

Human Vision:

Imagini din mediul inconjurator => Capturarea imaginii de catre ochi => Creierul analizeaza imaginea => Output

Computer Vision:

Imagini de input => Identificarea imaginii (citire, procesare) => Analizarea imaginii => Output

Imaginile sunt formate din pixeli ('picture elements').

Pixelul este cea mai mică unitate de masura a unei imagini digitale care poate fi afisata si reprezentată pe un dispozitiv digital (ecran).



Imaginile pot avea moduri diferite de reprezentare. In modul RGB, cel mai des intalnit, fiecare pixel este codificat prin 3 culori: rosu, (Red), verde (Green), albastru (Blue), fiecare culoarea avand o valoare din intervalul [0, 255]

1. **Hue (H):** Represents the color type and is measured in degrees from 0 to 360. For example, red is at 0 degrees, green at 120 degrees, and blue at 240 degrees
2. **Saturation (S):** Indicates the intensity or purity of the color, ranging from 0% (gray) to 100% (full color). Higher saturation means more vivid colors
3. **Value (V):** Represents the brightness of the color, ranging from 0% (black) to 100% (full brightness). Higher value means a brighter color

Dataset de lucru: [Butterfly Image Classification](#), [MNIST Dataset](#) (pentru colorare)

Citirea / Scrierea / Afisarea imaginilor:

1. OpenCV

Citirea unei imagini color

```
image = cv.imread('image name.png')
```

Variabila image este de tipul numpy-array uint8. Dimensiunea variabilei este de: H, W, 3, unde H este inaltimea imaginii, W este latimea, iar 3 este numarul de canale RGB.

Citirea unei imagini in tonuri de gri

```
image = cv.imread('image name.png', cv.IMREAD_GRAYSCALE)
```

Dimensiunea variabilei este de: H, W.

Afisarea unei imagini

```
cv.imshow('windowName', image)
```

```
cv.waitKey(0) % se asteapta apasarea unei taste
```

```
cv.destroyAllWindows() % se inchide fereastra
```

Operatii cu o matrice de intensitati

Fie img un numpy-array 100×100 de tipul uint8 obtinuta astfel:

```
img = cv.resize(cv.cvtColor(cv.imread(image_name.jpg), cv.COLOR_BGR2GRAY),(100, 100))
```

2. PIL

Citirea unei imagini

```
im = Image.open(image_name.png)
```

Afisarea dimensiunii

```
im.size
```

Codificarea unei imagini

```
im.mode
```

Afisare imagine si plotare

```
plt.imshow(im)
```

```
plt.show()
```

Cum se poate face conversia de la RGB la GRAY? (fara a utiliza functii specifice)

- + implementarea propriei functii de conversie folosind numpy si broadcasting folosind formula ([RGB to Grayscale Conversion Calculator | Good Calculators](#))

```
# gray_image = np.zeros((image_bgr.shape[0], image_bgr.shape[1]), dtype=np.uint8)
```

```
for i in range(image_bgr.shape[0]):
```

```
    for j in range(image_bgr.shape[1]):
```

```
        b, g, r = image_bgr[i, j] # Get B, G, R values
```

```
gray_value = int(0.299 * r + 0.587 * g + 0.114 * b)
gray_image[i, j] = gray_value
```

Considerand o imagine alb-negru, cum am putea sa o coloram? (din mnist)

Conversie RGB -> GRAY cu ajutorul functiilor:

- `im.covert()`
- `cv.COLOR_BGR2GRAY`

OpenCV foloeste formatul BGR, deci daca se citeste o imagine cu `cv2.imread()` o interpreteaza default ca BGR

Imaginile pot fi si binarizate (transformate in imagini binare).

- se alege un prag, toate valorile care se afla sub prag sunt setate ca negru, iar cele care se afla peste prag sunt setate ca alb.

Sortarea elementelor/intensitatilor din matricea imaginii (`np.sort`), punand elementele sortate intr-un vector `x` de dimensiuni `HxWx 1`

- + `plotarea` valorilor lui `x`

Afisare submatrice care corespunde sfertului matricei `img` din partea dreapta-jos.

Transformare imagine: fiecare pixel din `C` are intensitatea egala cu pixelul corespunzator din `img` din care se scade intensitatea medie a imaginii; pixelii cu intensitatea < 0 vor fi setati ca avand intensitatea $= 0$.

Filtru pentru o imagine: eliminarea zgomotului dintr-o imagine sau imbunatatirea unor caracteristici

ai.stanford.edu/~syyeung/cvweb/tutorial1.html

[Image Filters in Python. I am currently working on a computer... | by Manvir Sekhon | Towards Data Science](#)

1. Filtru median

- utilizat pentru reducerea zgomotului din imagini, in special a zgomotului de tip salt-and-pepper, care apare ca pixeli albi si negri distribuiti aleatoriu in imagine.
- implementare: inlocuirea fiecarui pixel cu valoarea mediana a pixelilor dintr-o vecinatate data, de obicei o matrice patratică 3×3 sau 5×5 . Pentru fiecare pixel din imagine, selectam vecinii acestuia intr-o matrice patratica, sortam valorile de intensitate si selectam valoarea mediana pentru a o atribui pixelului central.

```

import numpy as np

def medianBlur(image, kernel_size=3):
    # Obținem dimensiunea imaginii și pregătim un tampon de ieșire
    height, width = image.shape
    output_image = np.zeros((height, width), dtype=image.dtype)

    # Calculăm marginile kernelului
    pad = kernel_size // 2

    # Aplicați filtrul median
    for i in range(pad, height - pad):
        for j in range(pad, width - pad):
            # Extragem fereastra curentă
            window = image[i - pad:i + pad + 1, j - pad:j + pad + 1]
            # Calculăm valoarea mediană și o setăm în pixelul de ieșire
            median_value = np.median(window)
            output_image[i, j] = median_value

    return output_image

# Exemplu de utilizare
import cv2
image = cv2.imread("image.jpg", cv2.IMREAD_GRAYSCALE)
output_median = medianBlur(image, kernel_size=3)
cv2.imwrite("image_median_custom.jpg", output_median)

```

2. Filtru gaussian

- folosit pentru netezirea imaginilor si reducerea zgomotului de inalta frecventa => un efect de estompare (blur)
- functia Gaussiana genereaza un kernel (sau masca) Gaussian

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

(x,y) reprezintă coordonatele din vecinătatea unui pixel (centrul kernelului este la (0,0),

- **Deviația standard (σ):** daca aceasta este mica, curba este ingusta, doar pixelii foarte apropiati de centru vor avea un impact semnificativ asupra valorii pixelului central. Daca in schimb sigma este mare, curba este mai lata, iar un numar mai mare de pixeli vor contribui la estomparea unui pixel, generand un efect de estompare mai puternic.
- implementare: construirea kernelului gaussian (kernelul Gaussian este o matrice n×n, de obicei 3×3, 5×5 sau 7×7). Pentru fiecare element al acestei matrice, folosim functia Gaussiana pentru a calcula o pondere bazata pe

distanța față de centrul kernelului.), după generarea kernelului, se vor normaliza valorile a.i suma lor să fie 1, intensitatea imaginii nefiind modificată.

```
import numpy as np
```

```
def gaussian_kernel(kernel_size=3, sigma=1.0):
    # Calculăm centrul kernelului
    kernel = np.zeros((kernel_size, kernel_size), dtype=np.float32)
    center = kernel_size // 2

    # Generăm kernelul Gaussian
    for i in range(kernel_size):
        for j in range(kernel_size):
            x = i - center
            y = j - center
            kernel[i, j] = np.exp(-(x**2 + y**2) / (2 * sigma**2))
    kernel /= (2 * np.pi * sigma**2)
    return kernel / np.sum(kernel) # Normalizăm kernelul

def gaussianBlur(image, kernel_size=3, sigma=1.0):
    # Obținem dimensiunea imaginii și pregătim un tampon de ieșire
    height, width = image.shape
    output_image = np.zeros((height, width), dtype=np.float32)

    # Obținem kernelul Gaussian
    kernel = gaussian_kernel(kernel_size, sigma)

    # Calculăm marginile kernelului
    pad = kernel_size // 2

    # Aplicați filtrul Gaussian
    for i in range(pad, height - pad):
        for j in range(pad, width - pad):
            # Extragem fereastra curentă
            window = image[i - pad:i + pad + 1, j - pad:j + pad + 1]
            # Aplicăm convoluția
            value = np.sum(window * kernel)
            output_image[i, j] = value

    return output_image.astype(image.dtype)

# Exemplu de utilizare
output_gaussian = gaussianBlur(image, kernel_size=5, sigma=1.5)
cv2.imwrite("image_gaussian_custom.jpg", output_gaussian)
```

-

- set de imagini (de date) ce contine imagini continand caracteristici comune
- aceasta colectie de imagini similare poate fi descrisa prin imaginea ei medie

Pentru colectia de imagini data, putem efectua:

- calculul imaginii medie color a colectiei (imagini color)
- calculul imaginii medie de intensitate a colectiei (imagini gray transformate)
- calculul matricii X, fiecare element $X[i,j]$ al matricei reprezinta deviatia standard

a intensitatilor pixelilor (i,j) din imaginile de intensitate corespunzatoare imaginilor din colectie

- normalizarea unor imagini: care ar putea fi metodele? (impartire la 255, utilizarea mediei si deviatiei standard, min-max scaling)
 - + ce sunt media si deviatia? care este relatia dintre ele? cum le putem calcula?

$$s = \sqrt{\frac{\sum (x - \bar{x})^2}{n - 1}}$$

- afisa cele trei imagini (color, gray, X)