Rotirea imaginilor cu orice unghi

Dezvoltator: Risco Bogdan (e-mail: risco.bogdan98@gmail.com)

Intrumator: Florea Camelia

Data predarii proiectului (zi/luna/an): 12/01/2021

CUPRINS

1. Sinteza Lucrarii

Proiectul realizat pe parcursul semestrului urmareste conceptul de **rotire a imaginilor cu orice unghi**. Problematica este importanta intrucat in acest fel se poate privi o imagine din mai multe perspective, oferind astfel intelesuri multiple si diverse unghiuri de abordare. Totodata, rotirea este esentiala in momentul in care se doresc **corectii ale fotografiilor** care au un **unghi** inestetic sau se doreste distorsionarea realitatii prezentata in imagine. Tehnica este importanta in mediul digital deoarece ofera sansa creatorilor de continut sa genereze postari originale, creative si inedite pentru un anumit public tinta (de exemplu in imaginile de prezentare pentru un produs de pe site-urile de e-commerce).

Pentru implementarea proiectului am folosit limbajul de programare **Python** prin intermediul programului **Spyder 3** (Anaconda). In urma acestor algoritmi am obtinut imagini rotite cu un unghi selectat, atat negativ, cat si pozitiv. Mai mult decat atat, am avut la dispozitie o histograma pentru fiecare imagine rotita si originala, cu scopul de a corecta in detaliu fotografia.

Avantajele principale ale acestei implementari sunt urmatoarele: posibilitatea de a corecta perspectiva dintr-o imagine, alegerea unui unghi dorit, vizualizarea histogramelor si corectarea erorilor aduse de aparatul foto. In ceea ce priveste **limitarile**, in urma rotirii imaginilor se pierd din **pixeli**, deci din informatia digitala de la nivelul fotografiei si apar **punctele negre**.

Datele de intrare ale proiectului sunt urmatoarele: imaginea pe nivele de gri si parametrii deformarii. **Datele de iesire** sunt: imaginea originala si histograma ei plus imaginea rotita impreuna cu histograma asociata acesteia.

2. Introducere

Intrucat mediul digital este dominat de elemente vizuale, respectiv imagini, este important ca acestea sa prezinte un nivel superior de calitate, claritate si coerenta. Din acest motiv, rotirea imaginilor cu un anumit unghi joaca un rol esential in diferite industrii si domenii de activitate pentru a corecta problemele ce tin de aparatul foto si de incadratura.

Algoritmul implementat presupune ca fiecare pixel are o pereche de coordonate (x, y) care descrie pozitia sa pe doua axe ortogonale din originea definita 0. In jurul acestei origini se roteste imaginea. Ceea ce am facut a fost sa luam valorile RGB la fiecare (x, y) locatie si sa rotim, dupa cum este necesar si sa scriem aceste valori in noua locatie. Aceasta noua locatie este obtinuta utilizand matricea de transformare.

Rezultatele sunt cele asteptate, calitatea imaginii s-a imbunatatit in mod vizibil, insa in unele cazuri am semnalat aparitia unor pixeli – puncte negre – ca urmare a modificarii unghiului cu diferite valori.

3. Fundamentare teoretica

3.1 Schema Bloc/Diagrama

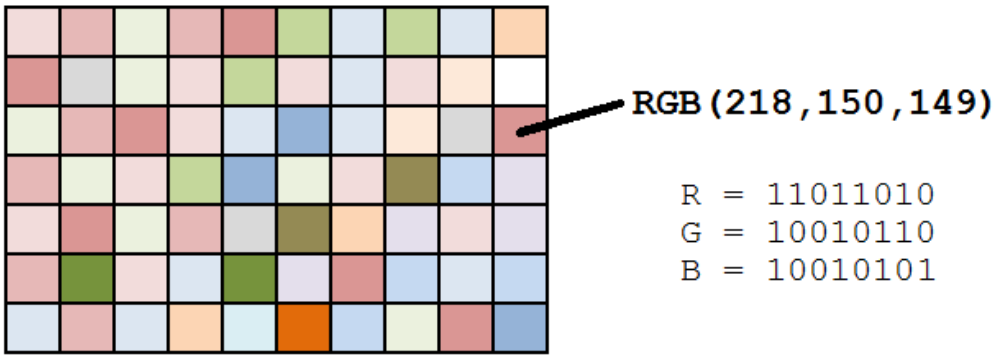

Fig1. Imagine RGB

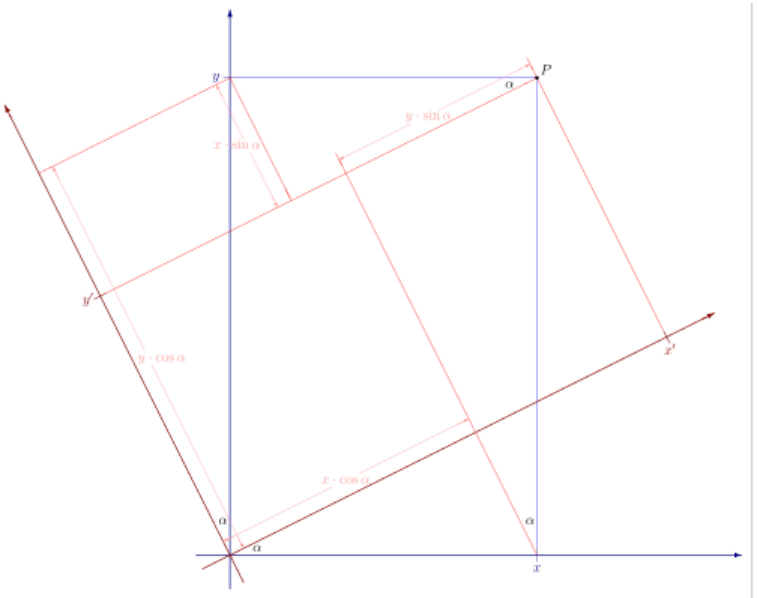

Fig2. Graficul pentru rotatie

3.2 Implementarea Agloritmului

Fiecare pixel are o pereche de coordonate *(x, y)* care descrie poziția sa pe două axe ortogonale din originea definită 0.

În jurul acestei origini se roteste imaginea. Ceea ce trebuie să facem este să luăm valorile RGB la fiecare *(x , y)* locație, rotim după cum este necesar, apoi scriem aceste valori în noua locație, luând în considerare *(x, y) în raport cu originea pe care am presupus-o. Noua locație este obținută* ***utilizând matricea de transformare.***

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

Fig3. Matricea de transformare

Când imaginea va fi rotită, dimensiunea cadrului care o conține se va schimba astfel încât să găsim noile dimensiuni folosim această formula :

new_width=|old_width×cosθ|+|old_height×sinθ|

new_height=|old_height×cosθ|+ |old_row×sinθ|

Formule matematice 2D

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

Matrice pentru rotație.

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta. \end{aligned}$$

Coordonatele punctului după rotație sunt x', y' și formulele pentru x' și y'.

$\begin{bmatrix} x \\ y \end{bmatrix}$ și $\begin{bmatrix} x' \\ y' \end{bmatrix}$

Vectorii au aceeași magnitudine și sunt separați cu un unghi θ.

$$z = x + iy$$

Punctul (x , y) din plan este reprezentat de numărul complex z.

$$\begin{aligned} e^{i\theta} z &= (\cos\theta + i\sin\theta)(x + iy) \\ &= x\cos\theta + iy\cos\theta + ix\sin\theta - y\sin\theta \\ &= (x\cos\theta - y\sin\theta) + i(x\sin\theta + y\cos\theta) \\ &= x' + iy', \end{aligned}$$

Acesta poate fi rotit printr-un unghi θ prin multiplicarea lui cu $e^{i\theta}$ Și extinderea produsului folosind formula lui Euler

$$e^{ix} = \cos x + i\sin x,$$

Formula lui Euler

Ecuatia pentru eliminarea punctelor negre, prin impartirea pe 3 etape si taierea lor.

$$\begin{vmatrix} 1 & -\tan(\theta/2) \\ 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 \\ \sin(\theta) & 1 \end{vmatrix} \begin{vmatrix} 1 & -\tan(\theta/2) \\ 0 & 1 \end{vmatrix}$$

părților reale și imaginare dă același rezultat bidimensională.

4. Implementarea solutie adoptate

Spyder este un mediu stiintific de dezovoltare bazat pe Python, este un mediu de dezvoltare integrat (IDE), o multiplatforma open-source fiind scris exclusiv in Python. Este proiectat de oameni de stiinta si este destinat exclusiv oamenilor de stiinta, analistilor de date si a inginerilor. Acesta are ca avantaje posibilitatea de breackpoints pentru debugging, curata variabilele automat, prelucrare grafica folosind Matplotlib.

Librarii utilizate:

- **cv2**
- **math** pentru: math.cos, math.sin;
- **numpy** reprezinta libraria mare :
  - array : Converteste imaginea sub fora de array
  - zeros : fac matricea plina cu zero unde o sa pun noile dimensiuni
  - uint8 : este un data type de tip unsigned de 8 biti si contine toate numerele de la 0 la 255
- PIL import Image : - pentru a manipularea imaginii: Ex: Open, save image.

Pentru inceput se deschide o fereastra care ne permite selectarea imaginii dintr-un folder,

Ulterior imaginea este convertita pe nivele de gri.

```
import math
import cv2
import matplotlib.pyplot as plt
import numpy as np
from tkinter import filedialog

imgFileName =  filedialog.askopenfilename(initialdir = "D:\PNI\_TestImagesPNI_labs\ImgTstColor\BerkeleyDataSet/",
                                          title = "Select file",
                                          filetypes = (("jpeg files","*.jpg"),("all files","*.*")))

print ("Fisierul selectat este: \n", imgFileName)
image = cv2.imread(imgFileName, 0)

print(image)

fig = plt.figure(figsize=(10,10))
plt.imshow(image, cmap = 'gray')
plt.show()
```

Fig1. Prelucrarea imaginii pe nivele de gri

Pentru rotirea imaginii am aplicat urmatoarea secventa de cod

```
for i in range(height):
    for j in range(width):
        y=image.shape[0]-1-i-original_centre_height
        x=image.shape[1]-1-j-original_centre_width

        new_y,new_x= forfecare(angle,x,y)

        new_y=new_centre_height-new_y
        new_x=new_centre_width-new_x

        if 0 <= new_x < new_width and 0 <= new_y < new_height and new_x >= 0 and new_y >= 0:
           output[new_y,new_x]=image[i,j]
```

Fig2. Rotirea imaginii

Functie de taiere pe 3 etape

Formula:

$$\begin{vmatrix} 1 & -\tan(\theta/2) \\ 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 \\ \sin(\theta) & 1 \end{vmatrix} \begin{vmatrix} 1 & -\tan(\theta/2) \\ 0 & 1 \end{vmatrix}$$

```
58  def forfecare(angle,x,y): #shear function
59      #shear 1
60      tangent=math.tan(angle/2)
61      new_x=round(x-y*tangent)
62      new_y=y
63      #shear 2
64      new_y=round(new_x*math.sin(angle)+new_y)
65      #shear 3
66      new_x=round(new_x-new_y*tangent)
67      return new_y,new_x
```

5. Rezultate experimentale

Imagine 1.

- Unghi : -15

Fig1. Imaginea rotatia la -15

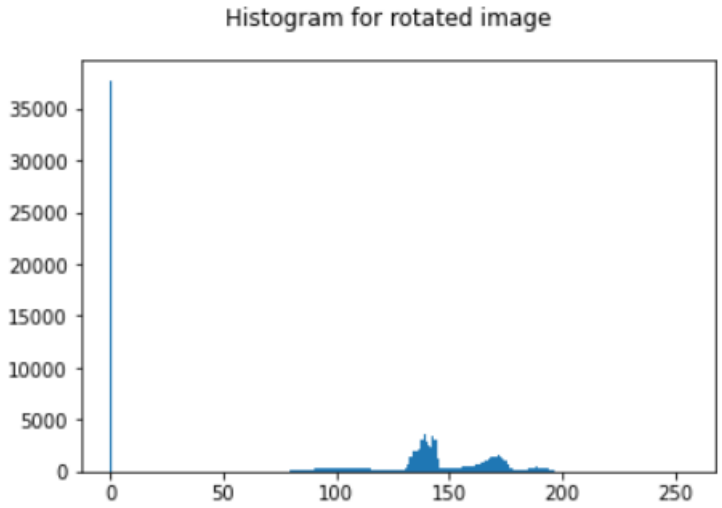

Fig2. Histograma la -15

- Unghi : 20

Fig3. Imaginea rotatia la 20

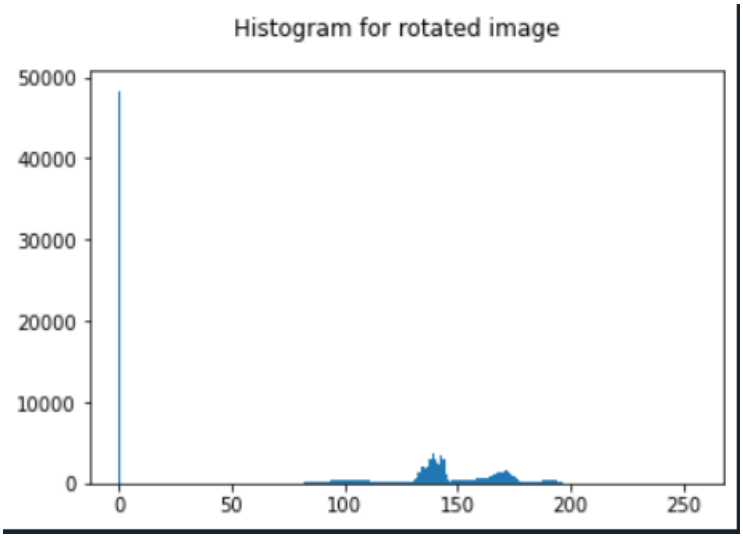

Fig4. Histograma la 20

- Unghi : -242

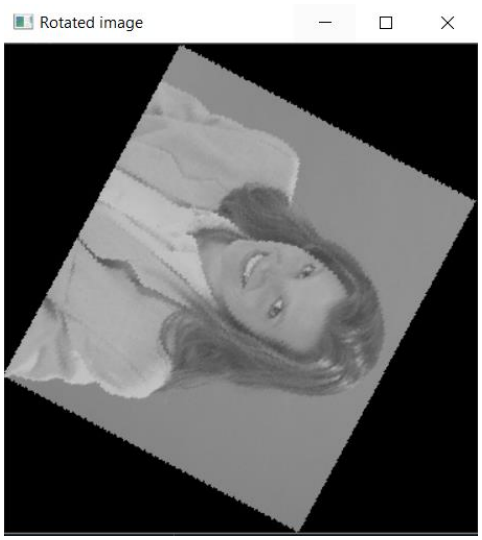

Fig5. Imaginea rotatia la -242

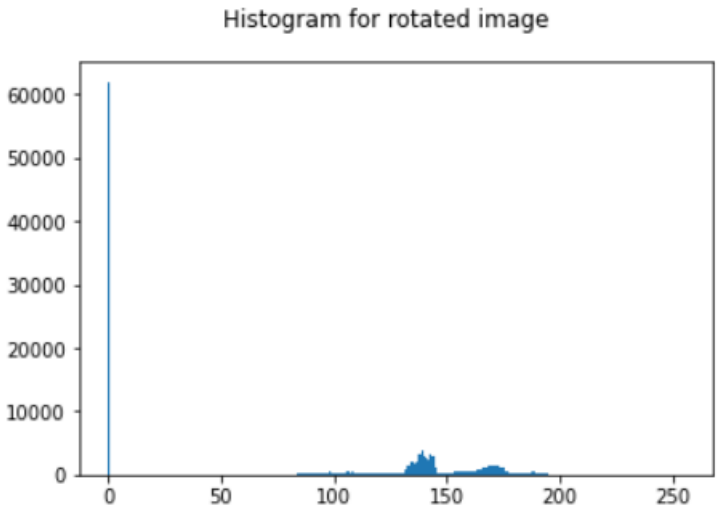

Fig6. Histograma la -242

Imagine 2.

- Unghi : 90

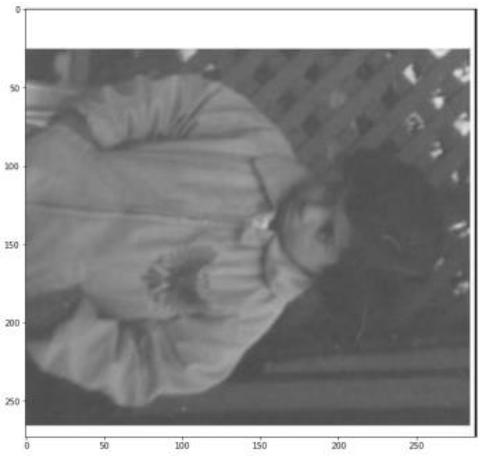

Fig7. Imaginea rotatia la 90

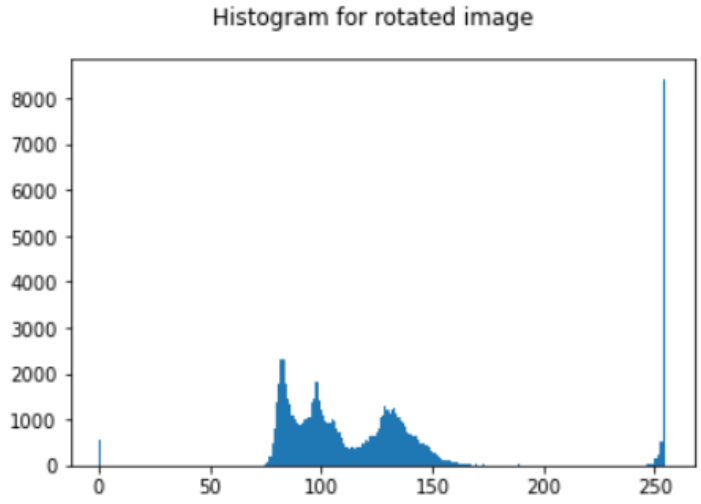

Fig8. Histograma la 90

Imagine 3.

- Unghi : 220

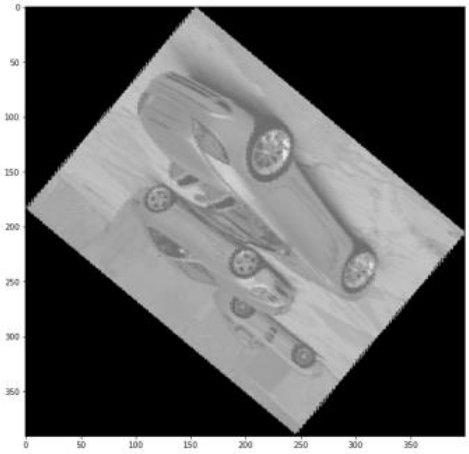

Fig9. Imaginea rotatia la 220

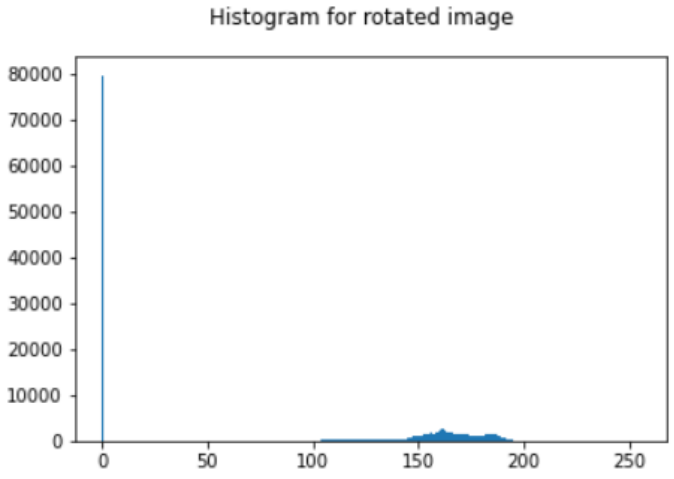

Fig10. Histograma la 220

Imagine 4.

- Unghi : -90

Fig11. Imaginea rotatia la -90

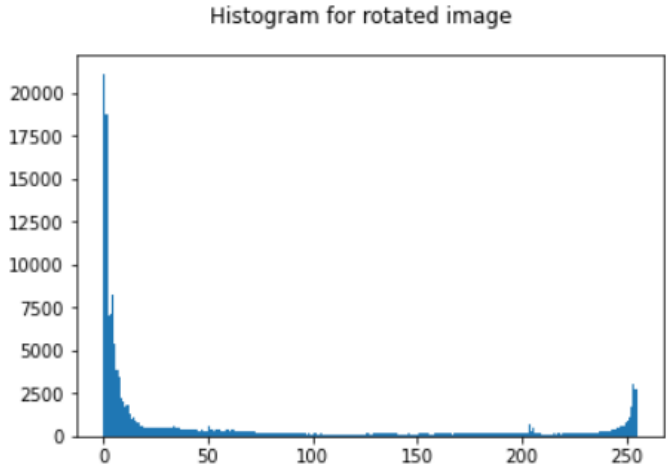

Fig12. Histograma la -90

Imagine 5.

- Unghi : 320

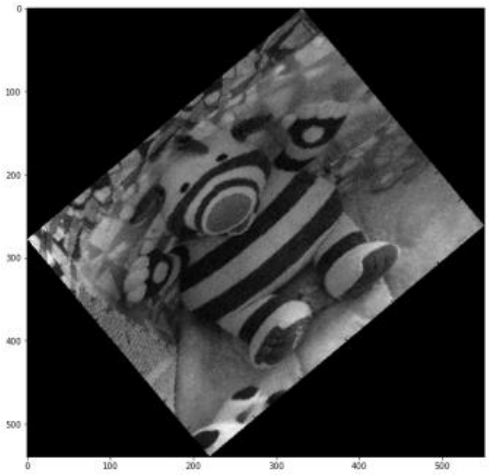

Fig13. Imaginea rotatia la 320

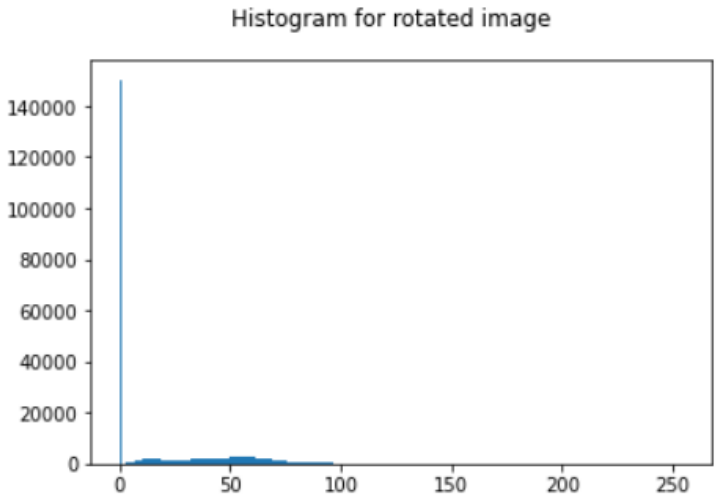

Fig14. Histograma la 320

6. Concluzie

In urma implementarii proiectului am ajuns la rezultatul dorit de a putea roti orice imagine cu un unghi ales, atat pozitiv cat si negativ intrucat algorimtul permite acest lucru. Pentru a obtine o calitate mai buna a imaginii rotite am implementat o functie prin taiere care imparte imaginea in alte 3 subimagini si le prelucreaza pentru obtinerea unui rezultat cat mai bun si eliminarea punctelor negre (pixelilor pierduti), insa rezultatul obtinut nu este perfect avand mici pierderi de pixeli la aumite unghiuri.

7. Bibliografie

1. https://en.wikipedia.org/wiki/Rotation_(mathematics)
2. William K. Pratt, Digital Image Processing, Wiley Fourth Edition (PDF)
3. https://homepages.inf.ed.ac.uk/rbf/HIPR2/rotate.htm

8. Anexe

```
import math
import cv2
import matplotlib.pyplot as plt
import numpy as np
from tkinter import filedialog

imgFileName =  filedialog.askopenfilename(initialdir =
"D:\PNI\_TestImagesPNI_labs\ImgTstColor\BerkeleyDataSet/",
                                          title = "Select file",
                                          filetypes = (("jpeg files","*.jpg"),("all files","*.*")))

print ("Fisierul selectat este: \n", imgFileName)
image = cv2.imread(imgFileName, 0)
#afisare matricii imaginii originale
print(image)

fig = plt.figure(figsize=(10,10))
plt.imshow(image, cmap = 'gray')
plt.show()

#shape-ul
height,width = image.shape
print(height,width)
#afisare histogrammei imaginii originale
plt.figure()
```

```
plt.hist(image.ravel(), 256, [0,255])
plt.suptitle('Histogram')
plt.show()

#shape-ul
imgGray = image.shape;
print(imgGray)



angle = input("Enter the Angle: ") # Introduc unghiul dorit
angle = int(angle) # Convertesc in numar real
angle = math.radians(angle) #Convertesc Grad-ul introdus in radiani



cos=math.cos(angle) # obtin cosinului radianului
sin=math.sin(angle) # obtin sinusul radianului

new_height = round(abs(image.shape[0]*cos)+abs(image.shape[1]*sin))+1
new_width = round(abs(image.shape[1]*cos)+abs(image.shape[0]*sin))+1

#Fac o matrice de zero unde o sa salvez imaginea rotita
output= np.zeros((new_height, new_width))
#image_copy=output.copy()

# Centrul imagini originale
```

```
original_centre_height = round(((image.shape[0]+1)/2)-1)
original_centre_width = round(((image.shape[1]+1)/2)-1)

# centrul noii imagini
new_centre_height= round(((new_height+1)/2)-1)
new_centre_width= round(((new_width+1)/2)-1)

def forfecare(angle,x,y): #shear function
    #shear 1
    tangent=math.tan(angle/2)
    new_x=round(x-y*tangent)
    new_y=y
    #shear 2
    new_y=round(new_x*math.sin(angle)+new_y)
    #shear 3
    new_x=round(new_x-new_y*tangent)
    return new_y,new_x

#Fac 2 for-uri, cu primul iterez elementele din height iar cu al doilea elementele din width
#Obtin cu y si x coordonatele din centrul imaginii originale
#Cu noile coordonate x si y obtin alte coordonate new_y si new_x unde pun valoarea rotunjita a pixelilor prin formula
for i in range(height):
    for j in range(width):
        y=image.shape[0]-1-i-original_centre_height
        x=image.shape[1]-1-j-original_centre_width
```

```
        new_y,new_x= forfecare(angle,x,y)


        new_y=new_centre_height-new_y
        new_x=new_centre_width-new_x
        #verific cu if pentru a nu avea erori pe perioada rotirii
        if 0 <= new_x < new_width and 0 <= new_y < new_height and new_x >= 0 and new_y >= 0:
            output[new_y,new_x]=image[i,j]  #imaginea rotita se salveaza in matricea de zero


cv2.destroyAllWindows() # se inchide fereastra externa cu imaginea afisata
imgSaveFileName = filedialog.asksaveasfile(title = 'Save',
                                filetypes = [('All Files', '*.*'), ('JPG Files', '*.jpg')] )

print(imgSaveFileName.name)
cv2.imwrite(imgSaveFileName.name, output)

fig = plt.figure(figsize=(10,10))
plt.imshow(output, cmap = 'gray')
plt.show()
#afisare histograma imaginii rotite
plt.figure()
plt.hist(output.ravel(), 256, [0,255])
plt.suptitle('Histogram for rotated image')
plt.show()

#salvez imaginea rotita intr-o imagine de test
path = r'C:\Users\risco\OneDrive\Desktop\Materii_UTCN\PNI_Proiect\test.jpg'
```

```
print("Image Saved!")
image = cv2.imread(path)

window_name = 'image'
cv2.imshow(window_name, image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```