2. Segmentarea imaginilor color (2): crearea unui model 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.

2.1. Construirea modelului de culoare al mainii

Modelul de culoare pt. piele va fi o histograma globala (o distributie de probabilitate de forma cvasi-gaussiana) calculata din multiple regiuni ale mainilor vizibile in setul de imagini de antrenare. Parametrii modelului vor fi media si deviatia standard a acestei distributii [1].

```
#define MAX_HUE 256
// variabile globale
int histc_hue[MAX_HUE]; // histograma cumulativa (in care se aduna valorile histogramelor locale)
```

2.1.1. Initializarea modelului

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

```
void L2_ColorModel_Init()
{
	memset(histc_hue, 0, sizeof(unsigned int) * MAX_HUE);
	printf("Initialize / reset the Global Hue histogram\n");
	waitKey(2000);
}
```

2.1.2. Construirea modelului de culoare al mainii

Se va adauga o functie pentru construirea modelui (histogramei globale). Se va considera un set de imagini relevante (folositi imaginile cu mana folosite si in laboratorul precedent). 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):

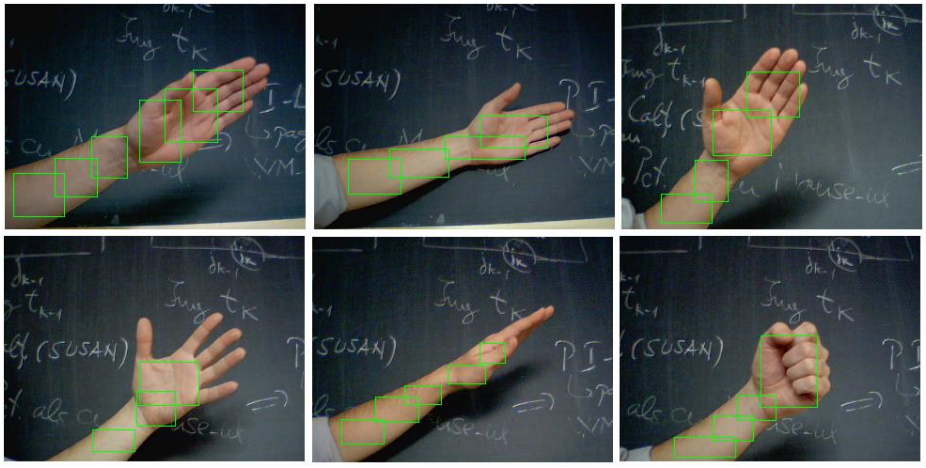

Fig. 2.1. Setul de imagini de antrenare folosit pentru construirea modelului

Selectia regiunilor de interes se va face manual cu ajutorul mouse-ului prin tratarea evenimentelor `EVENT_LBUTTONDOWN` pentru selectia unui punctului de start al unei regiunii de interes rectangulare (`Pstart`), `EVENT_MOUSEMOVE` pentru selectarea dinamica a regiunii selectate (pentru implementarea actiunii/efectului de mouse-dragging [2] si `EVENT_LBUTTONUP` pentru selectia punctului final (`Pend`) al regiunii de interes rectangulare.

```
Point Pstart, Pend; // Punctele/colturile aferente selectiei ROI curente (declarate
global)
bool draw = false;  // Variabila globala care arata starea actiune de mouse dragging:
true - in derulare

//Functia CallBack care se apeleaza la declansarea evenimentelor de nmouse
void CallBackFuncL2(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;
		draw = true;
		printf("Pstart: (%d, %d)  ", Pstart.x, Pstart.y);
	}
	else if (event == EVENT_MOUSEMOVE)
	{
		if (draw == true) // actiune de mouse dragging in derulare (activ)
		{
			// desenarea  se face intr-o copie a matricii sursa
			Mat temp = srcg.clone();
			rectangle(temp, Pstart, Point(x, y), Scalar(0, 255, 0), 1, 8, 0);
			imshow("src", temp);
		}
	}
	else if (event == EVENT_LBUTTONUP)
	{
		// punctul de final (diametral opus) al ROI rectangulare
		draw = false; // actiune de mouse dragging s-a terminat (inactiva)
		Pend.x = x;
		Pend.y = y;
		printf("Pend: (%d, %d)  ", Pend.x, Pend.y);
		// sortare crescatoare a celor doua puncta selectate dupa x si y
		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);
		rectangle(srcg, roi, Scalar(0, 255, 0), 1, 8, 0);
		// desenarea selectiei rectangulare se face peste imaginea sursa
		imshow("src", srcg);

		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]++;
				}

			int countg = 0, countl = 0;
			//acumuleaza histograma locala in cea globala
			for (int i = 0; i < MAX_HUE; i++) {
				histc_hue[i] += hist_hue[i];
				countg += histc_hue[i];
				countl += hist_hue[i];
			}
			printf("Histogram count (global / local) = %d / %d \n", countg, countl);
			// afiseaza histohrama locala
			showHistogram("H local histogram", hist_hue, MAX_HUE, 200, true);
			// afiseaza histohrama globala
			showHistogram("H global histogram", histc_hue, MAX_HUE, 200, true);
		}
}

// Functia principala in care se citesc imaginile sursa si se face legatura intre
// functia de mouse CallBack si fereastra in caare este afisata imaginea sursa
void L2_ColorModel_Build()
{
		//Mat srcg; declarata global
		Mat hsv;
		// Citestea imaginea din fisier
		char fname[MAX_PATH];
		while (openFileDlg(fname))
		{
			srcg = imread(fname);
			int height = srcg.rows;
			int width = srcg.cols;

			// Aplicare FTJ gaussian pt. eliminare zgomote / netezire imagine
		// http://opencvexamples.blogspot.com/2013/10/applying-gaussian-filter.html
			GaussianBlur(srcg, srcg, Size(5, 5), 0, 0);

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

			// Componenta de culoare Hue a modelului HSV
			Mat channels[3];

			cvtColor(srcg, hsv, COLOR_BGR2HSV); // conversie RGB -> HSV

			split(hsv, channels);

			// Componentele de culoare ale modelului HSV
			Mat H = channels[0] * 255 / 180;
			Mat S = channels[1];
			Mat V = channels[2];

			// asociere functie de tratare a evenimentelor MOUSE cu ferestra "src"
			// Ultimul parametru al functiei este adresa lamatricea H
			setMouseCallback("src", CallBackFuncL2, &H);
```

```
            imshow("src", srcg);

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

2.1.3. 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 L2_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 fundal 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 [1]:

$$\bar{g} = \mu = \int_{-\infty}^{+\infty} g \cdot p(g) dg = \sum_{g=0}^{L} g \cdot p(g) = \frac{1}{M} \sum_{g=0}^{L} g \cdot h(g) \quad (2.1)$$

$$\sigma = \sqrt{\sum_{g=0}^{L} (g - \mu)^2 \cdot p(g)} \quad (2.2)$$

Unde:

g = hue
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 L2_ColorModel_Save()
```

Pentru setul de imagini de antrenare (Fig. 2.1) modelul arata ca si in figura de mai jos, cu parametri aproximativi:

```
hue_mean = 17
hue_std = 5
```

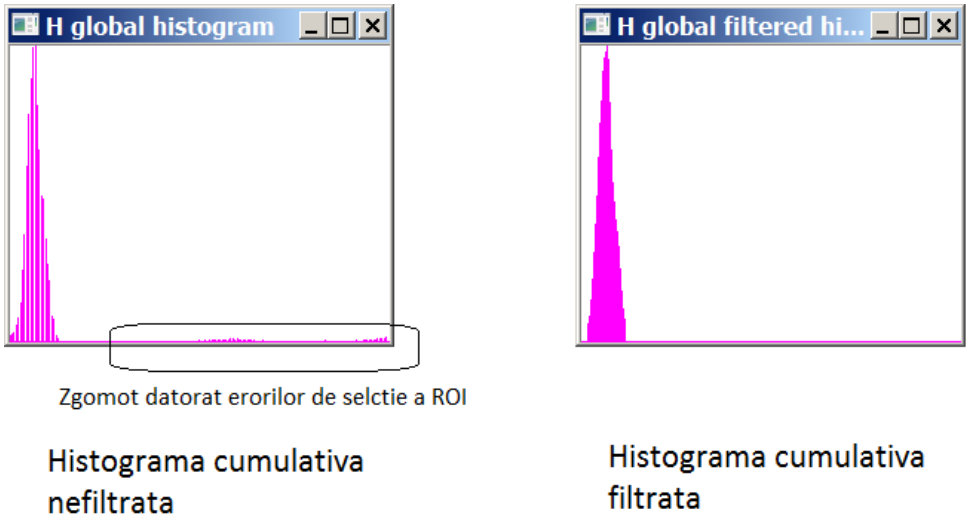

Fig. 2.2. Modelului de culoare al mainii: hisograma globala nefiltrata (stanga) si filtrata (dreapta)

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

```
// pregatire pt. scriere valoari model in fisier
	FILE *fp;
	// Hue
	fp = fopen("D:\\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);
```

2.2. Segmentarea imaginii / mainii

2.2.1. 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 FUNDAL.

Unde: k este o valoare cuprinsa intre 2 .. 3 (doarece latime unei distributii gaussiene este de aproximativ `6*hue_std`)

Pentru implementarea operatia de segmentare, se poate folosi sablonul prezentat in `L3_ColorModel_Build()` din care se elimina 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.

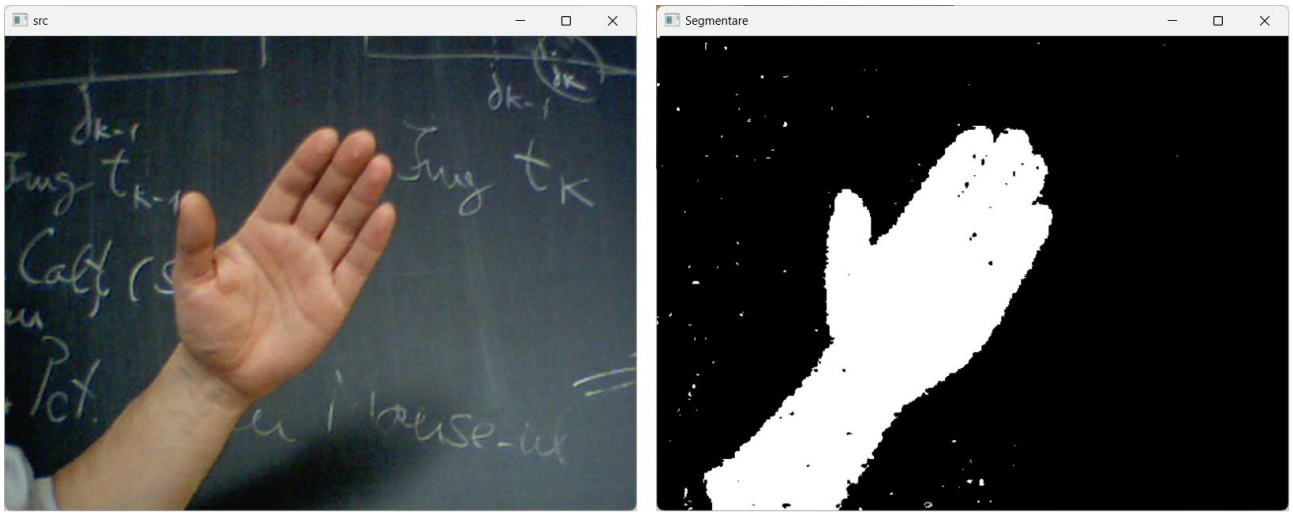

Fig. 2.3. Rezultatul segmentarii pt. hue_mean = 17, hue_std = 5, k=2.5. Pixelii OBIECT sunt desenati cu alb si cei de FUNDAL cu negru (conventia din OpenCV).

2.2.2. Postprocesarea imaginii segmentate

Imaginea segmentata (ex. Fig. 2.3) poate prezenta erori de segmentare: pixeli de obiect clasificati gresit ca fiind pixeli de fundal si invers. O metoda simpla de postprocesare consta in eliminarea acestor zgomote (pixeli albi in interiorul obiectului si pixeli negrii in zona de fundal) prin operatii morfologice (dilatare, eroziune) aplicate in mod repetat / succesiv.

Pentru operatiile morfologice se vor folosi functiile implementate in OpenCV. **Atentie:** operatiile sunt implementate in logica inversa (fata de cum ati fost obisnuiti la laboratorul de Procesarea Imaginilor): dilatarea si eroziunea in OpenCV se fac folosind conventia: pixel OBIECT =„alb” respectiv: pixel FUNDAL=”negru”.

Exemple de utilizare:

```
// crearea unui 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);

// crearea unui 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 (forma si dimensiune). Eliminati intai zgomotul din zonele de fundal (prin operatii de eroziune). Daca in obiectul de interes au ramas sau sa-au marit 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:

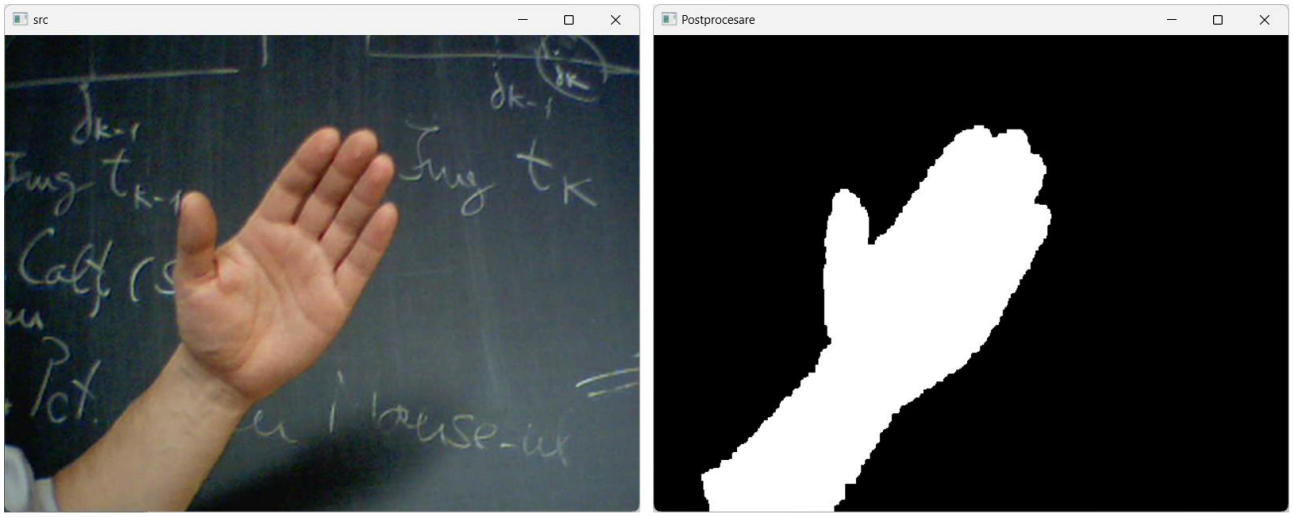

Fig. 2.4. Rezultatul dupa postprocesarea cu operatii morfologice cu un element structural patrat de dimensiune 3x3, aplicat in mod repetat: 2 x eroziuni + 4x dilatari + 2x eroziuni.

2.2.3. Calculul proprietatilor geometrice si extragerea conturului obiectului segmentat

Daca nu s-au putut elimina toate erorile de segmentare prin aplicarea postprocesarilor morfologice (au ramas zone mici reziduale de pixeli obiect in zona de fundal) se poate face o filtrare suplimentara a acestora pe baza de arie in functia `Labeling` (definita in modulul *Functions.cpp*), inlocuind constanta „0” cu o valoare mai mare decat aria celui mai mare obiect segmentat eronat:

```
void Labeling(const string& name, const Mat& src, bool output_format)
{
. . .
      if (arie > 0)
      {
      double xc = m.m10 / m.m00; // coordonata x a CM al componentei conexe idx
      double yc = m.m01 / m.m00; // coordonata y a CM al componentei conexe idx
```

```
. . .
}
```

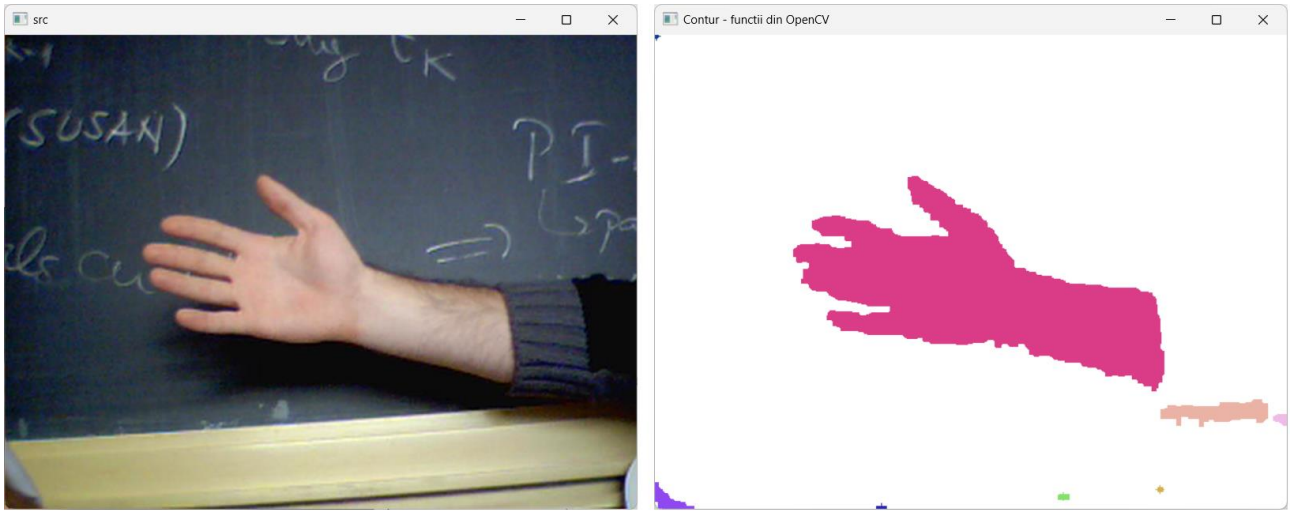

Fig. 2.4. Rezultatul dupa etichetarea obiectelor folosind functii din OpenCV (functia Labeling, cu parametrul `output_format = true` – afisare obiecte/etichete pline) in cazul unei segmentari imperfecte in care au ramas zone mici reziduale de pixeli obiect in zona de fundal.

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) si se pot extrage pixelii de contur [1]. 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 apelati functi d eprocesare pe imagini alb-negru implementate in OpeCV.

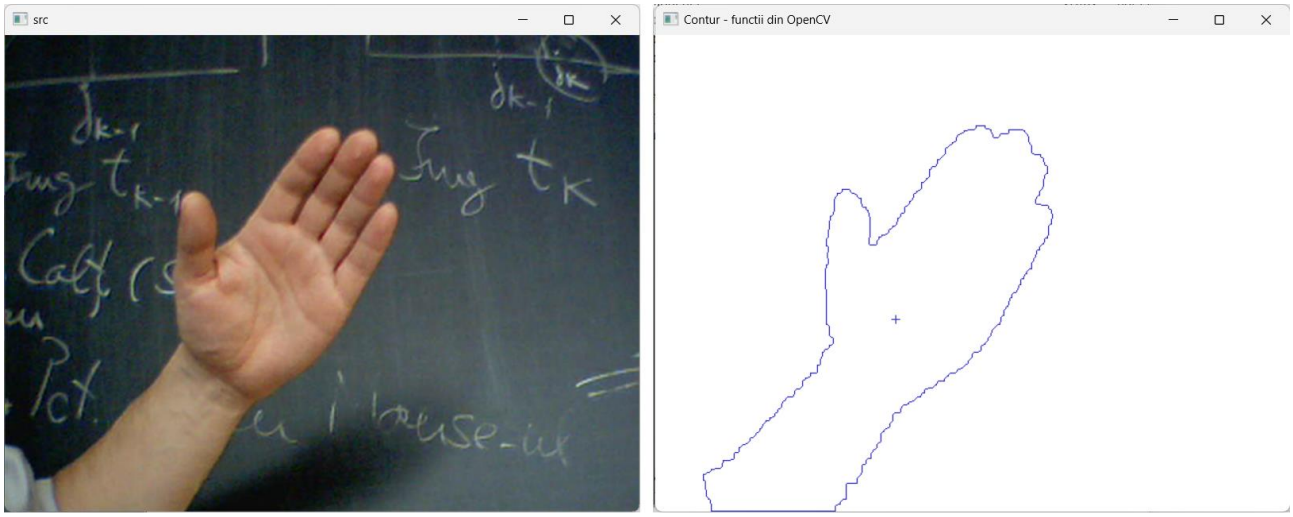

Fig. 2.5. Rezultatul dupa extragerea conturului si detectia centrului de masa folosind functii din OpenCV (functia Labeling, cu parametrul `output_format = false` – afisare obiecte prin contur)

2.2.4. 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 !!!).

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

$$y\text{-}yc = slope * (x\text{-}xc) \tag{2.3}$$

unde: *(xc,yc)* - coordonatele centrului de masa al obiectului
slope = tan(teta), teta este unghiul axei de alungire [radiani]

Algoritm:

1. Inlocuiti *x* cu *0* respectiv *Width-1* iar pentru fiecare *x*, inlocuiti *y* cu *0* respectiv *Height-1* in ecuatia (2.3) 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.

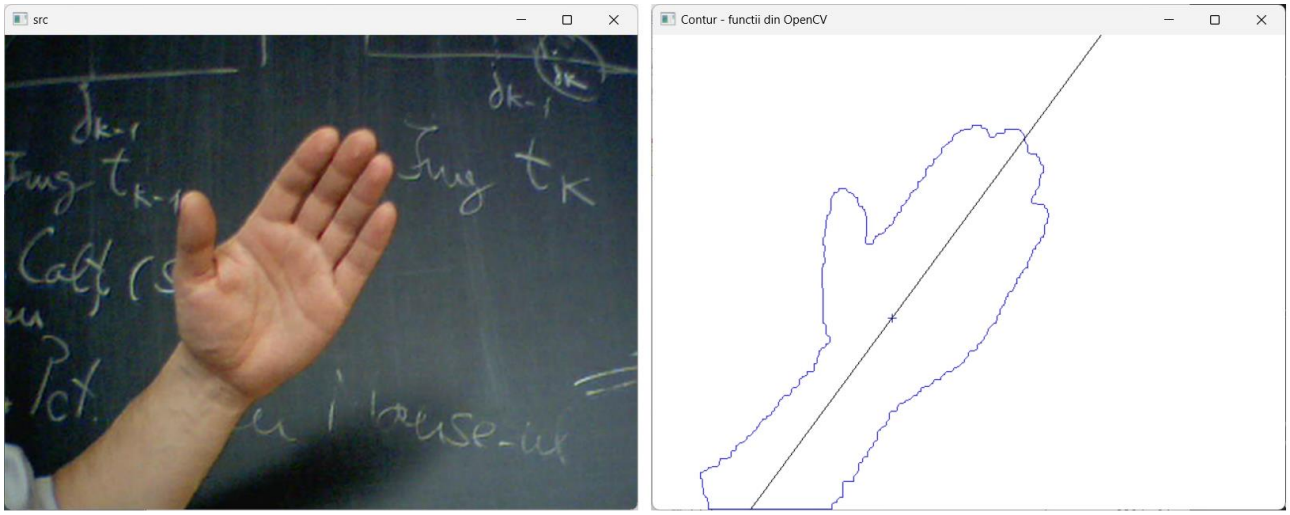

Fig. 2.6. Rezultatul dupa extragerea conturului, afisarea centrului de masa si a axai de alungire calculate.

2.3. Mersul lucrării

Presupunand modelului de culoare deja construit pentru setul de imagini dat:

```
hue_mean = 17
hue_std = 5
```

se va adauga o functie de procesare care sa integreze urmatoarii pasi d eprocesare descrisi in capitolul 2.2:

a. Segmentarea imaginilor prin clasificarea la nivel de pixel (2.2.1). Se va alege o valoare optima pt. k, pentru fiecare imagine in parte din setul dat.
   Observatie: este util sa filtrati imaginea sursa inainte de a aplica segmentarea cu un filtru trece jos: `GaussianBlur(src, src, Size(5, 5), 0, 0);`

b. Postprocesarea imaginii segmentate pentru eliminarea zgomotelor folosind filtre morfologice (2.2.2).
c. Extrage contrurul mainii segmentate si calculul proprietatilor geometrice ale obiectului segmentat (integrati o copie a functia `Labeling` din modulul *Functions* (2.2.3).
d. Desenarea axei de alungire a mainii peste imaginea finala obtinuta la punctul c. (2.2.4).

2.4. Bibliografie

[1] S. Nedevschi, T. Marita, R. Danescu, F. Oniga, R. Brehar, I. Giosan, C. Vancea, R. Varga, Procesarea Imaginilor - Îndrumător de laborator, editia a 2-a, Editura U.T. Press, Cluj-Napoca, https://biblioteca.utcluj.ro/carti-online-cu-coperta.html, 2023.

[2[Open Computer vision Library, Reference guide, Mouse as a Paint-Brush, https://docs.opencv.org/4.x/db/d5b/tutorial_py_mouse_handling.html
[3] OpenCV, On-line reference manual and tutorials: Erosion & Dilation, https://docs.opencv.org/4.9.0/db/df6/tutorial_erosion_dilatation.html
[4] OpenCV, On-line reference manual and tutorials: Find Contours and Draw Contours https://docs.opencv.org/4.9.0/df/d0d/tutorial_find_contours.html, https://docs.opencv.org/4.9.0/d3/dc0/group__imgproc__shape.html#gadf1ad6a0b82947fa1fe3c3d497f260e0
https://docs.opencv.org/4.9.0/d6/d6e/group__imgproc__draw.html#ga746c0625f1781f1ffc9056259103edbc
[5] OpenCV, On-line reference manual and tutorials: Moments, https://docs.opencv.org/4.9.0/d8/d23/classcv_1_1Moments.html, https://docs.opencv.org/4.9.0/d0/d49/tutorial_moments.html