**Cursul 1**

|  |  |
| --- | --- |
| **Introducere în prelucrarea imaginilor**  **De ce limbajul Python?**  **Elemente de baza ale limbajului Python**  **Tipuri de date Python**  **Instrucțiunea condițională if**  **Instrucțiunea repetitivă for**  **Funcții**  **Lucru cu imagini**  **Modul de reprezentare al imaginilor** | **Mediul de programare PyCharm**  **Bibiloteca Matplotlib**  **Biblioteca OpenCV**  **Citirea și afișarea imaginilor**  **Interpolarea imaginilor**  **Aplicații**  **Întrebări**  **Webografie** |

**O imagine valorează cât o mie de cuvinte (San Antonio, Lumina Pictorial Revista de Război)**

**Funcţiile imaginii**

* **Documentară - imaginea este mai concretă şi mai reală decât proiecţia unei discuţii**
* **Socială-permite exprimarea unor situaţii şi opinii în comunitate**
* **Educativa - dezvoltă aptitudini , competenţe, simţul proporţiilor şi al armoniei**
* **De comunicare - transmite opinii despre viaţă sau diferite evenimente**
* **Descoperă următoarele funcții în acest curs…**

**Introducere în prelucrarea imaginilor**

Prelucrarea imaginilor reprezintă o multitudine de metode de transformare și interpretare a informației vizuale, reprezentate în formă digitală.

Procesarea digitală a imaginii folosește algoritmi care, prin prelucrarea pixelilor, sunt capabili să modifice o imagine digitală. Forma matematică a unei imagini este matricea.[[1]](#footnote-1)

Domenii de cercetare

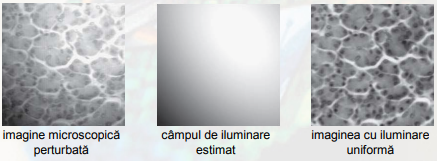
* Detecția de trăsături (Feature Detection)
* Compresia (Compression)
* Analiza imaginilor de profunzime (Range image analysis)
* Modelarea şi reprezentarea formelor (Shape modeling and representation)
* Stereo viziunea (Stereo vision)
* Viziunea color (Color vision)
* Analiza mișcării (Motion analysis)
* Extragerea trăsăturilor invariante (Invariant feature extraction)
* Detecția obiectelor (Object detection)
* Urmărirea obiectelor (Object tracking)
* Recunoașterea obiectelor 3D (3D object recognition)[[2]](#footnote-2)

Unde își găsește aplicații PI ? (TAPAI\_BIonescu\_M1.pdf)

* Aplicații de “consum” ;
* Imagistică satelitară ;
* Supraveghere video ;
* Automatizări industriale ;
* Interacție om-maşină ;
* Teledetecție (Remote sensing)
* Imagistica medicala (Medical Imaging)
* Evaluare nedistructivă (Non-Destructive evaluation)
* Studii criminalistice (Forensic Studies)
* Textile (Textiles)
* Știința materialelor(Material science)
* Militar (Military)
* Industrie cinematografică (Film Industry)
* Procesarea documentelor (Document Processing)
* Arte grafice (Graphic arts)
* Industria tipografiei (Printing industry)[[3]](#footnote-3)

Aplicații de consum (TAPAI\_BIonescu\_M1.pdf):

* iluminarea neuniformă a imaginii constă în fluctuații artificiale ale intensității anumitor zone din imagine ce duc la alterarea informației (ex. imagini medicale)[[4]](#footnote-4)



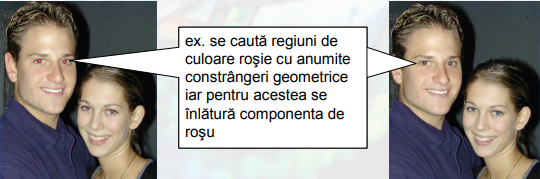
* defectele de contrast au ca rezultat eliminarea de informație utilă din imagine şi/sau alterarea acesteia.
* contrast = o măsură a raportului dintre intensitatea maximă şi cea minimală din imagine[[5]](#footnote-5).



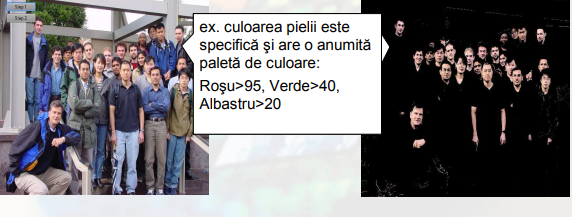
* o altă problemă frecventă, chiar inerentă sistemelor de achiziție de imagini o reprezintă prezența informațiilor parazite în imagine de tip zgomot și artefacte.[[6]](#footnote-6)



* lumina blitzului camerei se declanșează prea repede pentru ca pupila să se închidă, aceasta se reflectă de peretele ochiului și revine afară prin pupilă. Culoarea roșie apare datorită vaselor de sânge ale coroidei ce vascularizează peretele ochiului.



* se dorește localizarea în imagine a regiunilor ce corespund culorii pielii (umane). - prezență umană în scenă !?



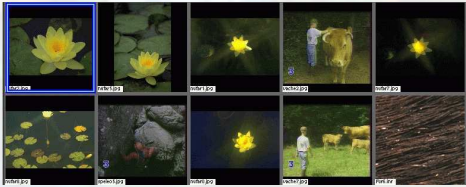
* algoritmi destinați aparatelor foto digitale şi camerelor video detecția culorii pielii (aplicații)



* gestiunea automată a colecțiilor multimedia CBIR = Content-Based Image Retrieval (număr foarte mare de imagini , cum găsim imaginea dorită , cum specificăm ce imagine dorim ?)



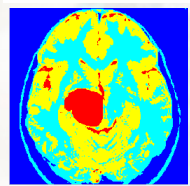
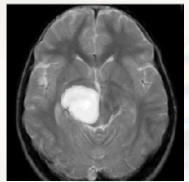
* informația textuală nu este suficientă pentru a exprima cererile de căutare (“query”) informații de conținut conversie automată de către CBIRS în atribute de conținut (adnotare): culoare, textură, formă, ...



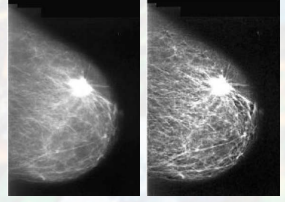
Imagistică medicală

Tipuri de imagini medicale

* radiografii ~ raze X
* MRI – Magnetic Resonance Imaging, folosește un câmp magnetic foarte puternic ce polarizează atomii de hidrogen din corp.
* microscopie radiografie MRI
* imagistică nucleară, ultrasunete, CT.[[7]](#footnote-7)

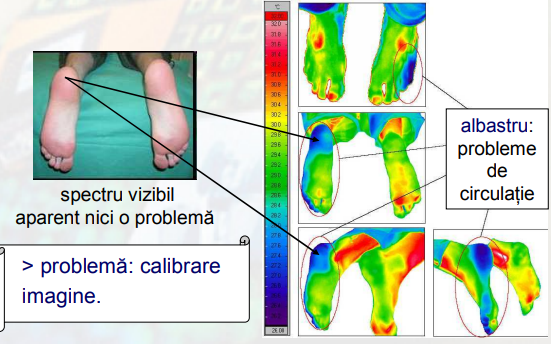


Detecție tumoare creier

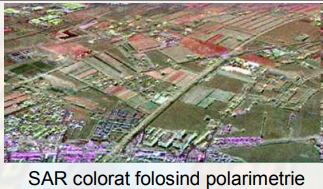


Detecție cancer mamar

* termografie: transformă radiația electromagnetică din spectrul infraroșu (~0.9–14 µm) în imagini



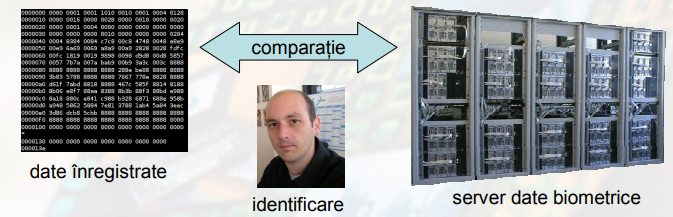
* Imagistică satelitară > “remote sensing” captarea de la distanță (“remotely”) a informațiilor relative la un anumit obiect sau fenomen (fără a interacționa fizic cu acesta). - dispozitivele de captură cele mai sofisticate: sateliții[[8]](#footnote-8)



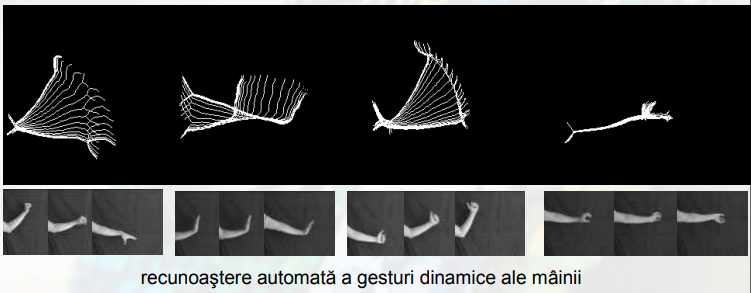
* Supraveghere video > detecția automată a evenimentelor de interes (continuare) securizare[[9]](#footnote-9)



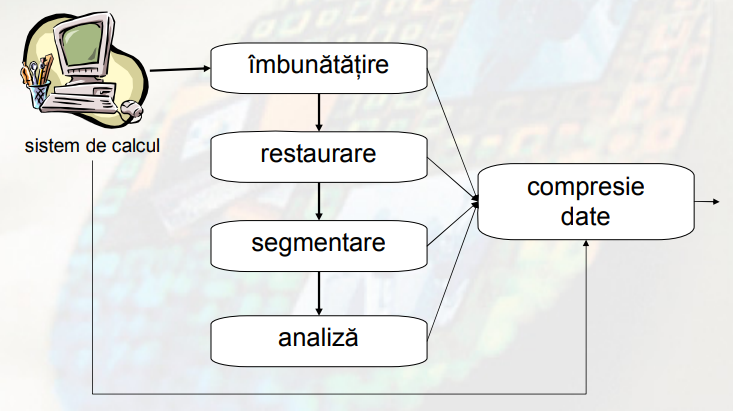
* Supraveghere video > indexare automată baze de înregistrări > sisteme biometrice de identificare (continuare) de comportament: scrisul de mână, voce, gesturi.



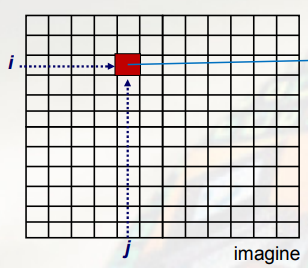
* Interacție om-maşină sisteme de analiză şi recunoaștere a gesturilor



Sistem de prelucrare a imaginilor la nivel logic (software) aceasta poate are următoarele etape[[10]](#footnote-10):



Modul de reprezentare al imaginilor



**imagine**a = o mulțime finită de pixeli[[11]](#footnote-11)

Elementul de bază al unei imagini 2D se numeşte pixel = picture element.

Fiecare pixel al imaginii este caracterizat prin:

- **valoare** - legată de “culoarea” transmisă (ex. alb, roşu, etc.)

- **poziție în imagine** -informație spațială.[[12]](#footnote-12)

**Forma pixelului**: pătrată, rotundă, rectangulară (grid) + dimensiunea pixelului fac legătura dintre imagine şi realitatea fizică.

**Rezoluția spațială** = numărul de pixeli independenți pe unitatea de lungime:[[13]](#footnote-13)

**-**dpi (dots per inch),

- ppi (pixels per inch),

- lpi (lines per inch).

Anumiți pixeli pot fi extrapolați artificial din pixeli reali înregistrați, pentru o rezoluție reală nu se ține cont de aceştia (camere Web ?).

- **dimensiunea unui pixel** = 1 / rezoluție spațială[[14]](#footnote-14)

ex. 300 dpi - 1 pixel = 1/300 inch = 3.3 m inch = 0.00838 cm

- valori uzuale: 150 dpi = 59,06 pixels/cm (~ 60)

300 dpi = 118,11 pixels/cm (~ 120) 600 dpi = 236,22 pixels/cm

(~ 240) 1200 dpi = 472,44 pixels/cm (~ 480)

**Cu cât mai mulți dpi cu atât mai clară este imaginea[[15]](#footnote-15)**

**Cum sunt reprezentate imaginile digitale ?**

Din punct de vedere al valorilor, imaginile digitale sunt de două feluri: imagini vectoriale = valoarea unui pixel este un vector - imagini color: un pixel este de regulă (cel mai frecvent) un triplet (R,G,B) = procent de roşu, de verde şi de albastru, unde fiecare valoare este pe 8 biți (0≤R≤255, etc.) [[16]](#footnote-16)



pixel = (84,187,254)

imagine color RGB (8+8+8 biți/pixel)

**Spațiu de culoare** = este un model matematic abstract ce descrie un mod de reprezentare al culorilor ca vectori de numere[[17]](#footnote-17)

(R,G,B) spațiul de culoare RGB

Din punct de vedere al valorilor, imaginile digitale sunt de două feluri:

**imagini vectoriale[[18]](#footnote-18)**

- imagini satelitare : un pixel poate fi un vector cu 3 – 200 componente. -

-imagini termografice : un pixel poate fi un vector cu 2 – 5 componente în banda IR (infra-roşu).

Din punct de vedere al semnificației valorilor, imaginile digitale sunt: **imagini de intensitate** = valorile pixelilor sunt direct proporționale cu mărimea fizică măsurată în scenă (culoare, intensitate luminoasă, temperatură, ...)

**De ce limbajul Python?**

* Python este folosit în automatizare;
* Este utilizat în Web Development;
* Sintaxa este simplă;
* Conține biblioteci pentru data science;
* Crește popularitatea printre developări;
* Crește cerințele de programare în Python in domeniul IT;
* Poți face parte dintr-o comunitate Pyhon;
* Testarea aplicațiilor create în Python (PyTest, Selenium, Faker, PyUnit, Unitest)[[19]](#footnote-19)

**Elemente de baza ale limbajului Python**

**Tipuri de date Python**

În Python se lucrează cu următoarele tipuri de date: numeric, logic (bool), secvențe (string-uri sau șiruri de caractere, liste, tupluri), tabele asociative (dictionary), mulțimi (set), clase, instanțe si excepții.

1. Tipul numeric este format din tipul întreg (int) real (float) și complex (a+bi, a partea reală, b este partea imaginara) compatibile cu operatorii:

a) Operatori aritmetici: +(adunare), - (scădere), \* (înmulțire), /(impărțire), //(împărțire întreagă), \*\*(ridicare la putere), % (modulo)

b) Operatori de comparație: ==,!=,<, > ,<=,>=

c) Operatori logici: and, or, not, is (întoarce true dacă ambii operanzi sunt referințe la același obiect), is not( întoarce true dacă operanzii sunt referințe la obiecte diferite); in (întoarce true dacă operandul 1 care este o valoare se află în operandul 2 care este o secvență) not in(Negarea operatorului anterior).

d) Operatori pe biți: & (AND Și logic bit cu bit ), | (OR Sau logic bit cu bit) , ^ (XOR Sau exclusiv bit cu bit),~ (NOT Inversare biți) << (zero fill left shift Shift stânga și completare cu 0 în dreapta), >> (Signed right shift Shift dreapta cu menținerea bitului de semn).

Exemplu:

**import** numpy **as** np  
x = 3  
print(type(x)) *# afiseaza "<class 'int'>"*print(x) *# afiseaza "3"*print(x + 1) *# adunare; afiseaza "4"*print(x - 1) *# scadere; afiseaza "2"*print(x \* 2) *# multiplica; afiseaza "6"*print(x \*\* 2) *# ridica la putere; afiseaza "9"*x += 1  
print(x) *# afișează "4"*x \*= 2  
print(x) *# afișează "8"*y = 2.5  
print(type(y)) *# afiseaza "<class 'float'>"*print(y, y + 1, y \* 2, y \*\* 2) *# afiseaza "2.5 3.5 5.0 6.25"*

q = 3 + 4j

r = 9 + 8j

print(q-r)

#6-4i

Observatie:

1.spre deosebire de alte limbaje Python-ul nu folosește operatorii de incrementare (++) respectiv decrementare (--).

2. Tipul de dată nu este specificat în prealabil

3.Pentru a crea o variabilă, folosim sintaxa: nume\_variabila = valoare.

Tipul de dată boolean este format din Valorile True și False și Operațiile and, or, not

Exemplu:

**import** numpy **as** np  
t = **True**f = **False**print(type(t)) *# afiseaza "<class 'bool'>"*print(t **and** f) *# operatorul logic AND; afiseaza "False"*print(t **or** f) *# operatorul logic OR; afiseaza "True"*print(**not** t) *# operatorul logic NOT; afiseaza "False"*print(t != f) *# operatorul logic XOR; afiseaza "True"*

**Tipuri de date standard**

**Secvențe:**

● Mulțimi finite și ordonate, indexate prin numere ne-negative.

● Dacă a este o secvență atunci:

○ len(a) returnează numărul de elemente;

○ a[0], a[1], …, a[len(a)-1] elementele lui a.

● Examples: [1, ‘a’]

**String-uri**:

● este o secvență imutabilă;

● caractere Unicode .

● Literali: ‘abc’, “abc”

**import** numpy **as** np  
buna = **'buna'** *# simboluri apostrof*ziua = **"ziua"** *# ghilimele.*print(buna) *# afișează "hello"*print(len(buna)) *# lungimea șirului; afișează "4"*hw = buna + **' '** + ziua *# concatenează șirurile*print(hw) *# afișează "buna ziua"*hw12 = **'%s %s %d'** % (buna, ziua, 12) *# sprintf formatează*print(hw12) *# afișează "buna ziua12"*

**Liste sunt** secvențe mutabile **echivalente cu tablourile putând conține date de mai multe tipuri**

• ex: [] sau [1, ‘a’, [1, 8]]

Exemplu

**import** numpy **as** np  
xs = [3, 1, 2] *# creează o lista*print(xs, xs[2]) *# afișează "[3, 1, 2] 2"*print(xs[-1]) *# valoare negativa indica numărarea de la sfârșitul listei; afișează "2"*xs[2] = **'zi'** *# lista poate conține elemente de diferite tipuri*print(xs) *# afișează "[3, 1, 'zi']"*xs.append(**'noapte'**) *# adaugă un element nou listei*print(xs) *# Prints "[3, 1, 'zi', 'noapte']"*x = xs.pop() *#elimina si returnează o lista fără ultimul element*print(x, xs) *# afișează "noapte [3, 1, 'zi']"*

Python permite extragerea unei părți dintr-o listă

**import** numpy **as** np  
nums = list(range(5)) *# range este o funcție încorporată care creează o listă de numere întregi*print(nums) *# afișează "[0, 1, 2, 3, 4]"*print(nums[2:4]) *# se extrage de la 2 4 4 (exclusive); afișând "[2, 3]"*print(nums[2:]) *# se extrage de la 2 la sfârșitul listei; afișând "[2, 3, 4]"*print(nums[:2]) *# se extrage o parte din listă de la început până la 2 (exclusive); afișând "[0, 1]"*print(nums[:]) *# afișează întreaga listă; afișând "[0, 1, 2, 3, 4]"*nums[2:4] = [8, 9] *# asignează o nouă listă [0, 1, 8, 9, 4]*print(nums)

**Mulțimi**

Multimile sunt seturi neordonate de elemente distincte.

**import** numpy **as** np  
animale = {**'pisica'**, **'caine'**}  
print(**'pisica' in** animale) *# verifică dacă pisica face parte din mulțimea animale și afișează TRUE*print(**'peste' in** animale) *# afișează "False"*animale.add(**'peste'**) *# adaugă un element la mulțime*print(**'peste' in** animale) *# afișează "True"*print(len(animale)) *# numărul de elemente din mulțime; afișează "3"*animale.add(**'pisica'**) *# adaugă un nou element în mulțime, dacă există deja atunci acesta este omis*print(len(animale)) *# afișează "3"*animale.remove(**'pisica'**) *# elimină un element din mulțime*print(len(animale)) *# afișează "2"*

Dicționar

Un dicționar este o mulțime de perechi (cheie, valoare).

Cheile trebuie sa fie imutabile.

Operații:

● creare {} sau {eu: 1, tu: 2}

● accesare valoare pe baza unei chei

**import** numpy **as** np  
a = {**'tu'**: 1, **'eu'**: 2,**'el'**: 3}  
print(a)  
*#get a value for a key*print(a[**'tu'**]) *# afișează "1"*  
print(a[**'eu'**]) *# afișează "2"*  
print(a[**'el'**]) *# afișează "3"*

**Instrucțiunea if**

if test expression:

statement(s)

**import** numpy **as** np  
a = 100  
b = 30  
**if** b > a:  
 print(**"b este mai mare decat a"**)  
**elif** a == b:  
 print(**"a si b sunt egale"**)  
**else**:  
 print(**"a este mai mare decat b"**)

**Instrucțiunea for**

for <var> in <iterable>:

<statement(s)>

**Exemplul 1**

**import** numpy **as** np  
a = [**'one'**, **'two'**, **'three'**]  
**for** i **in** a:  
print(i)

*# afiseaza one two three*

**Exemplul 2**

**import** numpy **as** np  
**for** x **in** range(6):  
 print(x)  
*#afiseaza 0 1 2 3 4 5*adj = [**"rosu"**, **"mare"**, **"gustos"**]  
fructe = [**"mar"**, **"strugure"**, **"cirese"**]  
**for** x **in** adj:  
 **for** y **in** fructe:  
 print(x, y)  
*# afiseaza   
# rosu mar  
# rosu strugure  
# rosu cirese  
# mare mar  
# mare strugure  
# mare cirese*

## if elif

if conditie:

bloc\_code\_1

elif conditie \_2:

bloc\_code\_2

elif conditie \_3: bloc\_code\_3

…

else:

bloc\_code\_n

**Exemplu**

**import** numpy **as** np  
num = 2000  
**if** 9 < num < 99:  
 print(**"numărul este format din doua cifre"**)  
**elif** 99 < num < 999:  
 print(**"numărul este format din trei cifre"**)  
**elif** 999 < num < 9999:  
 print(**"numărul este format din patru cifre"**)  
**else**:  
 print(**"numărul are mai mult de patru cifre"**)

# afișează : **numărul are mai mult de patru cifre**

**Funcții**

Funcțiile permit definirea unui bloc de cod care poate fi utilizat într-un program[[20]](#footnote-20).

Funcțiile se declară folosind keyword-ul **def[[21]](#footnote-21)**, acestora nu li se specifică tip de return sau tipuri pentru parametri.

Python oferă mai multe funcții predefinite, cum ar fi print(), len() sau type(), sau funcții definite de utilizator în cadrul programelor.

Inputul funcției sunt parametri formali care urmează a fi prelucrați cu niște operații și returnează un rezultat sub formă de output.

Sintaxa:

def numele\_funcției(<argumentele folosite în funcție, despărțite prin virgulă>):

# <instrucțiuni>

#numele\_funcției(parametri efectivi)[[22]](#footnote-22)

Exemplu

Să se calculeze valoarea funcției următoare pentru un n dat.



n =8

def subp1(x):

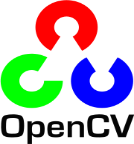
s = 0

for i in range(x):

s += 1/(i+1)

return s

print( subp1(n) )

 **Lucru cu imagini**

Lucru cu imagini face obiectul acestui curs.

OpenCV[[23]](#footnote-23) este un proiect de computer vision dezvoltat de Intel Corporations, un aport important aducându-l-l experți de la Intel Rusia, precum *Intel Performance Librarly.*

Inițial proiectul OpenCV a avut în vedere realizarea unei infrastructuri comune pentru programatori astfel codurile sursă să fie transferabile și inteligibile.

Versiunea Alpha a fost în anul 2000, urmând ca versiunea 1.0 să fie lansată în anul 2006, in momentul actual al anului 2022 se operează cu versiunea OpenCV – 4.6.0

*OpenCV este aplicat în domenii de interes major precum:*

* Studierea imaginilor pentru 2D și 3D
* Studierea mișcării sau a gesturilor
* Recunoaștere facială
* Preluarea imaginilor cu algoritmi de segmentare.
* Analiza imaginilor
* Augmentare
* Pe lângă cele enumerate OpenCV include o bibliotecă de [învățare automată](https://ro.wikipedia.org/wiki/%C3%8Env%C4%83%C8%9Bare_automat%C4%83) din categoria Vector Machine sau Deep Learning.[[24]](#footnote-24)

Asigurați-vă că aveți instalată biblioteca OpenCV

1.1 Citim o imagine si o afișăm intr-o fereastra

**import** cv2  
*#stabilim calea către imagine*path =**r'C:\Users\elev\Desktop\python\lena.jpg'***#citim imaginea cu funcția imread*img = cv2.imread(path)  
*#afișăm imaginea cu funcția imshow*cv2.imshow(**'Poza noastră'**, img)  
*#adugăm un timp de așteptare pentru afișarea imaginii*cv2.waitKey(5000)  
*#închidem toate ferestrele*cv2.destroyAllWindows()

Discuții:

1. Calea absoluta către imagine este data de path, pentru a lucra cu propriile imagini adăugați calea din propriul PC, sau salvați imaginea în dosarul curent.

2. Citirea unei imagini cu funcția imread

Sintaxă: cv2.imread(path, flag)[[25]](#footnote-25)

Parametrii:  
path: calea către imagine  
flag: este un argument folosit pentru afișarea imaginilor color, cu niveluri de gri sau imagini cu canal de tip alpha. In funcția imread argumentul flag poate fi cv2.IMREAD\_COLOR sau 1, cv2.IMREAD\_GRAYSCALE sau 0, cv2.IMREAD\_UNCHANGED sau -1

3. Afișarea unei imagini cu funcția imshow

**Sintază:** cv2.imshow(window\_name, image)  
**Parametrii:**   
**window\_name:** numele feresteri în care va fi agfișată imaginea.   
**image:** variabila de tip imagine.  
4. Funcția **waitkey() afiseză pe ecran o imaginile pe o perioadă de timp limitat după care dispare de pe ecran. Dacă funcția primește ca argument 0 atunci imaginea va fi afișată pe ecran până la apăsarea unei taste.**

5. destroyAllWindows()închide toate ferestrele generate de prelucrările cu OpenCV.Daca funcția primește ca argument numele unei imagini atunci va fi închisă doar fereastra care conține imaginea respectivă.

1.2 Salvarea unei imagini cu extensia dorita

**import** cv2  
path = **r'C:\Users\elev\Desktop\python\lena.jpg'**img = cv2.imread(path)

*#salvăm imaginea cu extensia png și verificăm statusul acesteia, daca #obținem TRUE atunci imaginea este salvată cu succes*

status = cv2.imwrite(**r'C:\Users\elev\Desktop\python\grey.png'**, img)  
print(**"Image written to file-system : "**, status)  
cv2.imshow(**'Lena'**, img)

**Bibiloteca Matplotlib**

Matplotlib este o bibliotecă open source din Python care servește ca un utilitar de vizualizare[[26]](#footnote-26). Cele mai multe dintre utilitarele Matplotlib se află în submodulul pyplot și sunt de obicei importate sub aliasul plt:

import matplotlib.pyplot as plt

Această librărie se instalează folosind comanda:

python -mpip install -U matplotlib

Cea mai folosită funcție din Matplotlib este **subplot**() , aceasta creează o grilă de subploturi cu un singur apel, oferind în același timp un control rezonabil asupra modului în care sunt create subploturi individuale.

[plt.subplots](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplots.html#matplotlib.pyplot.subplots)(m, n, t) va crea un panou cu m linii, n coloane, iar t reprezinta pozitia pe care va fi asignată imaginea.

Exemplul următor va crea un panou de dimensiuea 2x2 iar imaginea cu numele img va fi pusă pe poziția 4. Imaginea va avea titlul Imagine originală.

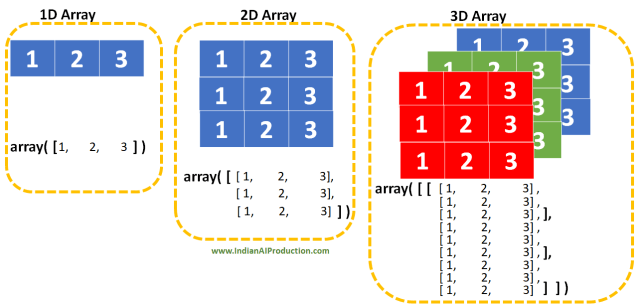
plt.subplot(224), plt.imshow(img,**'gray'**),plt.title(**Imagine originală**)

1.3 Afișarea caracteristicilor unei imagini și redimensionarea acesteia

Când se lucrează cu OpenCV Python, imaginile sunt stocate în **ndarray numpy**. Pentru a obține forma sau dimensiunea imaginii, utilizați ndarray.shape pentru a obține dimensiunile imaginii. Apoi, puteți utiliza indexul variabilei de dimensiuni pentru a obține lățimea, înălțimea și numărul de canale pentru fiecare pixel.

Ndarray este una dintre cele mai importante clase din biblioteca NumPy python. Practic este o matrice multidimensională sau n-dimensională de dimensiune fixă cu elemente omogene (adică tipul de date al tuturor elementelor din matrice este același).

O matrice multidimensională arată așa:



Pentru redimensionare folosim funcția cu sintaxa:

cv2.resize(src, dsize[, dst[, fx[, fy[, interpolation]]]])

unde:

|  |  |
| --- | --- |
| src | Imaginea imput |
| dsize | Dimensiunea cu care se redimensionează imaginea output |
| fx | Un factor de scalare opțional aplicabil de-a lungul axei orizontale |
| fy | Un factor de scalare opțional aplicabil de-a lungul axei orizontale |
| interpolation | Un argument opțional care va apela următoarele metode de interpolare  INTER\_NEAREST – interpolare bazată pe cel mai apropiat vecin  INTER\_LINEAR – interpolare bilineară (implicită)  INTER\_AREA – eșantionare bazată pe relația dintre ariile zonelor de pixeli.  NTER\_CUBIC –interpolare bicubică bazată pe vecinătățile 4×4 pixeli  INTER\_LANCZOS4 – interpolare Lanczos interpolation pe vecinătățile 8×8 pixeli |

În codul următor, am citit o imagine pe care să o redimensionăm iar apoi am folosim ndarray.shape pentru a obține dimensiunile imaginii.

**import** cv2  
path =**r'C:\Users\elev\Desktop\python\lena.jpg'**img = cv2.imread(path)  
height = img.shape[0]  
width = img.shape[1]  
channels = img.shape[2]  
dimensions = img.shape  
print(**'Înălțimea imaginii:'**, height)  
print(**'Lățimea imaginii : '**, width)  
print(**'Numărul de canale:'**, channels)  
print(**'Dimensiunile imaginii:'**, dimensions)  
cv2.imshow(**"Imaginea originală"**, img)  
*#redimensionarea imaginii*imM = cv2.resize(img, (400, 400))  
heightM = imM.shape[0]  
widthM = imM.shape[1]  
channelsM = imM.shape[2]  
dimensionsM = imM.shape  
print(**'Înălțimea imaginii redimensionate: '**, heightM)  
print(**'Lățimea imaginii redimensionate: '**, widthM)  
print(**'Numărul de canale redimensionate:'**, channelsM)  
print(**'Dimensiunile imaginii redimensionate:'**, dimensionsM)  
cv2.imshow(**"Imaginea redimensionata"**, imM)  
cv2.waitKey()  
cv2.destroyAllWindows()

Ca output avem rezultatele

Înălțimea imaginii: 232

Lățimea imaginii : 217

Numărul de canale: 3

Dimensiunile imaginii: (232, 217, 3)

Caracteristicile imaginii redimensionate

Înălțimea imaginii redimensionate: 400

Lățimea imaginii redimensionate: 400

Numărul de canale redimensionate: 3

Dimensiunile imaginii redimensionate: (400, 400, 3)

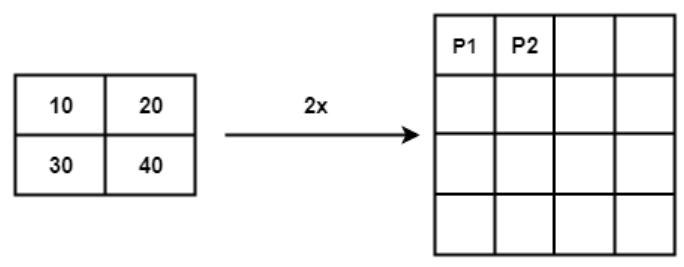
**Interpolarea imaginilor**

Scalarea este o operație geometrica care redimensionează o imagine. Redimensionarea imaginii se realizează cu ajutorul unor factori de scalare, acești fac ca un grup de pixeli din imaginea originală sa fie înlocuit cu un alt grup, înlocuirea făcându-se prin interpolarea intensităților. Cele mai cunoscute interpolări care vor fi studiate sunt: Interpolarea bazată pe cel mai apropiat vecin, biliniară și bicubică.[[27]](#footnote-27)

**I. Interpolarea bazată pe cel mai apropiat vecin**

O interpretare grafică funcționează astfel:

* (a) să presupunem că avem o imagine 2×2



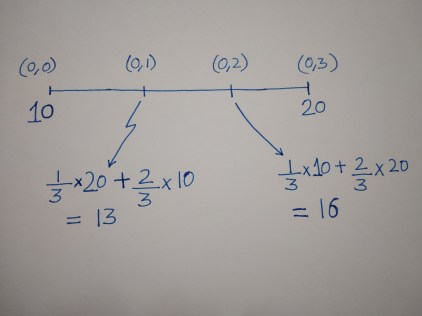
(a) (b)

* Se stabilește un factor de scalare 2x
* (b) se obține imaginea din figura (b)
* Atașăm pixelului din imaginea necunoscută notat cu „P1” o valoare egala cu cea mai apropiată valoare a intensității acesta din zona 2x2.
* Așa se procedează și cu pixelul notat cu P2
* Algoritmul continuă până la ultimul pixel din imagine

**II. Interpolarea biliniară**

Interpolarea biliniară presupune aplicarea unei interpolări liniare în două direcții. Astfel, folosesc cei mai apropiați 4 vecini, se face media lor ponderată iar noul rezultat este atașat pixelului din imaginea nouă.

Interpolarea liniară înseamnă că estimăm valoarea rezultată folosind polinoame liniare. Să presupunem că avem doi pixeli cu intensitățile de 10 și 20 și dorim să calculăm valorile dintre ele. Interpolarea liniară simplă se calculează astfel:



**III. Interpolarea bicubică**

Interpolarea bicubică poate fi folosită pentru a face texturile să arate mai frumos atunci când sunt scalate, este o metodă mai bună decât interpolarea standard biliniară a texturii.

Interpolarea bi-lineară folosește 4 vecini cei mai apropiați pentru a determina ieșirea, în timp ce bi-cubica folosește 16 (4×4 vecinătate).

Distribuția ponderilor se face diferit.

* Calculați poziția pixelului țintă în imaginea originală.
* Determinați cei șaisprezece pixeli învecinați din jurul pixelului țintă.
* Aplicați ponderi fiecărui pixel învecinat în funcție de poziția sa relativă față de pixelul țintă.
* Calculați valoarea pixelului interpolat prin însumarea contribuțiilor ponderate ale pixelilor vecini.
* Repetați procesul pentru fiecare pixel țintă din imaginea de ieșire.
* **import** cv2  
  img = cv2.imread(**'Lenna.png'**)  
  **from** matplotlib **import** pyplot **as** plt  
  bicubic\_img = cv2.resize(img,**None**, fx = 2, fy = 2, interpolation = cv2.INTER\_CUBIC)  
  near\_img = cv2.resize(img,**None**, fx = 2, fy = 2, interpolation = cv2.INTER\_NEAREST)  
  bilinear\_img = cv2.resize(img,**None**, fx = 2, fy = 2, interpolation = cv2.INTER\_LINEAR)  
  plt.subplot(221), plt.imshow(img, **'gray'**),plt.title(**'Imagine originala'**)  
  plt.subplot(222), plt.imshow(bicubic\_img ,**'gray'**),plt.title(**'bicubic'**)  
  plt.subplot(223), plt.imshow(bilinear\_img,**'gray'**),plt.title(**'cel mai apropiat vecin'**)  
  plt.subplot(224), plt.imshow(bilinear\_img,**'gray'**),plt.title(**'bilinara'**)  
  plt.show()  
  cv2.waitKey(1000)  
  cv2.destroyAllWindows()

Discuții

* Shape este un atribut cu ajutorul căruia afișăm dimensiunile imaginii.
* Dacă doriți ca proiectul dumneavoastră să conțină mai multe scripturi python reluați procedura de creare a unui fișier cu extensia py.
* Este avantajos să lucrați cu mai multe fișiere python în același proiect deoarece acestea folosesc în comun toate bibliotecile atașate proiectului.
* Când se dorește rularea altui fișier python, mergeți în secțiunea Edit Configuration iar din fereastra Run/Debug Configurations->Script path->selectați fișierul

**Webografie**

* <https://bionescu.aimultimedialab.ro/index_files/tapai/TAPAI_BIonescu_M1.pdf>
* https://koaha.org/wiki/Elaborazione\_delle\_immagini
* <https://www.telacad.ro/10-motive-pentru-python/?gclid=CjwKCAjwi8iXBhBeEiwAKbUofTlzg1_MOXYoVS4-xSlsb5KsFp1D56cDso06bleLW9cH_ytxq4a07RoC3mkQAvD_BwE>
* <https://ocw.cs.pub.ro/courses/asc/laboratoare/01>
* <https://cs231n.github.io/python-numpy-tutorial/#numpy>
* <https://mahotas.readthedocs.io/en/latest/>
* <https://opensource.com/article/19/3/python-image-manipulation-tools>
* <https://www.geeksforgeeks.org/python-opencv-cv2-imread-method/>
* <https://theailearner.com/2018/12/29/image-processing-bicubic-interpolation/>
* <https://theailearner.com/2018/12/29/image-processing-bicubic-interpolation/>
* https://indianaiproduction.com/python-numpy-array/

Anexa imagini

|  |  |
| --- | --- |
| Black is the darkest color, and something we call an achromatic color, meaning it has no hue. | https://www.worldatlas.com/articles/is-black-a-color.html |

1. Care este forma matematică a unei imagini? [↑](#footnote-ref-1)
2. Enumerați trei domenii de cercetare în care este folosit procesul de prelucrare a imaginilor [↑](#footnote-ref-2)
3. Enumerați trei tipuri aplicații care folosesc tehnica PI [↑](#footnote-ref-3)
4. Ce reprezintă iluminarea neuniformă? [↑](#footnote-ref-4)
5. Ce este contrastul? [↑](#footnote-ref-5)
6. De ce tip este informația parazită dintr-o imagine? [↑](#footnote-ref-6)
7. Enumerați trei tipuri de imagini cu care se lucrează în imagistica medicala [↑](#footnote-ref-7)
8. Ce reprezintă imagistica satelitara? [↑](#footnote-ref-8)
9. Cu ce scop se folosește PI în supraveghere video? [↑](#footnote-ref-9)
10. Enumerați etapele unui sistem de prelucrare a imaginilor [↑](#footnote-ref-10)
11. Definiți noțiunea de imagine [↑](#footnote-ref-11)
12. Care sunt elementele de bază ale unei imagini? [↑](#footnote-ref-12)
13. Definiți noțiunea de rezoluție spațială [↑](#footnote-ref-13)
14. Dați un exemplu de dimensiune a unui pixel? [↑](#footnote-ref-14)
15. Cine da claritatea imaginii? [↑](#footnote-ref-15)
16. Cum sunt reprezentate imaginile digitale ? [↑](#footnote-ref-16)
17. Ce este un spațiu de culoare? [↑](#footnote-ref-17)
18. Dați un exemplu de imagine vectorială [↑](#footnote-ref-18)
19. Enumerați trei avantaje ale limbajului Python [↑](#footnote-ref-19)
20. Definiți noțiunea de funcție. [↑](#footnote-ref-20)
21. Care este keyword-ul specific funcțiilor definite in Python [↑](#footnote-ref-21)
22. Sintaxa unui funcții în Python [↑](#footnote-ref-22)
23. Instrumentul necesar folosit în Python pentru prelucrarea imaginilor [↑](#footnote-ref-23)
24. Enumerați domeniile de interes major pentru care este folosit instrumentul OpenCv [↑](#footnote-ref-24)
25. Numele funcției folosit pentru citirea imaginilor [↑](#footnote-ref-25)
26. La ce folosește utilitarul Matplotlib? [↑](#footnote-ref-26)
27. Enumerați tipurile de interpolări ale imaginilor. [↑](#footnote-ref-27)