1. Introducere în utilizarea bibliotecii OpenCV

1.1. Introducere

Scopul acestei lucrări de laborator este de a familiariza studenții cu aplicația cadru care va fi folosită în activitatea practică la disciplina Procesarea Imaginilor.

Cunoștințele necesare pentru a putea parcurge acest laborator sunt:

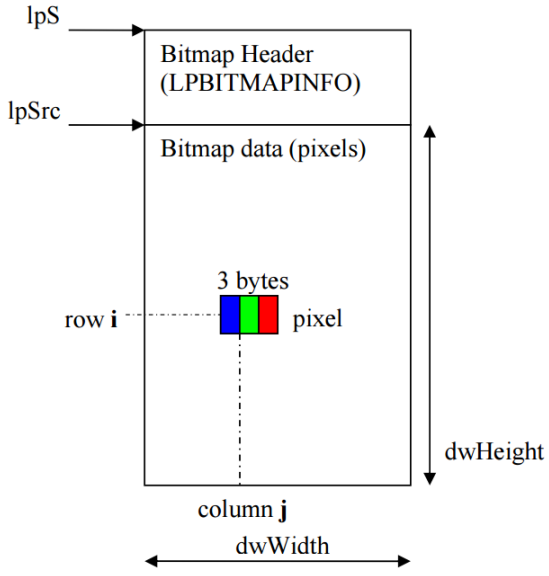
- Obligatorii: Programarea în C++, Structuri de date și algoritmi
- Opționale: *Metode orientate pe obiect, Algoritmi fundamentali, Algebră liniară, Matematici discrete, C++, Visual C++ (Visual Studio)*

1.2. Formatul de imagine Bitmap

Formatul "bmp" este folosit pentru a stoca imagini în formă necomprimată. Acest format stochează imaginile digitale în format rastru (matrice), independent de dispozitivul de afișare, și poate stoca imagini monocrome sau color cu mai multe nivele de adâncime a culorii. Nivelul de adâncime a culorii determină numărul de culori ale unei imagini, și dimensiunea memoriei necesare pentru a stoca această imagine. Fișierul "bmp" are următoarea structură:

- Un antet (header) al fișierului bitmap – conține semnătura tipului bmp, dimensiunea fișierului și punctul de început al șirului de pixeli;
- Antetul DIB – conține informații precum dimensiunea imaginii (înălțime, lățime), și numărul de biți per pixel;
- Tabelul culorilor (sau tabelul look-up) – pentru imagini color care folosesc paletă;
- Șirul de pixeli – imaginea propriu zisă, ca un șir unidimensional de valori, ce poate include și valori de umplutură pentru alinierea datelor.

Următoarea imagine ilustrează formatul bitmap pentru o imagine de 24 de biți. Dimensiunile imaginii, înălțime și lățime, sunt notate cu dwHeight și dwWidth.



1.3. Vedere de ansamblu asupra mediului OpenCV

Mediul în care veți lucra conține biblioteca OpenCV 4.5.1 împreună cu Visual Studio 2019. Setările pentru include sunt preconfigurate, și toate bibliotecile statice și dinamice sunt incluse în soluție.

Sarcina voastră este de a crea funcții noi care vor fi apelate din funcția principală (main). Aceste funcții trebuie grupate în funcție de ședințele de laborator la care veți participa și trebuie să primească nume sugestive. Toate exemplele de cod presupun că ați inclus namespace-ul cv (*using namespace cv*), altfel toate clasele și metodele Open CV trebuie prefixate cu **`cv::`** . Următorul fragment de cod vă indică ce trebuie să faceți pentru a introduce noi funcții (textul pe fond gri indică ce trebuie să adăugați în plus):

```
void negative_image(){
     //implement function
}
int main(){
     int op;
     do{
          printf("Menu:\n");
          //...
          printf(" 7 - L1 Negative Image \n");
          //...
          printf(" 0 - Exit\n\n");
          printf("Option: ");
          scanf("%d",&op);
          switch (op)
          {
               //...
               case 7:
                    negative_image();
                    break;
          }
     }
     while (op!=0);
     return 0;
}
```

Trebuie să salvați ce ați lucrat la fiecare ședință de laborator. Dimensiunea proiectului poate fi redusă apelând scriptul clean.bat, care șterge toate fișierele generate de compilator. O altă metodă este de a salva doar fișierul cpp principal, deoarece restul fișierelor proiectului nu se schimbă.

1.4. Clasa Mat

În OpenCV imaginile sunt stocate în memorie ca obiecte ale clasei Mat. Această clasă este o clasă pentru manipularea matricelor generice bidimensionale, sau a matricelor cu mai multe dimensiuni.

Câmpurile importante ale clasei Mat sunt:

- rows – numărul de rânduri ale matricei = înălțimea imaginii;
- cols – numărul de coloane ale matricei = lățimea imaginii;
- data – un pointer la locația din memorie a pixelilor imaginii

Cea mai simplă și curată metodă de a crea un obiect de tip Mat este de a apela constructorul cu trei parametri:

```
Mat img(rows, cols, type);
```

Ultimul parametru indică tipul de date stocat în matrice. Un exemplu de tip este CV_8UC1, care reprezintă: 8 biți, fără semn, un singur canal. În general primul număr după CV_ reprezintă numărul de biți, litera reprezintă tipul, iar Cx indică numărul de canale pentru un pixel.

Tipurile de date esențiale sunt prezentate în următorul tabel:

Codul type	**Tipul de date**	**Utilizare**
CV_8UC1	unsigned char	Imagine monocromă (8bits/pixel)
CV_8UC3	Vec3b	Imagine color (3x8bits/pixel)
CV_16SC1	short	Stocare date
CV_32FC1	float	Stocare date
CV_64FC1	double	Stocare date

Exemplu 1 – crearea unei imagini monocrome de dimensiune 256x256:

```
Mat img(256,256,CV_8UC1);
```

Exemplu 2 – crearea unei imagini color cu 720 rânduri și 1280 coloane:

```
Mat img(720,1280,CV_8UC3);
```

Exemplu 3 – crearea unei matrice cu numere reale de dimensiune 2x2, care conține valorile: [1 2; 3 4], și afișarea acesteia:

```
float vals[4] = {1, 2, 3, 4};
Mat M(2,2,CV_32FC1,vals); //constructor cu 4 parametri
std::cout << M << std::endl;
```

După cum se poate observa, un obiect de tip Mat se poate afișa folosind stream-ul de ieșire standard.

Pentru o descriere detaliată a clasei Mat, se poate consulta documentația oficială: https://docs.opencv.org/4.5.1/d3/d63/classcv_1_1Mat.html

1.5. Citirea unei imagini din fișier

Pentru a deschide un fișier imagine și a stoca conținutul acestuia într-un obiect de tip Mat, se folosește funcția imread:

```
Mat img = imread("path_to_image", flag);
```

Primul parametru este calea relativă sau absolută spre fișierul imagine; al doilea parametru, flag, poate fi:

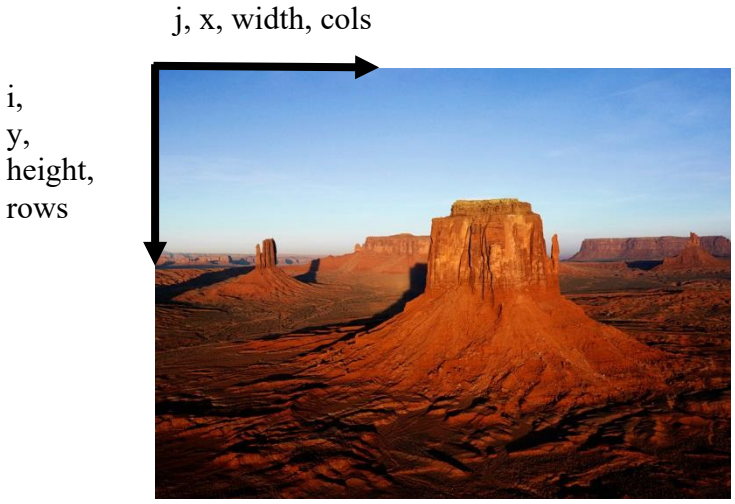
- IMREAD_UNCHANGED (-1) – deschide imaginea în același format în care este salvată pe disc;
- IMREAD_GRAYSCALE (0) – deschide imaginea ca imagine monocromă (grayscale); încărcarea face conversia la CV_8UC1 (1 canal, unsigned char);
- IMREAD_COLOR (1) – se încarcă imaginea și se convertește la CV_8UC3 (3 canale unsigned char);

Exemplu 1 – se deschide o imagine din directorul curent, în formatul în care a fost salvată:

```
Mat img = imread("cameraman.bmp", -1);
```

1.6. Accesarea datelor din imagine

Elementele din matrice sunt indexate conform convenției standard din matematică. Acest lucru înseamnă că originea va fi în colțul stânga sus al imaginii, primul parametru va indica rândul (crescător în jos), iar al doilea parametru va indica coloana (crescător spre dreapta). Următoarea figură ilustrează schema de indexare:



Respectați această convenție întotdeauna, pentru a evita erorile. Când parcurgeți o imagine folosind două cicluri for, ciclul exterior va parcurge rândurile, iar cel interior coloanele.

Pentru a accesa pixelul de la coordonatele (i, j) dintr-o imagine monocromă img, se va folosi metoda *at*:

```
unsigned char pixel = img.at<unsigned char>(i,j);
```

După cum se poate observa, trebuie să specificați tipul de date care sunt stocate în matrice (unsigned char).

Pentru acces mai rapid se poate utiliza direct pointerul la datele matricei:

```
unsigned char pixel = img.data[i*img.step[0] + j];
```

Toți pixelii sunt stocați într-un șir unidimensional, rând după rând, și pixelii de pe rând ordonați de la stânga la dreapta, începând cu adresa indicată de pointerul `img.data`. Pe un rând pot exista mai mulți pixeli decât numărul de coloane, deci **accesarea pixelilor folosind `i*img.cols+j` poate duce la rezultate greșite**!

Se poate obține un pointer individual pentru rândul i:

```
unsigned char pixel = img.ptr(i)[j];
```

Pentru a accesa cele trei componente de culoare a unui pixel de pe rândul i și coloana j a unei imagini color, trebuie folosit tipul dedicat Vec3b:

```
Vec3b pixel = img.at<Vec3b>(i,j);
unsigned char B = pixel[0];
unsigned char G = pixel[1];
unsigned char R = pixel[2];
```

Vec3b este un vector cu trei componente de tip byte (unsigned char), și este tipul recomandat pentru manipularea imaginilor color.

Codul poate fi simplificat prin utilizarea clasei cu șablon Mat_<T> , subclasă a clasei Mat, care permite omiterea tipului pentru operațiile de acces. La crearea unui obiect Mat_<T> trebuie să specificați tipul care este stocat în matrice.

```
Mat_<uchar> img = imread("fname", IMREAD_GRAYSCALE);
uchar pixel = img(i,j);
```

În exemplul de mai sus am folosit specificatorul de tip uchar, care este echivalent cu unsigned char. Accesarea unei valori (pixel) de la o anumită poziție permite și citirea, și scrierea acesteia.

1.7. Vizualizarea unei imagini

Pentru a vizualiza o imagine se folosește funcția imshow, urmată de un apel al funcției waitKey:

```
imshow("image", img);
waitKey(0);
```

Aceste linii de cod vor afișa imaginea într-o fereastră nouă și apoi programul va aștepta ca utilizatorul să apese o tastă. Funcția waitKey are un singur parametru: cât de mult să aștepte după acțiunea utilizatorului (în milisecunde). Valoarea zero înseamnă că sistemul va aștepta un timp infinit.

Folosiți întotdeauna waitKey după imshow. Ferestrele de tip imagine pot fi mutate sau redimensionate, pentru a facilita procesul de analiză a rezultatelor procesării.

1.8. Salvarea/scrierea unei imagini pe disc

Pentru a salva o imagine, folosiți funcția imwrite:

```
imwrite("fname", img);
```

Numele fișierului trebuie să conțină calea (directorul), numele și extensia, care va determina formatul în care se va face scrierea.

1.9. Funcție exemplu

Următoarea funcție încarcă o imagine monocromă și o transformă în negativul ei:

```
void negative_image(){
	Mat img = imread("Images/cameraman.bmp",
					IMREAD_GRAYSCALE);
	for(int i=0; i<img.rows; i++){
		for(int j=0; j<img.cols; j++){
			img.at<uchar>(i,j) = 255 - img.at<uchar>(i,j);
		}
	}
	imshow("negative image",img);
	waitKey(0);
}
```

Fișierul imagine trebuie să fie localizat în directorul Images, inclus în directorul proiectului aplicației cadru.

1.10. Activitate practică individuală

1. Descărcați și compilați aplicația *OpenCVApplication*.
2. Testați funcția negative_image().
3. Implementați o funcție care schimbă nivelele de gri cu un factor aditiv.
4. Implementați o funcție care schimbă nivelele de gri cu un factor multiplicativ. Salvați imaginea rezultat.
5. Creați o imagine color de dimensiune 256 x 256. Împărțiți imaginea în 4 cadrane egale, și colorați acestea, din stânga-sus până în dreapta-jos, astfel: alb, roșu, verde, galben.
6. Creați o matrice 3x3 de tip float, determinați inversa ei și tipăriți-o.
7. **Salvați-vă ceea ce ați lucrat. Utilizați aceeași aplicație în laboratoarele viitoare. La sfârșitul laboratorului de procesare a imaginilor va trebui să prezentați propria aplicație cu algoritmii implementați!!!**

1.11. Bibliografie

https://docs.opencv.org/4.5.1/index.html