[ 6004.015625 -1877.95703125 -2161.08984375 ... 0.
-0. -0. ]
...
[ 0. -0. -0. ... -0.
-0. -0. ]
[ 0. -0. 0. ... -0.
0. -0. ]
[ 0. 0. 0. ... 0.
0. -0. ]]
```

Imaginea in domeniul frecventa:

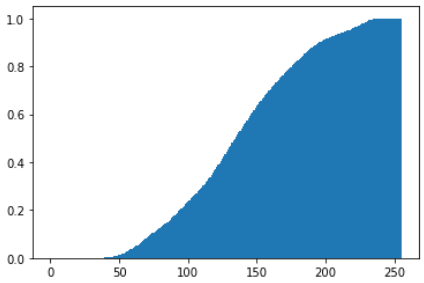


Coeficientul de compactare este de 8.929824829101562 %.

Rezultatul filtrarii:



Histograma imaginii filtrate:



Comparam imaginea filtrata cu imaginea originala.
MSE = 90.39240264892578

In [11]:

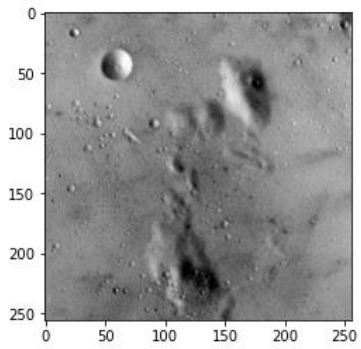
- Pentru imaginea nr. 4:

In [6]: runfile('C:/Users/CRIS/Desktop/proiect_final/proiect_v2/proiect1.py', wdir='C:/Users/CRIS/Desktop/proiect_final/proiect_v2')
Test functionare algoritm:

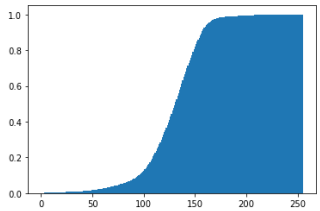
Matricea Hadamard de ordinul 8:
```
[[ 1 1 1 1 1 1 1 1]
[ 1 -1 1 -1 1 -1 1 -1]
[ 1 1 -1 -1 1 1 -1 -1]
[ 1 -1 -1 1 1 -1 -1 1]
[ 1 1 1 1 -1 -1 -1 -1]
[ 1 -1 1 -1 -1 1 -1 1]
[ 1 1 -1 -1 -1 -1 1 1]
[ 1 -1 -1 1 -1 1 1 -1]]
```

Matricea Hadamard de ordinul 8 ordonata:
```
[[ 1. 1. 1. 1. 1. 1. 1. 1.]
[ 1. 1. 1. 1. -1. -1. -1. -1.]
[ 1. 1. -1. -1. -1. -1. 1. 1.]
[ 1. 1. -1. -1. 1. 1. -1. -1.]
[ 1. -1. -1. 1. 1. -1. -1. 1.]
[ 1. -1. -1. 1. -1. 1. 1. -1.]
[ 1. -1. 1. -1. -1. 1. -1. 1.]
[ 1. -1. 1. -1. 1. -1. 1. -1.]]
```

Imaginea originala:



Histograma imaginii originale:



Matricea imagine:
[[144. 160. 144. ... 73. 71. 76.]
[144. 160. 144. ... 73. 71. 76.]
[142. 151. 144. ... 67. 66. 78.]
...
[129. 127. 133. ... 131. 146. 135.]
[111. 127. 127. ... 137. 146. 142.]
[133. 131. 133. ... 138. 149. 149.]]

Numarul de linii: 256

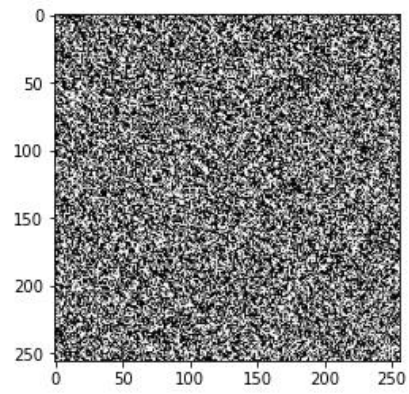
Numarul de coloane: 256

Matricea Hadamard de ordinul 256 :
[[0.0625 0.0625 0.0625 ... 0.0625 0.0625 0.0625]
[0.0625 -0.0625 0.0625 ... -0.0625 0.0625 -0.0625]
[0.0625 0.0625 -0.0625 ... 0.0625 -0.0625 -0.0625]
...
[0.0625 -0.0625 0.0625 ... -0.0625 0.0625 -0.0625]
[0.0625 0.0625 -0.0625 ... 0.0625 -0.0625 -0.0625]
[0.0625 -0.0625 -0.0625 ... -0.0625 -0.0625 0.0625]]

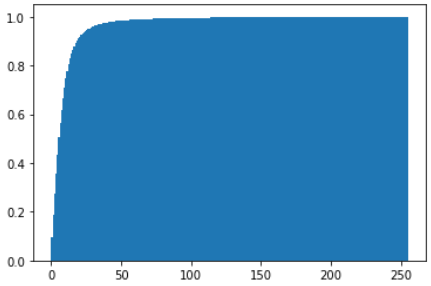
Matricea Hadamard de ordinul 256 ordonata:
[[0.0625 0.0625 0.0625 ... 0.0625 0.0625 0.0625]
[0.0625 0.0625 0.0625 ... -0.0625 -0.0625 -0.0625]
[0.0625 0.0625 0.0625 ... 0.0625 0.0625 0.0625]
...
[0.0625 -0.0625 0.0625 ... -0.0625 0.0625 -0.0625]
[0.0625 -0.0625 0.0625 ... 0.0625 -0.0625 0.0625]
[0.0625 -0.0625 0.0625 ... -0.0625 0.0625 -0.0625]]

Matricea transformarii in forma neordonata:
[[3.27065664e+04 -1.35742188e+01 -9.40234375e+00 ...
1.57929688e+01
3.82773438e+01 1.94492188e+01]
[1.23867188e+01 6.10546875e+00 -3.66015625e+00 ...
3.90625000e-03
2.98828125e+00 3.70703125e+00]
[3.03320312e+01 -6.33984375e+00 -7.96484375e+00 ...
4.02343750e-01
-5.59765625e+00 5.16796875e+00]
...
[-5.51953125e+00 4.02343750e-01 8.90234375e+00 ... -
5.74218750e-01
3.01953125e+00 -8.71093750e-01]
[-1.28867188e+01 8.39453125e+00 7.75390625e+00 ...
1.96992188e+01
1.52773438e+01 1.98085938e+01]
[-7.94140625e+00 -1.00195312e+01 -1.16015625e+00 ...
2.06640625e+00
-2.92968750e-01 -5.37109375e+00]]

Imaginea transformata(in forma neordonata):

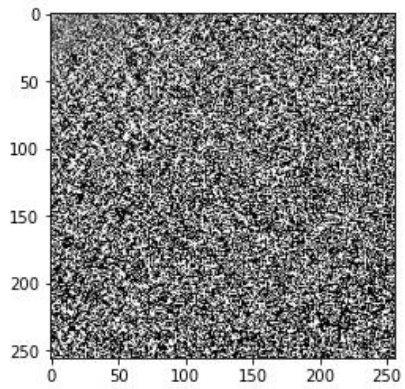


Histograma imaginii in domeniul frecventa:

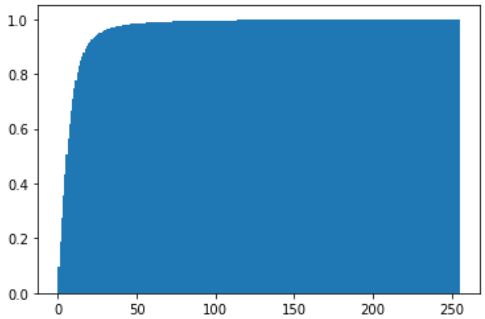


Matricea transformarii in forma ordonata:
[[3.27065664e+04 -6.22824219e+02 2.76076953e+03 ... -
3.12109375e+00
5.21757812e+01 -1.35742188e+01]
[2.54283203e+03 1.80863281e+02 -8.30449219e+02 ... -
4.69921875e+00
-2.00585938e+01 -7.16796875e+00]
[-3.10785156e+02 4.44152344e+02 -5.27441406e+02 ...
1.62304688e+01
8.96484375e+00 7.38671875e+00]
...
[-1.03867188e+01 -7.54296875e+00 1.14453125e+00 ... -
6.94921875e+00
9.37890625e+00 9.73828125e+00]
[-1.56640625e+00 -1.78515625e+00 -3.69140625e+00 ... -
1.42578125e+00
-1.50390625e+00 -4.70703125e+00]
[1.23867188e+01 4.49609375e+00 -4.47265625e+00 ...
1.46484375e+00
6.13671875e+00 6.10546875e+00]]

Imaginea transformata(in forma ordonata):



Histograma imaginii in domeniul frecventa:

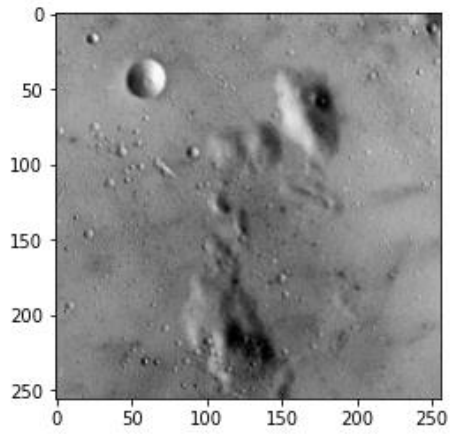


Energia in domeniul imaginii: 1120135341.0

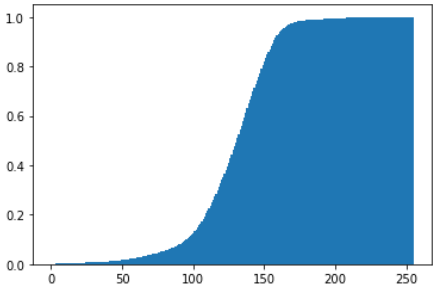
Energia in domeniul transformat: 1120135341.0

ENERGIA SE CONSERVA!!!

Imaginea refacuta:

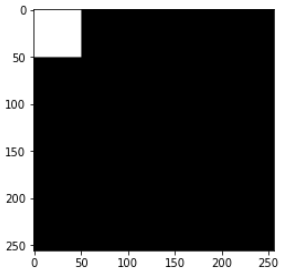


Histograma imaginii refacute



Eroarea la refacere:
MSE = 0.0

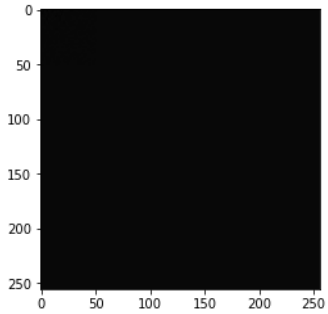
Masca de filtrare:



Masca de filtrare sub forma matriceala:
[[1. 1. 1. ... 0. 0. 0.]
[1. 1. 1. ... 0. 0. 0.]
[1. 1. 1. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]

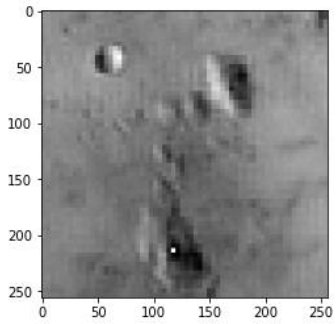
Matricea dupa filtrare:
[[32706.56640625 -622.82421875 2760.76953125 ... -0.
0. -0.]
[2542.83203125 180.86328125 -830.44921875 ... -0.
-0. -0.]
[-310.78515625 444.15234375 -527.44140625 ... 0.
0. 0.]
...
[-0. -0. 0. ... -0.
0. 0.]
[-0. -0. -0. ... -0.
-0. -0.]
[0. 0. -0. ... 0.
0. 0.]]

Imaginea in domeniul frecventa:

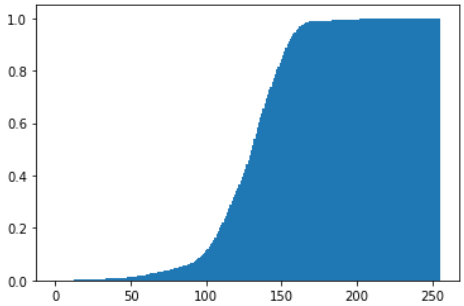


Coeficientul de compactare este de 3.96881103515625 %.

Rezultatul filtrarii:



Histograma imaginii filtrate:



Comparam imaginea filtrata cu imaginea originala.
MSE = 140.44361877441406

In [7]:

- Pentru imaginea nr. 5:

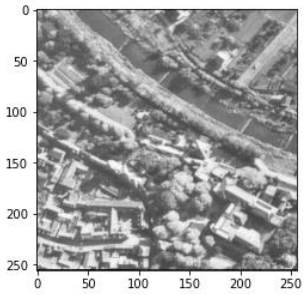
In [8]: runfile('C:/Users/CRIS/Desktop/proiect_final/proiect_v2/proiect1.py', wdir='C:/Users/CRIS/Desktop/proiect_final/proiect_v2')

Test functionare algoritm:

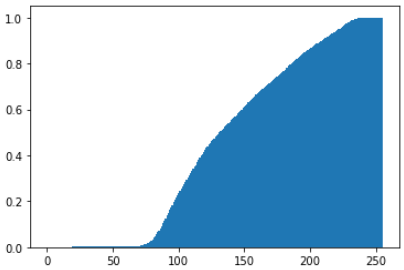
Matricea Hadamard de ordinul 8:
[[1 1 1 1 1 1 1 1]
[1 -1 1 -1 1 -1 1 -1]
[1 1 -1 -1 1 1 -1 -1]
[1 -1 -1 1 1 -1 -1 1]
[1 1 1 1 -1 -1 -1 -1]
[1 -1 1 -1 -1 1 -1 1]
[1 1 -1 -1 -1 -1 1 1]
[1 -1 -1 1 -1 1 1 -1]]

Matricea Hadamard de ordinul 8 ordonata:
[[1. 1. 1. 1. 1. 1. 1. 1.]
[1. 1. 1. 1. -1. -1. -1. -1.]
[1. 1. -1. -1. -1. -1. 1. 1.]
[1. 1. -1. -1. 1. 1. -1. -1.]
[1. -1. -1. 1. 1. -1. -1. 1.]
[1. -1. -1. 1. -1. 1. 1. -1.]
[1. -1. 1. -1. -1. 1. -1. 1.]
[1. -1. 1. -1. 1. -1. 1. -1.]]

Imaginea originala:



Histograma imaginii originale:



Matricea imagine:
[[122. 182. 184. ... 137. 136. 130.]
[129. 178. 188. ... 128. 135. 146.]
[166. 170. 185. ... 134. 131. 152.]
...
[102. 138. 125. ... 87. 82. 93.]
[92. 146. 127. ... 78. 83. 90.]
[2. 2. 4. ... 29. 25. 32.]]

Numarul de linii: 256

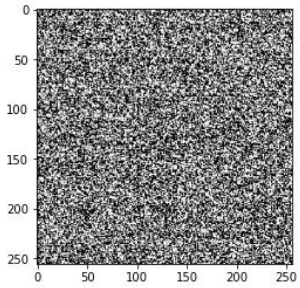
Numarul de coloane: 256

Matricea Hadamard de ordinul 256 :
[[0.0625 0.0625 0.0625 ... 0.0625 0.0625 0.0625]
[0.0625 -0.0625 0.0625 ... -0.0625 0.0625 -0.0625]
[0.0625 0.0625 -0.0625 ... 0.0625 -0.0625 -0.0625]
...
[0.0625 -0.0625 0.0625 ... -0.0625 0.0625 -0.0625]
[0.0625 0.0625 -0.0625 ... 0.0625 -0.0625 -0.0625]
[0.0625 -0.0625 -0.0625 ... -0.0625 -0.0625 0.0625]]

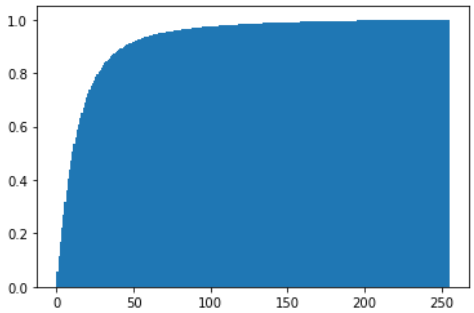
Matricea Hadamard de ordinul 256 ordonata:
[[0.0625 0.0625 0.0625 ... 0.0625 0.0625 0.0625]
[0.0625 0.0625 0.0625 ... -0.0625 -0.0625 -0.0625]
[0.0625 0.0625 0.0625 ... 0.0625 0.0625 0.0625]
...
[0.0625 -0.0625 0.0625 ... -0.0625 0.0625 -0.0625]
[0.0625 -0.0625 0.0625 ... 0.0625 -0.0625 0.0625]
[0.0625 -0.0625 0.0625 ... -0.0625 0.0625 -0.0625]]

Matricea transformarii in forma neordonata:
[[3.59698086e+04 -1.98398438e+01 -1.75390625e+00 ... -
1.41289062e+01
-8.80859375e+00 -5.16015625e+00]
[1.23027344e+02 -4.29296875e+00 -4.10156250e-01 ...
6.68359375e+00
-4.35546875e+00 -2.85156250e-01]
[8.94179688e+01 -2.51171875e+00 -1.77851562e+01 ...
1.53554688e+01
-1.72773438e+01 -1.43164062e+01]
...
[9.08867188e+01 -1.18359375e+00 1.18554688e+01 ... -
2.38281250e-01
5.50390625e+00 7.44921875e+00]
[7.27929688e+01 9.11328125e+00 4.19648438e+01 ...
2.65742188e+01
2.21914062e+01 2.41523438e+01]
[-1.12253906e+02 1.08203125e+00 3.10546875e+00 ... -
1.16015625e+00
-6.96484375e+00 1.16796875e+00]]

Imaginea transformata(in forma neordonata):

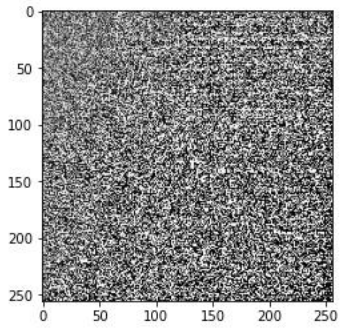


Histograma imaginii in domeniul frecventa:

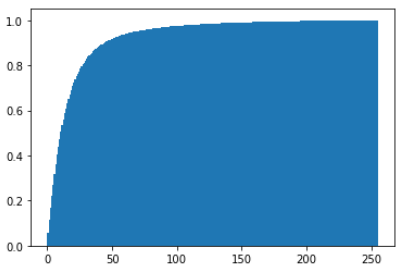


Matricea transformarii in forma ordonata:
[[3.59698086e+04 1.65542969e+02 1.02628906e+02 ...
4.71484375e+00
-1.51992188e+01 -1.98398438e+01]
[-8.50199219e+02 6.49957031e+02 -3.95128906e+02 ... -
2.04960938e+01
1.42929688e+01 8.73046875e+00]
[9.74882812e+01 2.19597656e+02 9.08085938e+01 ...
2.64453125e+00
-1.94257812e+01 1.33984375e+00]
...
[9.70820312e+01 1.88945312e+01 1.71484375e+00 ...
1.66015625e+00
-1.20703125e+00 -1.05078125e+00]
[-1.28542969e+02 -1.27773438e+01 1.18359375e+00 ... -
2.30468750e-01
1.13671875e+00 6.83593750e-01]
[1.23027344e+02 4.90234375e+00 1.29414062e+01 ...
1.63671875e+00
5.19531250e-01 -4.29296875e+00]]

Imaginea transformata(in forma ordonata):



Histograma imaginii in domeniul frecventa:

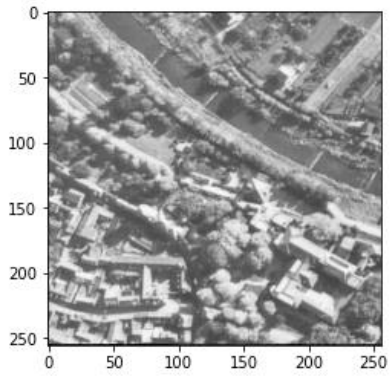


Energia in domeniul imaginii: 1428560619.0

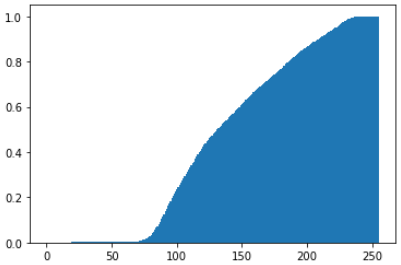
Energia in domeniul transformat: 1428560619.0

ENERGIA SE CONSERVA!!!

Imaginea refacuta:

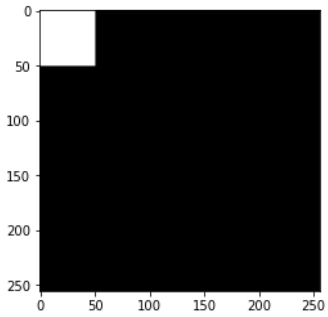


Histograma imaginii refacute



Eroarea la refacere:
MSE = 0.0

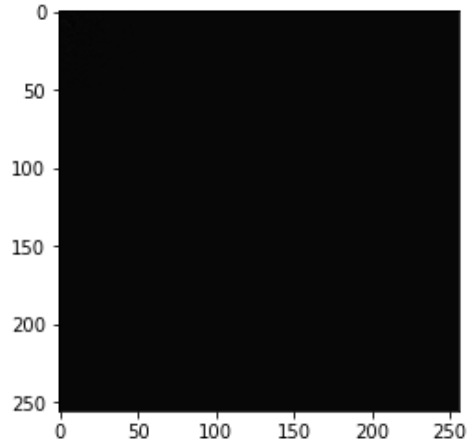
Masca de filtrare:



Masca de filtrare sub forma matriceala:
[[1. 1. 1. ... 0. 0. 0.]
[1. 1. 1. ... 0. 0. 0.]
[1. 1. 1. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]

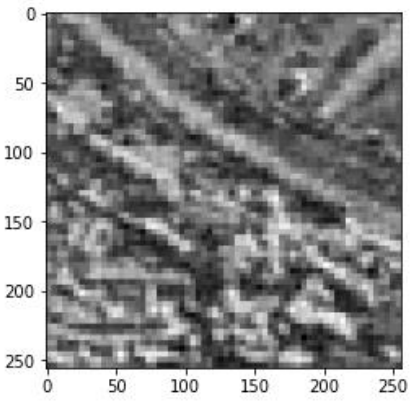
Matricea dupa filtrare:
[[35969.80859375 165.54296875 102.62890625 ... 0.
-0. -0.]
[-850.19921875 649.95703125 -395.12890625 ... -0.
0. 0.]
[97.48828125 219.59765625 90.80859375 ... 0.
-0. 0.]
...
[0. 0. 0. ... 0.
-0. -0.]
[-0. -0. 0. ... -0.
0. 0.]
[0. 0. 0. ... 0. 0. -0.]]

Imaginea in domeniul frecventa:

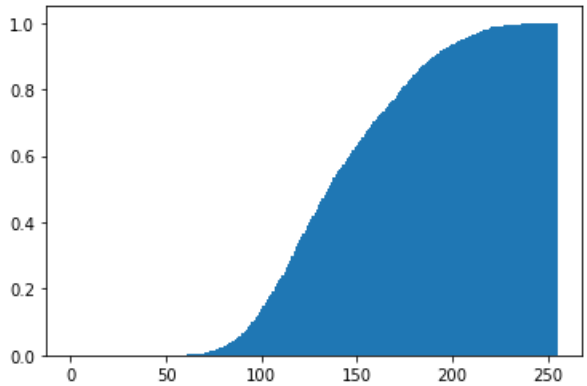


Coeficientul de compactare este de 3.96881103515625 %.

Rezultatul filtrarii:



Histograma imaginii filtrate:



Comparam imaginea filtrata cu imaginea originala.
MSE = 720.9340972900391

In [9]:

5.Concluzii

Folosind metoda pe care am abordat-o mai sus, se poate determina ușor, în mod recursiv transformata Hadamard ordonată a unui bloc de pixeli corespunzător unei matrici pătratice provenite dintr-o imagine de test, put imagine de test, obținând o bună compactare a energiei în domeniul transformat. De asemenea, am reușit să ilustrăm, în aceeași manieră, filtrarea imaginii și parametrii privind refacerea acesteia.

În ceea ce privește eventualele îmbunătățiri aduse programului, ceea ce s-ar mai putea face, în primă instanță, ar fi împărțirea unei imagini în blocuri mai mici, putând aplica transformata Hadamard pentru fiecare bloc în parte și concatenând apoi rezultatele. O altă idee ar putea fi reprezentată de rularea algoritmului, nu pe o imagine individuală, ci pe un set de imagini.

Bibliografie

[1] T. Stathaki, „Digital Image Processing," Imperial College of London, Departament of Electrical and Electronic Engineering.

[2] B. Vijay și J. Swathy, „An Efficient Hadamard Transform Oriented Digital Image-In-Image Watermarking," *International Journal for Research in Applied Science and Engineering Technology,* vol. 3, nr. V, pp. 576-580, 2015.

[3] Y. R. K. Rao și J. E. Hershey, Hadamard matrix analysis and synthesis with applications to communications and signal/image processing, Spinger Science+Business Media LLC, 1997.

[4] A. Vlaicu , Prelucrarea numerică a imaginilor, Cluj-Napoca: Editura Albastră, 1997.

[5] Transformări rectangulare-transformata Hadamard-Probleme rezolvate, [Online 19.10.2020],

[6] https://www.w3schools.com/python

Anexă-codul realizat în Python

```
# -*- coding: utf-8 -*-
"""
Created on Sun Nov  1 12:44:39 2020

@author: CRIS
"""

import cv2
import matplotlib.pyplot as plt
import numpy as np
import math

"""calculam in mod recursiv matricae Hadamard sub forma nenormalizata"""
def Hadamard(N):
    if N>2:
        Matrice1=np.concatenate((Hadamard(N/2),Hadamard(N/2)),axis=1,out=None)
        Matrice2=np.concatenate((Hadamard(N/2),-Hadamard(N/2)),axis=1,out=None)
        Matrice3=np.concatenate((Matrice1,Matrice2),axis=0,out=None)
        H=Matrice3
    else:
        aux2=1
        A=[[1,1],[1,-1]]
        H=np.dot(aux2,A)
    return H

"""functie ce ordoneaza o matrice primita ca si parametru
in functie de numarul de schimbari de semn de pe fiecare linie"""
def ordonare(H):
    nr_linii=len(H)
    nr_coloane=len(H[0])
    v=np.zeros(nr_linii,dtype=int)
    k=0
    for i in range(nr_linii):
        nr=0;
        for j in range(nr_coloane-1):
            if H[i][j] != H[i][j+1]:
                nr+=1
        v[k]=nr
        k=k+1
    #print(v)
    """ in vectorul am obtinut valorea numarului schimbarilor de semn pentru fiecare linie in parte,
    in ordinea liniilor
    """
    H_nou=np.zeros((nr_linii,nr_coloane))
    i=0
    for k in range(nr_linii):
        for j in range(nr_coloane):
            H_nou[v[k]][j]=H[i][j]
        i+=1
    return H_nou

"""suma patratelor modulelor coeficientilor"""
def energie(M,N):
    suma=0
    for m in range(0,N):
        for n in range(0,N):
            suma+=math.pow(np.abs(M[m][n]),2)
    return suma
"""functia main"""
if __name__ == "__main__":
    """Generarea matricii Hadamard"""
    print("Test functionare algoritm: ")
    Nn=8 #ordinul matricii
    H8=Hadamard(Nn) #apelul functiei
    print("\nMatricea Hadamard de ordinul 8: ")
    print(H8)
    H8_ord=ordonare(H8) #apelul functiei care returneaza matricea ordonata
    print("\nMatricea Hadamard de ordinul 8 ordonata: ")
    print(H8_ord)

    """Import imagine de test"""
```

```
Imagine_de_intrare="PeppersGri.bmp"
U=cv2.imread(Imagine_de_intrare,0).astype(float)

print("\nImaginea originala: \n")
plt.imshow(U,cmap='gray')
plt.show()

print("\nHistograma imaginii originale: ")
plt.figure()
(n,bins,patches)=plt.hist(U.ravel(),256,[0,255],density=True,cumulative=True)
plt.show()

print("\nMatricea imagine: ")
print(U,"\n")
nr_linii=len(U)
print("\nNumarul de linii: ",nr_linii)
nr_coloane=len(U[0])
print("\nNumarul de coloane: ",nr_coloane)

if nr_linii==nr_coloane:
    """Generarea matricii Hadamard"""
    N=nr_linii
    H=Hadamard(N)

    """Normalizare"""
    var=1/math.sqrt(N)
    H=np.dot(var,H)
    print("\nMatricea Hadamard de ordinul ",N,":")
    print(H,"\n")
    H_ord=ordonare(H)
    print("\nMatricea Hadamard de ordinul ",N," ordonata: ")
    print(H_ord,"\n")

    """Transformata Hadamard
    V=H*U*H
    """
    aux=np.dot(H,U) #H*U
    V=np.dot(aux,H) #(H*U)*H

    print("\nMatricea transformarii in forma neordonata: ")
    print(V)
    print("\nImaginea transformata(in forma neordonata): ")
    plt.figure()
    plt.imshow(V.astype(np.uint8),cmap = 'gray')
    plt.show()

    print("\nHistograma imaginii in domeniul frecventa: \n")
    plt.figure()
    (n,bins,patches)=plt.hist(V.ravel(),256,[0,255],density=True,cumulative=True)
    plt.show()

    aux=np.dot(H_ord,U)
    V2=np.dot(aux,H_ord)
    print("\nMatricea transformarii in forma ordonata: ")
    print(V2,"\n")

    print("\nImaginea transformata(in forma ordonata): ")
    plt.figure()
    plt.imshow(V2.astype(np.uint8),cmap = 'gray')
    plt.show()

    print("\nHistograma imaginii in domeniul frecventa: ")
    plt.figure()
    (n,bins,patches)=plt.hist(V2.ravel(),256,[0,255],density=True,cumulative=True)
    plt.show()

    """Conservarea energiei"""
    membru_stang=energie(V,N)
    print("\nEnergia in domeniul imaginii: ",membru_stang,"\n")
    membru_drept=energie(U,N)
    print("\nEnergia in domeniul transformat: ",membru_drept,"\n")
    if membru_stang==membru_drept:
        print("\nENERGIA SE CONSERVA!!!")
    else:
        print("\nENERGIA NU SE CONSERVA!!!")
```

```
"""Imaginea refacuta"""
aux=np.dot(np.transpose(np.linalg.inv(H_ord)),V2)
V_out=np.dot(aux,np.transpose(H_ord))
print("\nImaginea refacuta: ")
plt.figure()
plt.imshow(V_out,cmap='gray')
plt.show()
print("\nHistograma imaginii refacute")
plt.figure()
(n,bins,patches)=plt.hist(V_out.ravel(),256,[0,255],density=True,cumulative=True)
plt.show()

"""Eroarea la refacere"""
print('\nEroarea la refacere: ')
mse = (np.square(U.astype(float) - V_out.astype(float))).mean()
print('MSE = ',mse)

"""Filtrare in domeniul frecventa"""
PercentKeptCoeffs=0.2
H,W=V2.shape
CoeffsMask=np.zeros(V2.shape)
CoeffsMask[:np.int(PercentKeptCoeffs*H),:np.int(PercentKeptCoeffs*W)]=1
print("\nMasca de filtrare: ")
plt.figure()
plt.imshow((CoeffsMask*255).astype(np.uint8),cmap = 'gray')
plt.show()
print("\nMasca de filtrare sub forma matriceala: ")
print(CoeffsMask)

"""Se inmulteste matricea imaginii in domeniul transformat cu masca de filtrare definita anterior"""
FiltImg=np.multiply(V2,CoeffsMask)
print("\nMatricea dupa filtrare: ")
print(FiltImg)
print("\nImaginea in domeniul frecventa: ")
plt.figure()
plt.imshow(FiltImg,cmap='gray')
plt.show()

"""Compactarea energiei"""
nr=0
for i in range(nr_linii):
    for j in range(nr_coloane):
        if np.abs(FiltImg[i][j])!=0:
            nr+=1
p=nr/(nr_linii*nr_coloane)
print("\nCoeficientul de compactare este de",p*100,"%. ")

"""Aplicam transfromata inversa"""
X=np.linalg.inv(H_ord)
OutImgFilt=(np.dot(np.dot(X,FiltImg),H_ord)).astype(np.uint8)

print("\nRezultatul filtrarii: ")
plt.figure()
plt.imshow(OutImgFilt,cmap='gray')
plt.show()
print("\nHistograma imaginii filtrate: ")
plt.figure()
(n,bins,patches)=plt.hist(OutImgFilt.ravel(),256,[0,255],density=True,cumulative=True)
plt.show()

"""Eroarea la refacere"""
print('\nComparam imaginea filtrata cu imaginea originala.')
mse = (np.square(U.astype(float) - OutImgFilt.astype(float))).mean()
print('MSE = ',mse)
```