1. Introducere in OpenCV. Framework-ul OpenCVApplication.

1.1. Introducere

Scopul acestui laborator este o recapitulare a Framework-ului *OpenCVApplication* (pe care l-ati folosit deja la *Procesarea Imaginilor*)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 este aplicabil pt. versiunea *OpenCV 2.4.xx* sau 3.4.xx 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 si in referinta [4].

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 poate fi Visual Studio 2013, 2017 sau 2019 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.

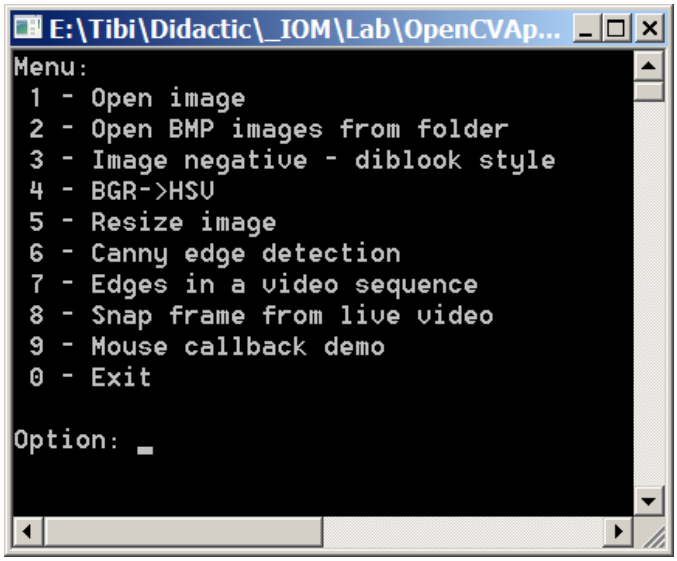

Fig. 1. Interfata cu utilizatorul.

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

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 cu metoda at
		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 compponentele 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 rgb = imread(fname);

		Mat hsvImg;
		Mat channels[3];
		cvtColor(rgb, hsv, CV_BGR2HSV);

		split(hsv, channels);

		// Componentele de culoare ale modelului HSV
		Mat H = channels[0]*255/180;
		Mat S = channels[1];
		Mat V = channels[1];

		imshow("input image", rgb);
		imshow("input image", hsv); // vizualizarea matricii hsv (ca un mat cu 3 canale) nu are semnificatie vizuala utila / relevanta

		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
		};
	}
}
```

Folosirea “plug-in”-ului “Image Watch”

Image Watch “plug-in” pentru Visual Studiu permite vizualizarea imaginilor (structuri de tip *Mat*) din memori in timp ce depanati o aplicatie. Folosirea plug-in-ului poate fi utila in gasirea erorilor sau la intelegerea unei anumite parti de cod prin vizualizarea continutului intermediar / final al imaginilor procesate.

Pentru deschiderea ferestrei “Image Watch” accesati din Visual Studio meniul “View” -> “Other Windows” -> “Image Watch”:

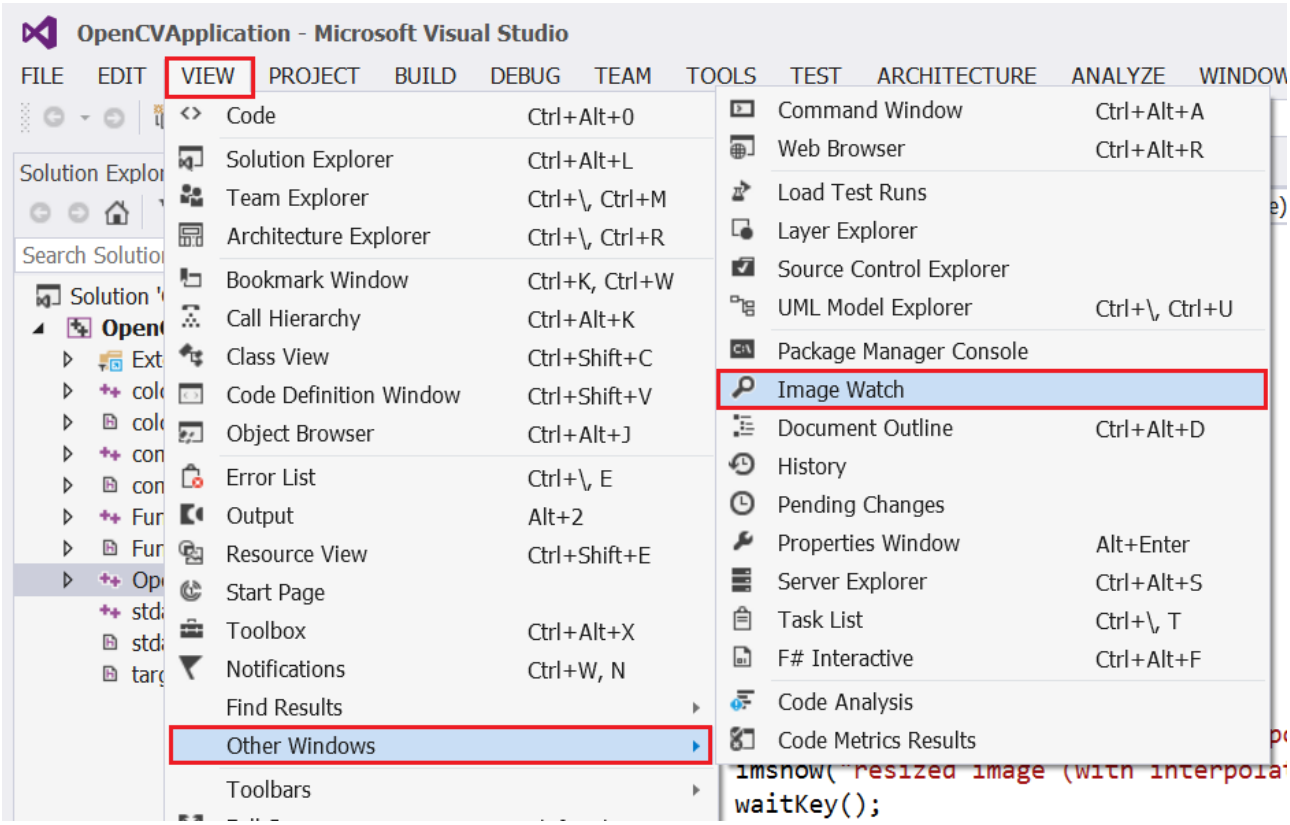

Fig. 2. Accesul la fereastra “Image Watch” in Visual Studio.

Pentru utilizare, inserati un breakpoint in functia pe care vreti sa o depanati. In fereastra “ImageWatch” veti putea vizualiza continutului imaginilor (variabilelor de tip Mat) aferente contextului local al functiei depanate.

Puteti face zoom-in/out pe o anumita zona din imagine (mouse scroll) si puteti vizualiza continutul canalelor imaginii la o anumita locatie (pixel): x y | c0 c1 … cN (canalele ci sunt afisate in ordinea in care apar in memorie). Mai multe informatii despre utilizarea Image Watch gasiti in referintele [5].

1.4. Mersul lucrarii

1. Implementati/testati functia `testBGR2HSV()`.

2. Adaugati o functie care afiseaza la linia de comanda coordonatele pe care ati facut click si valorile componentlor de culoare H,S,V ale pixelului pe care se face click pe imaginea sursa(rgb) (vezi `testBGR2HSV()`, `testMouseClick()`, `MyCallBackFunc()`). Verificati corectitudinea implementarii folosind utilitarul `ImageWatch`.

3. Adaugati o functie care afiseaza pe imaginea sursa (rgb) folosind functia `putText` valorile componentlor de culoare H,S,V ale pixelului peste care se deplaseaza mouse-ul (`CV_EVENT_MOUSEMOVE`) si valorile coordonatelorcurente (vezi `testBGR2HSV()`, `testMouseClick()`, `MyCallBackFunc()`).Verificati corectitudinea implementarii folosind utilitarul `ImageWatch`.

Observatii:
- deoarece functiei de callback puteti sa ii transmiteti un singur parametru (matricea hsv), va trebui sa declarati matrivea sursa (rgb) global (in afara corpului functiei);

- ca sa nu alterati sursa, la afisare (in if-ul aferent evenimentului mouse-move din functia de callback) faceti o copie/clona a imaginii sursa(rgb) intr-o alta matrice temp in care veti adauga textul: `Mat temp = rgb.clone();`

```
char msg[100];
sprintf(msg, "string fotmatare",valori);
putText(temp, msg, Point(5, 20), FONT_HERSHEY_SIMPLEX, 0.7, CV_RGB(255, 0, 0), 2, 8);
imshow("My Window", temp);
```

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
[4] Procesarea Imaginilor - Indrumator de laborator, http://users.utcluj.ro/~tmarita/IPL/IPLab/IPLAB.htm
[5] Image Watch: https://imagewatch.azurewebsites.net/ImageWatchHelp/ImageWatchHelp.htm , http://docs.opencv.org/2.4/doc/tutorials/introduction/windows_visual_studio_image_watch/windows_visual_studio_image_watch.html