1. Introducere in OpenCV. Framework-ul OpenCVApplication.

1.1. Introducere

Scopul acestui laborator este de a prezenta Framework-ul *OpenCVApplication* menit sa faciliteze folosirea si integrarea bibliotecii OpenCV (Open Computer Vision Library) (http://opencv.org/). Aceasta biblioteca integreaza implementari de algoritmi din domeniul viziunii artificale de la procesari de baza pana la metode complexe de nivel inalt.

Exemplele din setul de lucrari care urmeaza utilizeaza versiunea *OpenCV 2.4.13*, precompilata, integrarea versiunilor urmatoare fiind similara.

Documentatille referitoare la aceasta biblioteca se pot accesa on-line la adresa: http://docs.opencv.org/2.4.13/ [1],[2] sau in foldderul *\doc* al aplicatiei *OpenCVApplication*. O sinteza a elementelor de baza legata de aceasta biblioteca gasiti si in referinta [3], Anexa 2.

1.2. Aplicatia OpenCV apllication

Aplicatia *OpenCVApplication* include toate librariile necesare din OpenCV astfel incat nu aveti nevoie de dependinte suplimentare pentru dezvoltarea/rularea aplicatiei (deci nu este neviue sa va instalati biblioteca OpenCV pe calculatorul personal).

IDE-ul folosit va fi Visual Studio 2013 (VS12) iar limbajul de dezvoltare C++.

Interactiunea cu utilizatorul se va face prin linia de comanda (fig. 1) iar pentru manipularea imagini/filmelor si afisarea rezultatelor se vor folosi functiile din GUI-ul OpenCV.

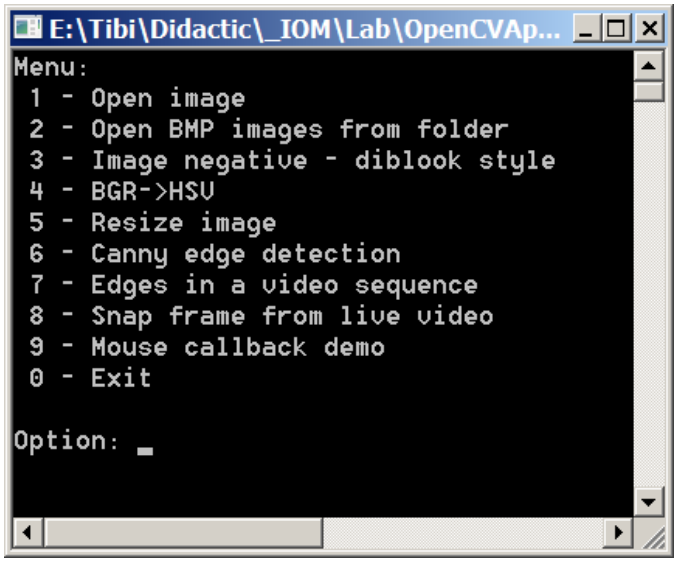

Fig. 1. Interfata cu utilizatorul.

Toate functiile de procesare noi se vor include/apela din modulul principal al aplicatiei: *OpenCVApplication.cpp* dupa urmatorul sablon:

```
int main()
{
	int op;
	do
	{
		system("cls");
		destroyAllWindows();
		printf("Menu:\n");
		printf(" 1 - Open image\n");
		printf(" 2 - Open BMP images from folder\n");
		printf(" 3 - Image negative - diblook style\n");
```

```
		printf(" 4 - BGR->HSV\n");
		printf(" 5 - Resize image\n");
		printf(" 6 - Canny edge detection\n");
		printf(" 7 - Edges in a video sequence\n");
		printf(" 8 - Snap frame from live video\n");
		printf(" 9 - Mouse callback demo\n");
		printf(" 0 - Exit\n\n");
		printf("Option: ");
		scanf("%d",&op);
		switch (op)
		{
			case 1:
				testOpenImage();
				break;
			case 2:
				testOpenImagesFld();
				break;
			case 3:
				testParcurgereSimplaDiblookStyle(); //diblook style
				break;
			case 4:
				//testColor2Gray();
				testBGR2HSV();
				break;
			case 5:
				testResize();
				break;
			case 6:
				testCanny();
				break;
			case 7:
				testVideoSequence();
				break;
			case 8:
				testSnap();
				break;
			case 9:
				testMouseClick();
				break;
		}
	}
	while (op!=0);
	return 0;
}
```

1.3. Exemple de functii implementate pentru manipularea si procesarea de imagini si secvente video

Deschiderea si afisarea unei imagini

```
void testOpenImage()
{
	char fname[MAX_PATH];
	while(openFileDlg(fname))
	{
		Mat src;
		src = imread(fname);
		imshow("image",src);
		waitKey();
	}
}
```

Deschidere imagie, calcul negativ si afisare susrsa si rezultat

Varianta lenta cu acces la pixeli folosind metoda *at*. Este ilustrat si modul in care puteti masura timpul de procesare:

```
void testNegativeImage()
{
	char fname[MAX_PATH];
	while(openFileDlg(fname))
	{
		double t = (double)getTickCount(); // Get the current time [s]

		Mat src = imread(fname,CV_LOAD_IMAGE_GRAYSCALE);
		int height = src.rows;
		int width = src.cols;
		Mat dst = Mat(height,width,CV_8UC1);
		// Asa se acceseaaza pixelii individuali pt. o imagine cu 8 biti/pixel
		// Varianta ineficienta (lenta)
		for (int i=0; i<height; i++)
		{
			for (int j=0; j<width; j++)
			{
				uchar val = src.at<uchar>(i,j);
				uchar neg = MAX_PATH-val;
				dst.at<uchar>(i,j) = neg;
			}
		}

		// Get the current time again and compute the time difference [s]
		t = ((double)getTickCount() - t) / getTickFrequency();
		// Print (in the console window) the processing time in [ms]
		printf("Time = %.3f [ms]\n", t * 1000);

		imshow("input image",src);
		imshow("negative image",dst);
		waitKey();
	}
}
```

Varianta rapida cu acces la pixeli prin intermediul pointerilor la zona de date (bitmap data), similar cu accesul folosit in framework-ul Diblook.

```
void testParcurgereSimplaDiblookStyle()
{
	char fname[MAX_PATH];
	while (openFileDlg(fname))
	{
		Mat src = imread(fname, CV_LOAD_IMAGE_GRAYSCALE);
		int height = src.rows;
		int width = src.cols;
		Mat dst = src.clone();

		double t = (double)getTickCount(); // Get the current time [s]

		// the fastest approach using the “diblook style”
		uchar *lpSrc = src.data;
		uchar *lpDst = dst.data;
		int w = src.step; // no dword alignment is done !!!
		for (int i = 0; i<height; i++)
```

```
                    for (int j = 0; j < width; j++) {
                        uchar val = lpSrc[i*w + j];
                        lpDst[i*w + j] = 255 - val;
                    }

            // Get the current time again and compute the time difference [s]
            t = ((double)getTickCount() - t) / getTickFrequency();
            // Print (in the console window) the processing time in [ms]
            printf("Time = %.3f [ms]\n", t * 1000);

            imshow("input image", src);
            imshow("negative image", dst);
            waitKey();
        }
}
```

Deschidere imagine (RGB24), conversie color in grayscale si afisare sursa si rezultat

Varianta lenta cu acces la pixeli folosind metoda *at*:

```
void testColor2Gray()
{
        char fname[MAX_PATH];
        while(openFileDlg(fname))
        {
            Mat src = imread(fname);
            int height = src.rows;
            int width = src.cols;
            Mat dst = Mat(height,width,CV_8UC1);
            // Asa se acceseaaza pixelii individuali pt. o imagine RGB
            // Varianta ineficienta (lenta)
            for (int i=0; i<height; i++)
            {
                for (int j=0; j<width; j++)
                {
                    Vec3b v3 = src.at<Vec3b>(i,j);
                    uchar b = v3[0];
                    uchar g = v3[1];
                    uchar r = v3[2];
                    dst.at<uchar>(i,j) = (r+g+b)/3;
                }
            }
            imshow("input image",src);
            imshow("gray image",dst);
            waitKey();
        }
}
```

Transformare imagine color din modelul RGB in HSV si afisare component H,S,V

Conversia intre cele 2 modele de culoare se face cu ajutorul functiei OpenCV `cvtColor(src, dest, tip_concersie)`. Imagine arezultata (modelul HSV) este o matrice cu elemente cu 24 biti/pixel. Este ilustrat accesul eficient la compunentele de culoare HSV cu ajutorul pointerilor la zona de date (pixel data). Rezultatele (componentele H,S,V) se for afisa in imagini de tip 8 biti/pixel.

```
void testBGR2HSV()
{
        char fname[MAX_PATH];
```

```
	while (openFileDlg(fname))
	{
		Mat src = imread(fname);
		int height = src.rows;
		int width = src.cols;

		// Componentele d eculoare ale modelului HSV
		Mat H = Mat(height, width, CV_8UC1);
		Mat S = Mat(height, width, CV_8UC1);
		Mat V = Mat(height, width, CV_8UC1);

		// definire pointeri la matricele (8 biti/pixeli) folosite la afisarea
		componentelor individuale H,S,V
		uchar* lpH = H.data;
		uchar* lpS = S.data;
		uchar* lpV = V.data;

		Mat hsvImg;
		cvtColor(src, hsvImg, CV_BGR2HSV);

		// definire pointer la matricea (24 biti/pixeli) a imaginii HSV
		uchar* hsvDataPtr = hsvImg.data;

		for (int i = 0; i<height; i++)
		{
			for (int j = 0; j<width; j++)
			{
				int hi = i*width * 3 + j * 3;
				int gi = i*width + j;

				lpH[gi] = hsvDataPtr[hi] * 510/360;	// lpH = 0 .. 255
				lpS[gi] = hsvDataPtr[hi + 1];		// lpS = 0 .. 255
				lpV[gi] = hsvDataPtr[hi + 2];		// lpV = 0 .. 255
			}
		}

		imshow("input image", src);
		imshow("H", H);
		imshow("S", S);
		imshow("V", V);

		waitKey();
	}
}
```

Deschiderea imagine, detectie puncte de muchie si afisare imagine sursa si rezultat

```
void testCanny()
{
	char fname[MAX_PATH];
	while(openFileDlg(fname))
	{
		Mat src,dst;
		src = imread(fname,CV_LOAD_IMAGE_GRAYSCALE);
		Canny(src,dst,40,100,3);
		imshow("input image",src);
		imshow("canny",dst);
		waitKey();
	}
}
```

Deschiderea secventa video, detectie puncte de muchie si afisare rezultat

```
void testVideoSequence()
{
	VideoCapture cap("Videos/rubic.avi"); // off-line video from file
	//VideoCapture cap(0);	// live video from web cam
	if (!cap.isOpened()) {
		printf("Cannot open video capture device.\n");
		waitKey(0);
		return;
	}

	Mat edges;
	Mat frame;
	char c;

	while (cap.read(frame))
	{
		Mat grayFrame;
		cvtColor(frame, grayFrame, CV_BGR2GRAY);
		Canny(grayFrame,edges,40,100,3);
		imshow("source", frame);
		imshow("gray", grayFrame);
		imshow("edges", edges);
		c = cvWaitKey(0);  // waits a key press to advance to the next frame
		if (c == 27) {
			// press ESC to exit
			printf("ESC pressed - capture finished\n");
			break;  //ESC pressed
		};
	}
}
```

1.4. Mersul lucrarii

1. Se va citi anexa 2 din referinta [3].

2. Se vor rula exemplele existente din aplicatia OpenCV Application. Se va studia codul sursa al exemplelor procesare (metodele de procesare din *OpenCVApplication.cpp*).

3. Adaugati o functie care realizeaza binarizarea unei imagini cu un prag arbitrar (specificat de utilizator). Optati pt variant eficienta/rapida de parcurgere a imaginii (vezi functia `testParcurgereSimplaDiblookStyle()`).

4. Adaugati o functie care realizeaza un flip orizontal si vertical al imaginii de intrare (atat pt. imagini cu 8 biti/pixel cat si 24 biti/pixel). Optati pt variant eficienta/rapida de parcurgere a imaginii (vezi functiile `testParcurgereSimplaDiblookStyle()`,`testBGR2HSV()`). Masurati/afisati timpul de procesare.

5. Adaugati o functie care afiseaza la linia de comanda valorile componentlor de culoare H,S,V ale pixelului peste care se face click (vezi `testBGR2HSV()`, `testMouseClick()`, `MyCallBackFunc()`).

Referinţe

[1] http://docs.opencv.org/
[2] The OpenCV Reference Manual, Release 2.4.13 (`doc\opencv2refman.pdf`), http://docs.opencv.org/2.4.13/

[3] S. Nedevschi, T. Marita, R. Danescu, F. Oniga, R. Brehar, I. Giosan, S. Bota, A. Ciurte, A. Vatavu, „Image Processing - Laboratory Guide”, UTPress Edition, 2016, ISBN 978-606-737-137-6, http://biblioteca.utcluj.ro/carti-online.html