3. Segmentarea imaginilor color (2): crearea unor modele de culoare si clasificarea pixelilor pe baza modelului

Scop: construirea unui model de culoare pentru pielea umana (in particular culoarea medie a pielii maini unui set de persoane), segmentarea imaginilor prin clasificarea pixelilor pe baza modelului construit, detectia mainii sub forma unui obiect unitar (se vor aplica postprocesari) si calculul proprietatilor geometrice simple ale obiectului obtinut.

3.1. Construirea modelului de culoare al mainii

Modelul de culoare pt. piele va fi o histograma globala (o distributie de probabilitate de forma cvasi-gaussiana). Parametrii modelului vor fi media si deviatia standard a acestei distributii (vezi L8/PI).

Aceasta histograma globala se va declara ca si o variabila globala (in OpenCVApplication.cpp):

```
#define MAX_HUE 256
//variabile globale
int histG_hue[MAX_HUE]; // histograma globala / cumulativa
```

3.1.1. Initializarea modelului

Se va adauga o functie pentru initializarea modelui (histogramei globale) prin setarea elementelor vectorului histogramei globale la 0:

```
void L3_ColorModel_Init()
{
	memset(histG_hue, 0, sizeof(unsigned int)*MAX_HUE);
}
```

3.1.2. Construirea modelului de culoare al mainii

Se va adauga o functie pentru construirea modelui (histogramei globale).

Se va considera un set de imagini relevante (luati imaginile cu mana folosite in L2). Pt. Fiecare imagine se vor considera/selecta regiuni de interes, in care se va calula o histograma locala a componetei de culoare Hue (normalizata in intervalul 0 ..255):

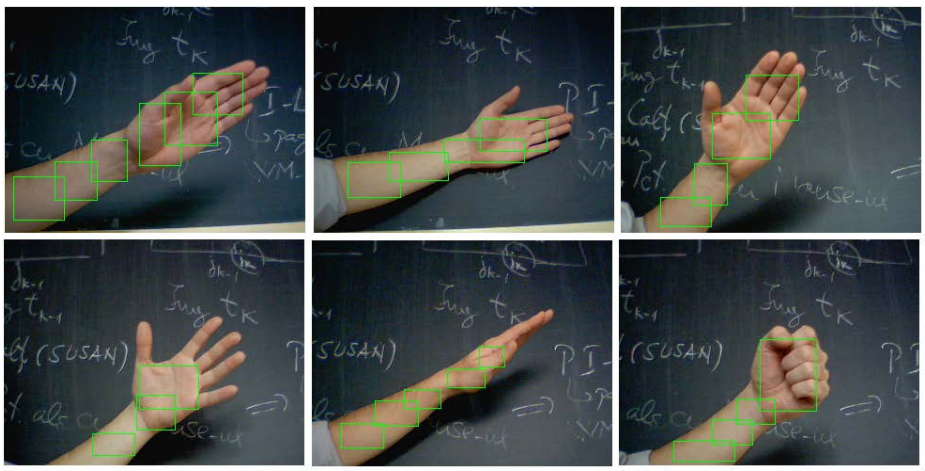

Fig. 3.1. Setul de imagini de antrenare folosit pentru construirea modelului

Selectia regiunilor de interes se va face manual cu ajutorul mouse-ului prin tratarea evenimentelor *LBUTTONDOWN* pentru selectia unui colt al regiunii de interes (`Pstart`) si RBUTTONDOWN pentru coltul diagonal (`Pend`) opus al regiunii de interes. Este important sa selectati intotdeuna perechi de puncte (`Pstart si Pend`) cu ajutorul apasarii successive: *LBUTTONDOWN* si RBUTTONDOWN.

```
Point Pstart, Pend; // Punctele/colturile aferente selectiei ROI curente (declarate
global)

void L3_ColorModel_Build()
{
	Mat src;
	Mat hsv;
	// Read image from file
	char fname[MAX_PATH];
	while (openFileDlg(fname))
	{
		src = imread(fname);
		int height = src.rows;
		int width = src.cols;

		// Aplicare FTJ gaussian pt. eliminare zgomote: essential sa il aplicati
		GaussianBlur(src, src, Size(5, 5), 0, 0);

		//Creare fereastra pt. afisare
		namedWindow("src", 1);

		// Componenta de culoare Hue a modelului HSV
		Mat H = Mat(height, width, CV_8UC1);

		// definire pointeri la matricea (8 biti/pixeli) folosita la stocarea
		// componentei individuale H
		uchar* lpH = H.data;

		cvtColor(src, hsv, CV_BGR2HSV); // conversie RGB -> HSV

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

		for (int i = 0; i<height; i++)
		{
			for (int j = 0; j<width; j++)
			{
				// index in matricea hsv (24 biti/pixel)
				int hi = i*width * 3 + j * 3;
				int gi = i*width + j; // index in matricea H (8 biti/pixel)
				lpH[gi] = hsvDataPtr[hi] * 510 / 360;  // lpH = 0 .. 255
			}
		}

		// Asociere functie de tratare a avenimentelor MOUSE cu ferestra curenta
		// Ultimul parametru este matricea H (valorile compunentei Hue)
		setMouseCallback("src", CallBackFuncL3, &H);

		imshow("src", src);

		// Wait until user press some key
		waitKey(0);
	}
```

```
}

void CallBackFuncL3(int event, int x, int y, int flags, void* userdata)
{
	Mat* H = (Mat*)userdata;
	Rect roi; // regiunea de interes curenta (ROI)

	if (event == EVENT_LBUTTONDOWN)
	{
		// punctul de start al ROI
		Pstart.x = x;
		Pstart.y = y;
		printf("Pstart: (%d, %d)  ", Pstart.x, Pstart.y);
	}
	else if (event == EVENT_RBUTTONDOWN)
	{
		// punctul de final (diametral opus) al ROI
		Pend.x = x;
		Pend.y = y;
		printf("Pend: (%d, %d)  ", Pend.x, Pend.y);
		// sortare puncte dupa x si y
		//(parametrii width si height ai structurii Rect > 0)
		roi.x = min(Pstart.x, Pend.x);
		roi.y = min(Pstart.y, Pend.y);
		roi.width = abs(Pstart.x - Pend.x);
		roi.height = abs(Pstart.y - Pend.y);
		printf("Local ROI: (%d, %d), (%d, %d)\n", roi.x, roi.y, roi.x + roi.width,
			roi.y + roi.height);

		int hist_hue[MAX_HUE];    // histograma locala a lui Hue
		memset(hist_hue, 0, MAX_HUE*sizeof(int));

		// Din toata imaginea H se selecteaza o subimagine (Hroi) aferenta ROI
		Mat Hroi = (*H)(roi);

		uchar hue;
		//construieste histograma locala aferente ROI
		for (int y = 0; y < roi.height; y++)
			for (int x = 0; x < roi.width; x++)
			{
				hue = Hroi.at<uchar>(y, x);
				hist_hue[hue]++;
			}
		//acumuleaza histograma locala in cea globala
		for (int i = 0; i < MAX_HUE; i++)
			histG_hue[i] += hist_hue[i];

		// afiseaza histohrama locala
		showHistogram("H local histogram", hist_hue, MAX_HUE, 200, true);
		// afiseaza histohrama globala / cumulativa
		showHistogram("H global histogram", histc_hue, MAX_HUE, 200, true);
	}
}
```

3.1.2. Postprocesarea modelului de culoare al mainii si calcularea parametrilor modelului de culoare al mainii

Se va adauga o functie pentru postprocesarea modelului, calculul parametrilor modelului (media si deviatia standard) si salvarea datelor intr-un fisier text.

```
void L3_ColorModel_Save()
{
	int hue, sat, i, j;
	int histF_hue[MAX_HUE]; // histograma filtrata cu FTJ
	memset(histF_hue, 0, MAX_HUE*sizeof(unsigned int));
```

Optional, se pot filtra elementele histogramei cumulative (modelului) cu un filtru trece jos (FTJ) de tip medie aritmetica sau gaussian.

```
	//Filtrare histograma Hue (optional)
#define FILTER_HISTOGRAM 1

#if FILTER_HISTOGRAM == 1
	// filtrare histograma cu filtru gaussian 1D de dimensiune w=7
	float gauss[7];
	float sqrt2pi = sqrtf(2 * PI);
	float sigma = 1.5;
	float e = 2.718;
	float sum = 0;
	// Construire gaussian
	for (i = 0; i<7; i++) {
		gauss[i] = 1.0 / (sqrt2pi*sigma)* powf(e, -(float)(i - 3)*(i - 3)
				/ (2 * sigma*sigma));
		sum += gauss[i];
	}
	// Filtrare cu gaussian
	for (j = 3; j<MAX_HUE - 3; j++)
	{
		for (i = 0; i<7; i++)
			histF_hue[j] += (float)histc_hue[j + i - 3] * gauss[i];
	}
#elif
	for (j = 0; j<MAX_HUE; j++)
		histF_hue[j] = histc_hue[j];

#endif // End of "Filtrare Gaussiana Histograma Hue"
```

De asemenea, **este recomandat** sa filtrati valorile din histograma model care sunt mai mici decat x% (x% = 1 ..10 %) din valoarea maxima a histogramei (pt. eliminarea eventualelor „zgomote” datorate selectiei imprecise, daca se selecteaza si zone de fond din afara perimetrului mainii).

Forma histogramei cumulative obtinute va fi asemanatoare unei distributii gaussiene. Pentru aceasta distributie se vor calcula media si deviatia standard (parametrii modelului) – vezi L8/PI:

$$\bar{g} = \mu = \int_{-\infty}^{+\infty} g \cdot p(g) dg = \sum_{g=0}^{L} g \cdot p(g) = \frac{1}{M} \sum_{g=0}^{L} g \cdot h(g)$$

$$\sigma = \sqrt{\sum_{g=0}^{L} (g - \mu)^2 \cdot p(g)}$$

Unde:

g = hue = 0 .. L

L = `MAX_HUE`

M = numarul de elemente din histograma model: `M += histF_hue[j]; j= 0.. MAX_HUE`

Afisarea histogramelor globale (nefiltrata si filtrata):

```
	showHistogram("H global histogram", histc_hue, MAX_HUE, 200, true);
	showHistogram("H global filtered histogram", histF_hue, MAX_HUE, 200, true);

	// Wait until user press some key
	waitKey(0);
} //end of L3_ColorModel_Save()
```

Pentru setul de imagini de antrenare (Fig. 3.1) modelul arata ca si in figura de mai jos, cu parametri aproximativi:

```
hue_mean = 16
hue_std = 5
```

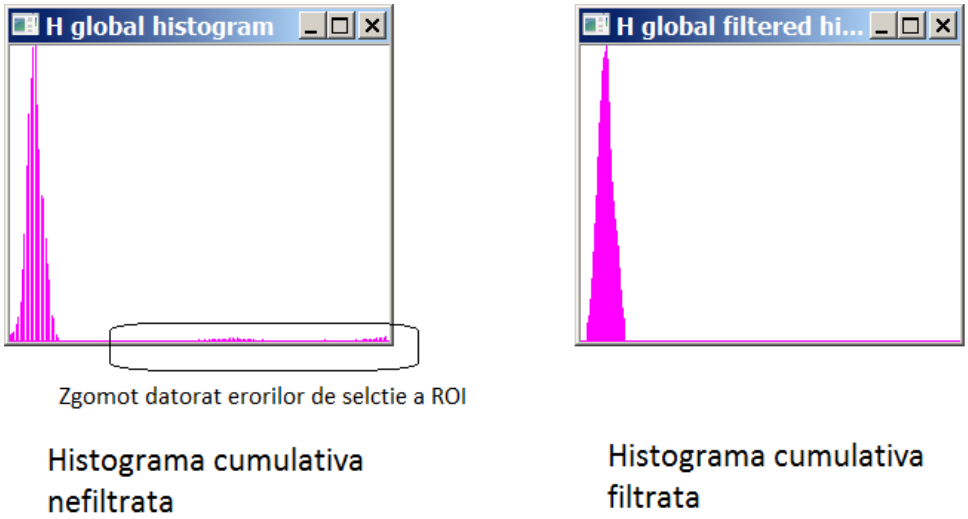

Fig. 3.2. Modelului de culoare al mainii: hisograma globala nefiltrata (stanga) si filtrata (dreapta)

Parametrii modelului (histograma modelului, media si deviatia standard) se pot salva optional intr-un fisier text (pentru verificare/debug):

```
// pregatire pt. scriere valoari model in fisier
	FILE *fp;

	// Hue
	fp = fopen("E:\\Hue.txt", "wt");

	fprintf(fp, "H=[\n");

	for (hue = 0; hue<MAX_HUE; hue++){
		fprintf(fp, "%d\n", histF_hue[hue]);
	}

	fprintf(fp, "];\n");
	fprintf(fp, "Hmean = %.0f ;\n", hue_mean);
	fprintf(fp, "Hstd = %.0f ;\n", hue_std);

	fclose(fp);
```

3.2. Segmentarea imaginii / mainii

3.2.1. Clasificarea pixelilor din imagine

Aceasta etapa consta in clasificarea pixelilor din imagine pe baza valorii curente a componentei Hue:

- Daca valoarea lui hue este in intervalul: `[hue_mean-k*hue_std .. hue_mean+k*hue_std]`, atunci pixelul respectiv se clasifica/eticheteaza ca fiind pixel de tip OBIECT.
- Altfel se clasifica/eticheteaza ca fiind de tip FOND.

Unde:

k = 2 .. 3

Adaugati o noua functie de procesare in care sa implementati operatia de segmentare. Puteti folosi sablonul prezentat in `L3_ColorModel_Build()` - renuntati la apelul `setMouseCallback`. Aplicati operatia de clasificare a pixelilor pe matricea H (component Hue). Puneti rezultatul (imaginea alb/negru) intr-o matrice destinatie (8biti/pixel). Afisati imaginea destinatie intr-o fereastra noua.

Observatie: verificati ca intervalul `[hue_mean-k*hue_std .. hue_mean+k*hue_std]`, sa nu depaseasca intervalul pe care ati definit / scalat valorile lui hue: `[0 .. MAX_HUE]`. Adaugati o conditie de limitare.

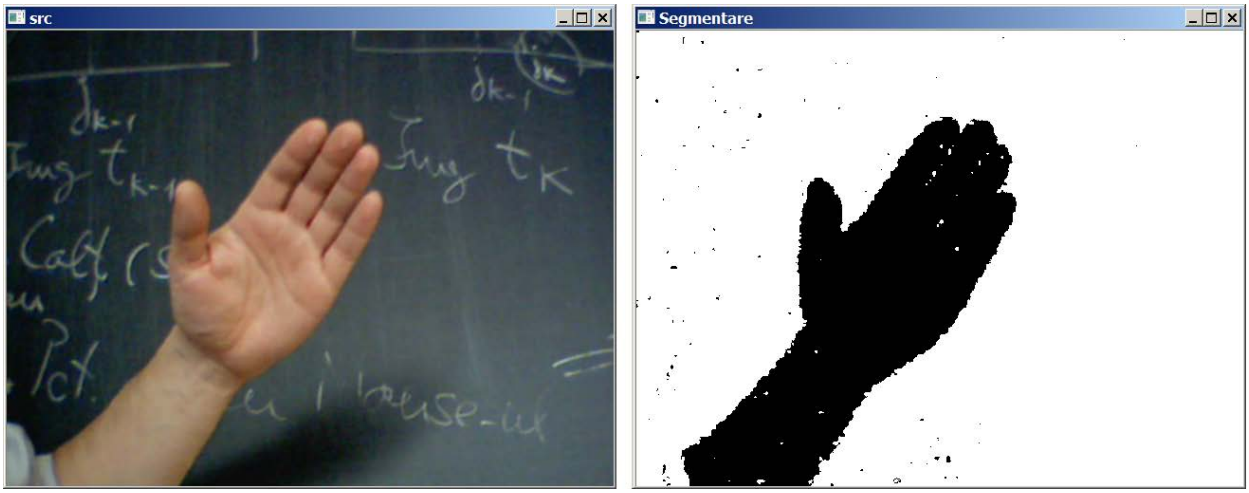

Fig. 3.3. Rezultatul segmentarii pt. k=2.5 in care pixelii OBIECT sunt desenati cu negru si cei de FOND cu alb.

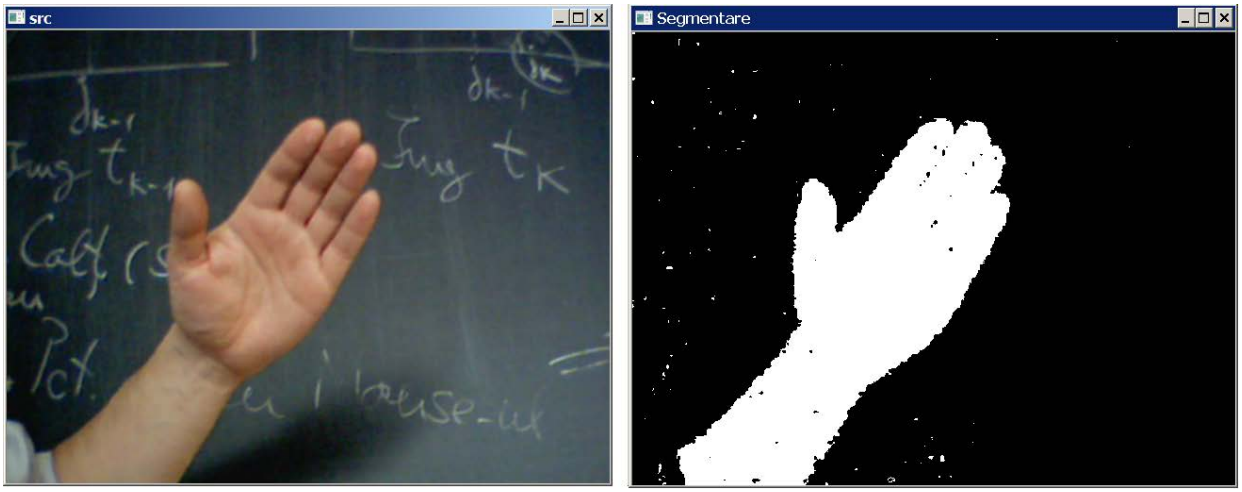

Fig. 3.4. Rezultatul segmentarii pt. k=2.5 in care pixelii OBIECT sunt desenati cu alb si cei de FOND cu negru (conventia din OpenCV).

3.2.2. Postprocesarea imaginii segmentate

Imaginea segmentata (ex. Fig. 3.3) poate prezenta zgomot. O metoda simpla de postprocesare consta in eliminarea acestor zgomote (pixeli albi in interiorul obiectului si pixeli negrii in zona de fond prin operatii morfologice (dilatare, eroziune) aplicate repetate / succesiv.

Pt. operatiile morfologice folositi functiile implementate in OpenCV. **Atentie:** operatiile sunt implementate in logica inversa (fata de cum ati fost obisnuiti la laboratorul de PI): dilatarea si eroziunea in OpenCV se fac pe OBIECT =„alb” (FOND=”negru”).

Exemple de utilizare:

```
// creare element structural de dimensiune 5x5 de tip cruce
Mat element1 = getStructuringElement(MORPH_CROSS, Size(5, 5));
//eroziune cu acest element structural (aplicata 1x)
erode(dst, dst, element1, Point(-1, -1), 1);

// creare element structural de dimensiune 3x3 de tip patrat (V8)
Mat element2 = getStructuringElement(MORPH_RECT, Size(3, 3));
// dilatare cu acest element structural (aplicata 2x)
dilate(dst, dst, element2, Point(-1, -1), 2);
```

Obs: pentru eliminarea zgomotului definiti/alegeti un singur tip element structural. Eliminati intai zgomotul din zonele de fond. Daca in obiectul de interes apar goluri umpleti aceste goluri. Ca sa nu modificati aria obiectului de interes trebuie ca numarul de eroziuni sa fie egal cu numarul de dilatari (cu acelasi tip de element structural) - vezi exemplul din figura de mai jos:

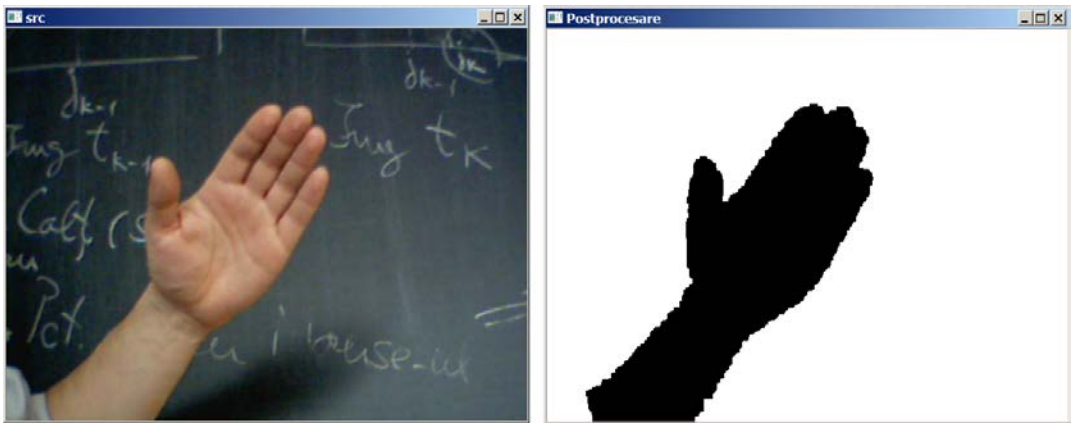

Fig. 3.5. Rezultatul dupa postprocesare cu operatii morfologice cu element structural patrat de 3x3: 2 x dilatari + 4x eroziuni + 2x dilatari (pixelii de obiect sunt negri).

Fig. 3.6. Rezultatul dupa postprocesare cu operatii morfologice cu element structural patrat de 3x3: 2 x eroziuni + 4x dilatari + 2x eroziuni (pixelii de obiect sunt albi: conventia OpenCV).

3.2.3. Extragerea conturului mainii

Dupa eliminarea zgomotelor (se presupune ca in imagine a ramas un singur obiect) se pot extrage anumite proprietati geometrice ale obiectului / mainii (aria, CM, axa de alungire), se pot extrage pixelii de contur etc. (vezi L4, L6 / PI).

Obs: in cazul in care ati optat pt. conventia *„pixelii OBIECT desenati cu negru si cei de FOND cu alb”*, codul pentru calculul proprietatilor geometrice si detectie a conturului pe care l-ati scris la laboratoarele de procesarea imaginilor (L4 si L6) ar trebui sa il puteti folosi/integra foarte simplu.

Ca si alternativa la operatiile de calculul a proprietatilor geometrice si detectie a conturului puteti folosi functiile din OpenCV. Un exemplu il gasiti in functia `Labeling` definita in modulul *Functions.cpp*. Pentru a utiliza functiile din OpenCV trebuie sa utilizati conventia *„pixelii OBIECT desenati cu alb si pixeli de FOND cu negru”* atunci cand faceti operatiile de segmentare si postprocesare (operatii morfologice).

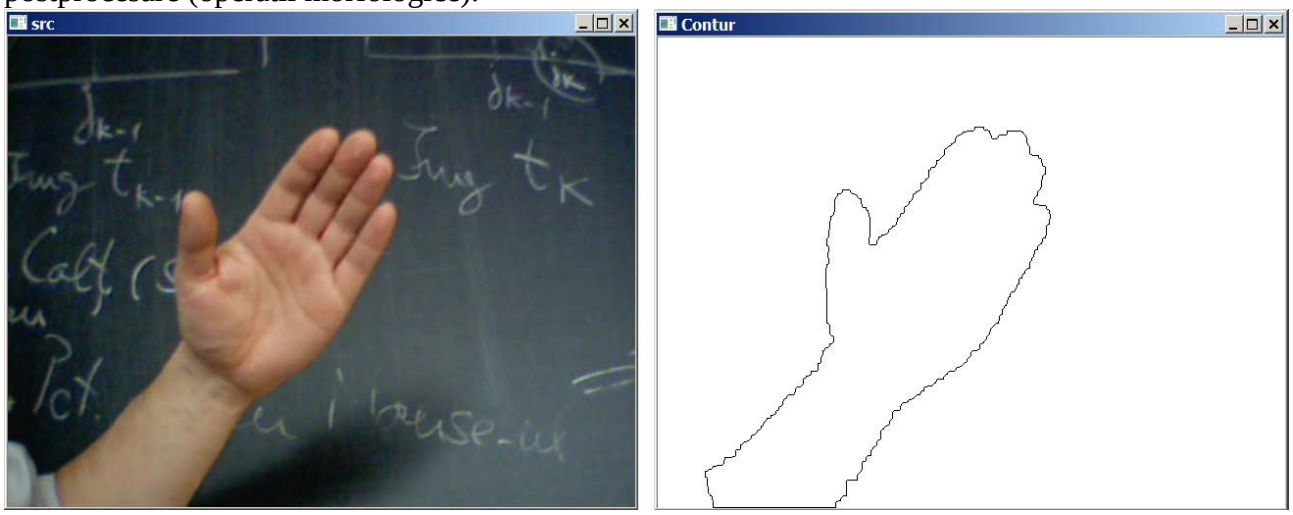

Fig. 3.5. Rezultatul dupa extragerea conturului *(„conventia pixelii OBIECT desenati cu negru si cei de FOND cu alb”)*- implementare algoritm L6/PI.

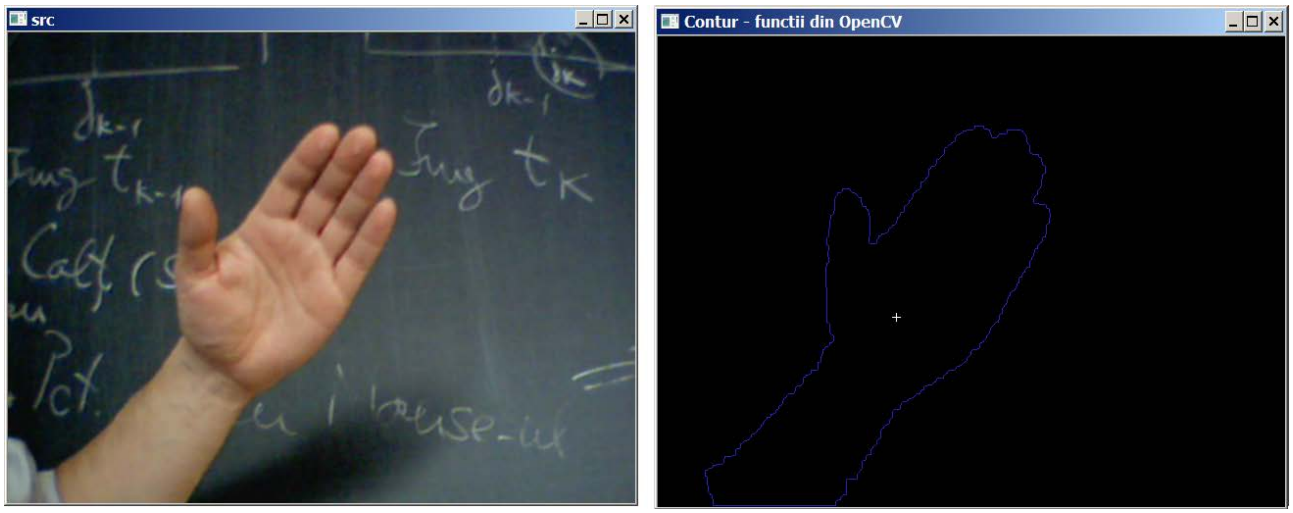

Fig. 3.6. Rezultatul dupa extragerea conturului folosind functii din OpenCV.

3.2.4. Desenarea axei de alungire a mainii

Pentru a desena axa de alungire a mainii puteti folosi functia din OpenCV:

```
line(img, P1, P2, color, thickness, 8);
```

Trebuie doar sa calculati 2 puncte intre care desenati linia. Pt. cazul de fata aceste 2 puncte ar trebui sa fie cele 2 puncte de intersectie ale axei de alungire cu marginile imaginii (*intercepts*) care se afla in intervalele:

x= [0 .. width-1] si y = [0 .. height-1] (exista doar 2 astfel de puncte !!!).

Pt. a calcula punctele *intercepts* scrieti ecuatia axei de alungire obtinute sub forma:

$$y-y1 = slope * (x-x1) \quad (1)$$

unde: *(x1,y1)* - coordonatele centrului de masa ale obiectului
slope = tan(teta), teta este unghiul axei de alungire [radiani]

1. Inlocuiti *x* cu *0* respectiv *Width-1* si *y* cu *0* respectiv *Height-1* in ec. (1) si gasiti coordonatele celor 4 puncte *intercepts*.
2. Selectati acele puncte (*intercepts*) care se afla in intervalele: *x= [0 .. width-1]* si *y = [0 .. height-1]* (exista doar 2 astfel de puncte).
3. Desenati linia intre aceste 2 puncte.

3.3. Mersul lucrării

1. Presupunand modelului de culoare deja construit pentru setul de imagini dat (vezi L2):

```
hue_mean = 16
hue_std = 5
```

se va adauga o functie de procesare care sa integreze urmatoarii pasi:

a. Segmentarea imaginilor prin clasificarea la nivel de pixel (3.2.1). Se va alege o valoare optima pt. k, pentru fiecare imagine in parte din setul dat.
b. Postprocesarea imaginii segmentate pentru eliminarea zgomotelor folosind filtre morfologice (3.2.2).
c. Extrage contrurul mainii segmentate si calculul proprietatilor geometrice ale obiectului segmentat (folositi implementarea proprie din L4&L6/PI sau integrati functia `Labeling` din modulul *Functions* (3.2.3).
d. Desenarea axei de alungire a mainii peste imaginea finala (3.2.4).

Nota: Afisati rezultatele intermediare dupa fiecare etapa de procesare pentru a valida corectitudinea rezultatelor (nu treceti la pasul urmator pana cand nu l-ati validat pe cel precedent !!!).

Bibliografie

[1] OpenCV, On-line reference manual and tutorials: Erosion & Dilation,
http://docs.opencv.org/2.4/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html
http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=dilate#dilate
http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=erode#erode
[2] OpenCV, On-line reference manual and tutorials: Find Contours,
http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=findcontours#findcontours
[3] OpenCV, On-line reference manual and tutorials: Draw Contours,
http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=drawcontours#drawcontours
[4] OpenCV, On-line reference manual and tutorials: Moments,
http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=moments#moments