Exercițiul 1: Lucrul cu imagini în Matlab – noțiuni de bază: importul, procesarea, exportul

Acest exemplu arată cum putem citi o imagine, cum ajustăm contrastul din imagine iar în final cum salvăm imaginea procesată (ajustată) într-un fișier.

Cerințe pentru Exercițiul 1:

1. Implementați în Matlab exercițiul 1, folosind imaginea înclusă în toolboxul de Image Processing (cea din exemplu). *Atașați codul final și rezultatele obținute la fiecare pas*
2. Folosind codul deja scris la cerința 1, modificați Pasul 3 din exercițiu și încercați alte metode de îmbunătățire a contrastului (imadjust și adapthisteq). *Atașați rezultatele obținute folosind cele 2 metode de îmbunătățire a contrastului. Care vi se pare că are rezultatele cela mai bune?*
3. Creați un script matlab separat care să facă conversia imaginilor color în imagini alb-negru, salvând pe disc rezultatul. Pentru aceasta, puteți utiliza funcția Matlab `in_img=rgb2gray(in_img);` după ce veți citi imaginea ca în pasul 1. După conversie, o puteți salva pe disc ca în pasul 4 din exercitiu. *Atașați codul obținut și rezultatul unei rulări (imaginea color originală și cea rezultată)*
4. Folosind o altă imagine de dimensiune redusă (o puteți reduce folosind tooluri de Windows ca "Paint" sau "Snipping Tool"), rulați exercițiul 1 cu această nouă imagine. *Atașați rezultatele obținute (asemeni cerintei 1, dar fără codul matlab)*
    a. Dacă imaginea este color, atunci folositi convertorul color → alb-negru implementat la cerința 3.
5. Deschideți și utilizați aplicația Matlab numita Image Viewer. Folosiți poze personale (*făcute de voi, nu cu voi, de exemplu o clădire, un peisaj*) și explorați funcționalitățile disponibile. *Atasați cel mai interesant rezultat*
6. Folosiți o imagine personală atât în varianta color cât și alb-negru (color → alb-negru implementat la cerința 3) și verificați histogramele pentru cele două variante. *Atașați cele două histograme și spuneți pe scurt ce observați.*
7. Realizați un script simplu de matlab care încarcă o imagine personală și afișează informațiile disponibile pentru acea imagine (vezi pasul 5 din exercițiul 1). *Atașați rezultatul rulării scriptului pe o imagine*

OBS: pentru imaginile personale, puteți folosi calea absolută spre imagine, dacă aceasta nu se află în workspace. Ex: C:\Users\.....\block_diagram.png)

IMPLEMENTARE Exercitiu 1:

Pasul 1: Citiți și afișați o imagine

Citiți o imagine folosind comanda **imread**. Exemplul citește una dintre imaginile-exemplu incluse în Image Processing Toolbox (o imagine a unei fetițe) într-un fisier numit **pout.tif** și o stochează într-un array numit **I**.

```
I = imread('pout.tif');
```

Pentru a afișa imaginea, folosim funcția **imshow**. Puteți vizualiza, de asemenea, o imagine în aplicația **Image Viewer**. Funcția **imtool** deschide aplicația **Image Viewer** care prezintă un mediu integrat pentru afișarea imaginilor și efectuarea unor sarcini simple de procesare a imaginilor. Aplicația **Image Viewer** oferă toate funcțiile de afișare a imaginilor din **imshow**, dar oferă acces la alte câteva instrumente pentru navigarea și explorarea imaginilor, cum ar fi bare de defilare (scroll bars), instrumentul Regiunea Pixelilor, instrumentul Informații Imagine și instrumentul de Ajustare a Contrastului.

```
imshow(I);
```

Pasul 2: Verificați imaginea în workspace

Verificați cum funcția **imread** stochează datele imaginii în workspace, folosind comanda **whos**. Puteți verifica, de asemenea, variabila în browserul Workspace. Funcția **imread** returnează datele imaginii din variabila **I**, care este un array de elemente 291x240 de date uint8.

```
whos I
```

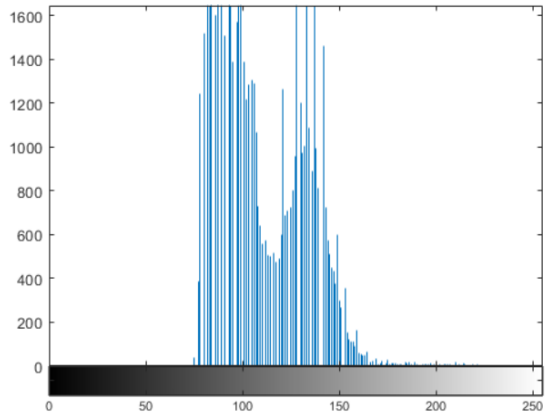
```
  Name        Size              Bytes  Class    Attributes

  I         291x240             69840  uint8
```

Pasul 3: Îmbunătățirea contrastului din imagine

Vizualizați distribuția intensității pixelilor din imagine (histograma). Imaginea **pout.tif** este o imagine de contrast oarecum scăzut. Pentru a vedea distribuția intensităților din imagine, creați o histogramă apelând funcția **imhist**. (Înaintea apelului **imhist**, adăugați comanda **figure**, astfel încât histograma să nu suprascrie afișarea imaginii **I** în fereastra figurii curente.) Observați cum histograma indică faptul că intervalul de intensitate al imaginii este destul de restrâns. Intervalul nu acoperă intervalul potențial de valori [0, 255], lipsind valorile mari si mici (care ar avea ca rezultat un contrast bun).

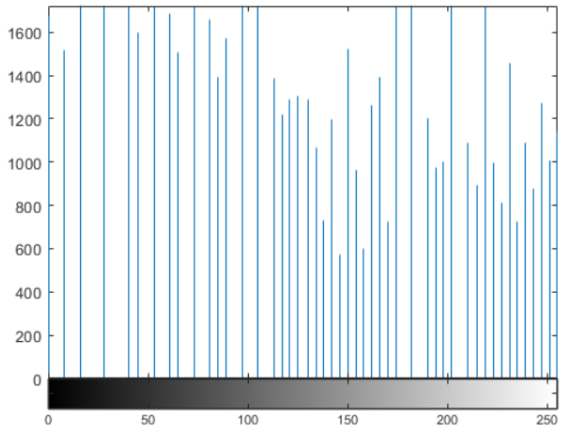
```
figure
imhist(I)
```



Îmbunătățiți contrastul imaginii folosind funcția **histeq** și apoi afișati imaginea. Egalizarea histogramei împarte valorile de intensitate pe întreaga gamă a imaginii. (*Toolboxul include multe alte funcții care efectuează ajustarea contrastului, inclusiv **imadjust** și **adapthisteq** plus alte instrumente interactive, cum ar fi instrumentul **Adjust contrast**, disponibil în **Image Viewer**.*)

```
I2 = histeq(I);
figure
imshow(I2)
```

Apelați din nou funcția **imhist** pentru a crea o histogramă a imaginii egalizate **I2**. Dacă comparați cele două histograme, puteți vedea că histograma I2 este răspândită pe întreaga gamă, comparativ cu histograma lui **I**.



Pasul 4: Salvați imaginea modificată pe disc

Scrieți imaginea **I2** într-un fișier pe disc, folosind funcția **imwrite**. Acest exemplu include extensia numelui de fișier '**.png**', astfel funcția **imwrite** scrie imaginea într-un fișier în format PNG (Portable Network Graphics), dar puteți specifica și alte formate valide.

```
imwrite (I2, 'pout2.png');
```

Pasul 5: Verificați conținutul imaginii salvate

Vizualizați ce a scris funcția **imwrite** în noul fișier salvat pe disc, folosind funcția **imfinfo**. Funcția **imfinfo** returnează informații despre imaginea din fișier, cum ar fi formatul, dimensiunea, lățimea și înălțimea.

```
imfinfo ( 'pout2.png');
```

Exercițiul 2: “Deblurul” unei imagini folosind un filtru Wiener

Acest exercițiu arată cum se utilizează deconvoluția Wiener pentru a deblura imaginile. Deconvoluția Wiener poate fi utilizată eficient atunci când caracteristicile de frecvență ale imaginii și zgomotul aditiv sunt cunoscute, cel puțin într-o anumită măsură.

Cerințe pentru Exercițiul 2:

1. Implementați în Matlab exercițiul 2, folosind imaginea inclusă în toolboxul de Image Processing (cea din exemplu). *Atașați codul final și rezultatele obținute la fiecare pas*
2. Folosind o altă imagine de dimensiune redusă (o puteți reduce folosind tooluri de Windows ca “Paint” sau “Snipping Tool”), rulați exemplul 2 cu această nouă imagine. *Atașați rezultatele obținute (asemeni cerinței 1, dar fără codul matlab)*
    a. Dacă imaginea este color, atunci folositi convertorul color → alb-negru implementat la cerința 3, Exercitiul 1.
3. Utilizați alți parametri pentru funcția PSF, pentru a simula blurul de mișcare (vezi Pasul 2). Rulați codul și faceți eventuale ajustări pentru a obține rezultate mai bune. *Atașați codul nou rezultat, rezultatele obținute și comentați eventualele modificări necesare.*

OBS: pentru imaginile personale, puteți folosi calea absolută spre imagine, dacă aceasta nu se află în workspace. Ex: *C:\Users\.....\block_diagram.png*

IMPLEMENTARE Exercițiu 2:

Pasul 1: Citiți și afișați o imagine neafectată de zgomot sau blur

Citiți și afișați o imagine “curată”, care nu are neclaritate sau zgomot.

```
Ioriginal = imread('cameraman.tif');
imshow(Ioriginal)
title('Imagine originala')
```

Pasul 2: Simulați blur de mișcare fără zgomot și apoi filtrați rezultatul

Simulați o imagine încețoșată (blur de mișcare) care ar putea rezulta din miscarea camerei. Mai întâi, creați o funcție de răspândire a punctelor, PSF, folosind funcția **fspecial** și specificând o mișcare liniară pe 21 de pixeli într-un unghi de 11 grade. Apoi, realizați convoluția funcției PSF cu imaginea, utilizând **imfilter**.

Imaginea originală are tipul de date uint8. Dacă dați o imagine uint8 ca intrare la **imfilter**, atunci funcția va cuantiza ieșirea pentru a returna o altă imagine uint8. Pentru a reduce erorile de cuantizare, convertiți imaginea la *double* înainte de a apela imfilter.

```
PSF = fspecial('motion',21,11);
Idouble = im2double(Ioriginal);
blurred = imfilter(Idouble,PSF,'conv','circular');
imshow(blurred)
title('Imagine incetosata (cu motion blur)')
```

Restaurați imaginea blurată folosind funcția **deconvwnr**. Imaginea blurată nu are zgomot, astfel încât puteți omite argumentul de intrare *noise-to-signal* (NSR).

```
wnr1 = deconvwnr(blurred,PSF);
imshow(wnr1)
title('magine cu blur restaurata')
```

Pasul 3: Simulați blur de mișcare cu zgomot gaussian și apoi filtrați rezultatul

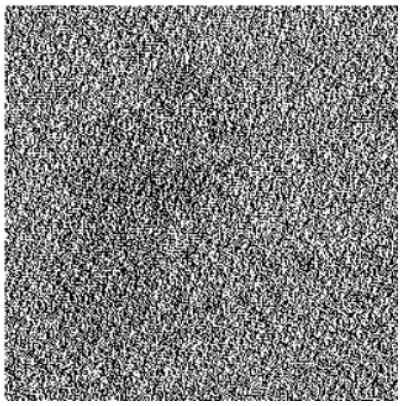
Adăugați zgomot gaussian cu valoare zero la imaginea neclară folosind funcția **imnoise**.

```
noise_mean = 0;
noise_var = 0.0001;
blurred_noisy = imnoise(blurred,'gaussian',noise_mean,noise_var);
imshow(blurred_noisy)
title('Imagine cu blur si zgomot')
```

Încercați să restaurați imaginea zgomotoasă și cu blur folosind **deconvwnr**, fără a oferi o estimare de zgomot. În mod implicit, filtrul de restaurare Wiener presupune ca NSR este egal cu 0. În acest caz, filtrul de restaurare Wiener este echivalent cu un filtru invers ideal, care poate fi extrem de sensibil la zgomotul prezent în imaginea de intrare.

În acest exemplu, zgomotul din această restaurare este amplificat într-un asemenea grad încât conținutul de imagine este pierdut.

```
wnr2 = deconvwnr(blurred_noisy,PSF);
imshow(wnr2)
title('Imagine cu blur si zgomot restaurata (NSR = 0)')
```



Încercați să restaurați imaginea zgomotoasă și cu blur folosind **deconvwnr** cu o valoare mai realistă a zgomotului estimat (NSR).

```
signal_var = var(Idouble(:));
NSR = noise_var / signal_var;
wnr3 = deconvwnr(blurred_noisy,PSF,NSR);
imshow(wnr3)
title('Imagine cu blur si zgomot restaurata (NSR estimat)')
```

Pasul 4: Simulați blur de mișcare și filtrați rezultatul ținând cont de eroarea de cuantizare pe 8 biți

Chiar și o cantitate de zgomot imperceptibilă din punct de vedere vizual poate afecta rezultatul. O sursă de zgomot în imagini este reprezentată de erorile de cuantizare provenite din lucrul cu imagini în reprezentarea *uint8*. Mai devreme, pentru a evita erorile de cuantizare, acest exercițiu a simulat o imagine încetoșată dintr-o imagine "curată" folosind tipul de date *double*. Acum, pentru a explora impactul erorilor de cuantizare asupra restaurării, simulați o imagine cu blur din imaginea curată, folosind tipul original de date *uint8*.

```
blurred_quantized = imfilter(Ioriginal,PSF,'conv','circular');
imshow(blurred_quantized)
title('Imagine cuantizata si cu blur')
```

Încercați să restaurați imaginea cuantificată și cu blur folosind **deconvwnr** fără a oferi o estimare a zgomotului. Chiar dacă nu s-a adaugat zgomot suplimentar, această restaurare este degradată în comparație cu restaurarea imaginii începțoșate în tipul de date *double*.

```
wnr4 = deconvwnr(blurred_quantized,PSF);
imshow(wnr4)
title('Imagine cuantizata si cu blur, restaurata (NSR = 0)');
```

Încercați să restaurați imaginea cuantificată estompată folosind **deconvwnr** cu o valoare mai realistă a zgomotului estimat.

```
uniform_quantization_var = (1/256)^2 / 12;
signal_var = var(Idouble(:));
NSR = uniform_quantization_var / signal_var;
wnr5 = deconvwnr(blurred_quantized,PSF,NSR);
imshow(wnr5)
title(' Imagine cuantizata si cu blur, restaurata (NSR estimat)');
```