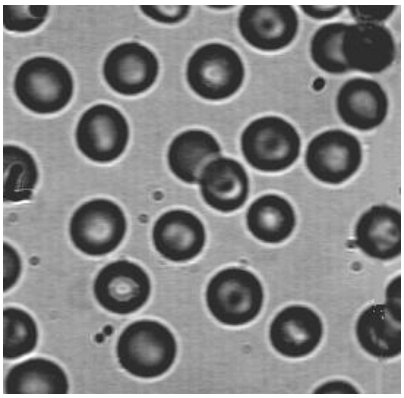
Documentatie proiect PI

Scuturici Vlad, grupa 30235

15. Implementati o metoda de detecţie a ariei, centrului de masă şi a diametrului mediu pentru fiecare celulă (obiect) din imaginea alăturată. Metoda trebuie să fie complet automată. Excludeti din analiza eventualele zgomote din imagine (ex. obiecte mici care nu sunt celule).



Specificatii detaliate

- **Specificarea formatului datelor de intrare si de iesire**

Intrare: Imaginea cu celulele

Ieșire: O listă de obiecte detectate, fiecare cu următoarele proprietăți: Arie,

Centru de masă, Diametrul.

- **Prezentarea temei, formulată exact, cu obiective clare și eventuale figuri explicative**

Detectarea automată a obiectelor în imagine, cu excluderea zgomotului și a obiectelor mici.

Calcularea metricilor specificați (arie, centru de masă, diametru) pentru fiecare obiect detectat.

- **Detalierea cerintelor functionale ale aplicatiei**

Aplicația trebuie să încarce și să proceseze imagini, aplicand o binarizare cu prag automat si filtre pentru reducerea zgomotului, apoi, va umple golurile lasate de primii 2 pasi folosing un algoritm BFS. Pasul urmator consta in selectarea celulelor, si adaugarea lor intr-o structura. Pe final, aplicația va calcula aria, centrul de masă și diametrul mediu.

Analiză și fundamentare teoretică

- **algoritmi utilizați și/sau propuși**

Pentru binarizare: Binarizarea cu prag automat implementata la laboratorul 8

Pentru eliminarea zgomotului: Filtru median

Pentru umplere spatii goale: BFS

Pentru calculare metricilor: Algoritmii utilizati la laborator

- **explicații/argumentări logice ale soluției alese**

In proiectul de detectare automata a obiectelor din imagini, am ales binarizarea cu prag automat pentru eficienta sa in separarea obiectelor de fundal, ceea ce faciliteaza procesul de segmentare. Folosim filtrul median pentru eliminarea zgomotului datorita eficacitatii sale in pastrarea detaliilor esentiale ale obiectelor, in timp ce algoritmul BFS ajuta la umplerea spatiilor goale, asigurand o detectie completa. Aceste metode sunt integrate pentru a optimiza precizia in calcularea ariei, centrului de masa si diametrului fiecarui obiect detectat.

- **structura logică și funcțională a aplicației.**

Aplicatia va fi impartita in module, fiecare avand scopul implementarii unei functionalitati necesare pentru implementarea algoritmilor. Modulele variaza de la operatii de baza cum ar fi deschiderea si inchiderea, la filtrul median care imbina aceste operatii, dar si module mai complexe cum ar fi cel de umplere a golurilor folosind BFS.

Analiză și fundamentare teoretică

- **schema generală aplicației**
- **descriere a fiecărei componente implementate, la nivel de modul, cu exemple de cod**

Aplicația de detectare a obiectelor din imagini este structurată modular, fiecare modul având responsabilități specifice, astfel încât să faciliteze întreținerea și scalarea. Schema generală include următoarele componente principale:

1. Modulul de Încărcare și Preprocesare a Imaginilor si de testare a aplicatiei - Responsabil pentru încărcarea imaginilor din surse externe și aplicarea functiei principale.

```
void testDetectCells() {
	char fname[] = "/path_to_folder/blood1.bmp";
	Mat src = imread(fname, IMREAD_GRAYSCALE);
	imshow("Original", src);
	detectCells(src);
	waitKey(0);
	}
```

2. Modulul principal – Functia care primeste imaginea si aplica pe rand toti pasii.

```
void detectCells(Mat src) {
	Mat temp = pragBinarizare(src);
	imshow("Binarizare", temp);
	temp = removeNoiseFromImage(temp, 9);
	imshow("Reducere zgomot", temp);
	fillHoles(temp);
	imshow("Umplere obiecte", temp);
	std::vector<Cell> cells;
	cells = cellListDetector(temp);
	calculateProperties(cells);
	displayProperties(cells);
	}
```

3. Modulul de binarizare cu prag automat – calculeaza un prag automat pe baza algoritmului de la laborator, iar apoi aplica binarizarea

```
 Mat pragBinarizare(Mat src) {
	int M = src.rows * src.cols;
	int h[256] = { 0 };
	int hc[256] = { 0 };
	float p[256] = { 0 };
	for (int i = 0; i < src.rows; i++) {
		for (int j = 0; j < src.cols; j++) {
			uchar g = src.at<uchar>(i, j);
			h[g] ++;
		}
	}
```

```
	for (int g = 0; g < 256; g++) {
		p[g] = (float)h[g] / M;
	}
	hc[0] = h[0];
	for (int g = 1; g < 256; g++) {
		hc[g] = hc[g - 1] + h[g];
	}
	int imin, imax = 0, ginmin, ginmax;
	int g;
	for (g = 0; g < 256; g++)
		if (h[g] > 0)
			break;
	imin = g;
	float hmax = 0;
	for (int y = 1; y < 256; y++)
		if (h[y] > hmax) {
			hmax = h[y];
			imax = y;
		}

	float e = 0.5;
	float Told = 0;
	float T = (imin + imax) / 2;
	do {
		Told = T;
		float m1 = 0, m2 = 0;
		float sum_m1 = 0, sum_m2 = 0;
		for (g = 0; g < Told; g++) {
			m1 += g * p[g];
			sum_m1 += p[g];
		}
		for (g = Told; g < 256; g++) {
			m2 += g * p[g];
			sum_m2 += p[g];
		}
		if (sum_m1 > 0) m1 /= sum_m1;
		if (sum_m2 > 0) m2 /= sum_m2;
		T = (m1 + m2) / 2;
	} while (abs(T - Told) > e);

	Mat dst(src.rows, src.cols, CV_8UC1);
	for (int i = 0; i < dst.rows; i++) {
		for (int j = 0; j < dst.cols; j++) {
			uchar pixel = src.at<uchar>(i, j);
			if (pixel < T)
				dst.at<uchar>(i, j) = 0;
			else
				dst.at<uchar>(i, j) = 255;
		}
	}
	return dst;
}
```

4. Modulul de reducere zgomot – aplica un filtru median si elimina zgomotul si obiectele mici

```
Mat removeNoiseFromImage(Mat& src, int kernel_size) {
```

```
	Mat filtered, opened, closed;
	filtered = medianBlur(src, kernel_size);
	opened = deschidere(filtered);
	closed = inchidere(opened);
	return closed;
	}
```

5. Modulul de umplere a obiectelor – umple goluri lasate de binarizare si filtrul median

```
void fillHoles(Mat& src) {

	int rows = src.rows;
	int cols = src.cols;
	Mat visited = Mat::zeros(src.size(), CV_8U);

	int dx[] = { -1, 1, 0, 0, -1, -1, 1, 1 };
	int dy[] = { 0, 0, -1, 1, -1, 1, -1, 1 };

	std::queue<Point> q;
	for (int i = 0; i < rows; i++) {
		for (int j = 0; j < cols; j++) {
			if ((i == 0 || i == rows - 1 || j == 0 || j == cols - 1) &&
src.at<uchar>(i, j) == BG && visited.at<uchar>(i, j) == 0) {
				q.push(Point(j, i));
				visited.at<uchar>(i, j) = 1;
			}
		}
	}

	while (!q.empty()) {
		Point p = q.front();
		q.pop();

		for (int k = 0; k < 8; k++) {
			int nx = p.x + dx[k];
			int ny = p.y + dy[k];

			if (nx >= 0 && nx < cols && ny >= 0 && ny < rows &&
src.at<uchar>(ny, nx) == BG && visited.at<uchar>(ny, nx) == 0) {
				q.push(Point(nx, ny));
				visited.at<uchar>(ny, nx) = 1;
			}
		}
	}

	for (int i = 0; i < rows; i++) {
		for (int j = 0; j < cols; j++) {
			if (src.at<uchar>(i, j) == BG && visited.at<uchar>(i, j) == 0) {
				src.at<uchar>(i, j) = FG;
			}
		}
	}
	}
```

6. Modulul de detectare a celulelor – va aduna intr-o structura o lista de celule, alcatuie dintr-o lista de pixeli

```
struct Cell {
	std::vector<Point2d> pointList;
	int area;
	int diameter;
	Point2d Center;
	};
```

7. Modulul de calculare a metricilor – va calcula pentru fiecare celula aria, diametrul si centrul de masa.

Pentru arie:

```
int a = 0;
for (int i = 0; i < (*src).rows; i++) {
	for (int j = 0; j < (*src).cols; j++) {
		Vec3b pixel = (*src).at<Vec3b>(y, x);
			if (equalPixels(pixel, (*src).at<Vec3b>(i, j))) {
				a += 1;
			}
		}
	}
```

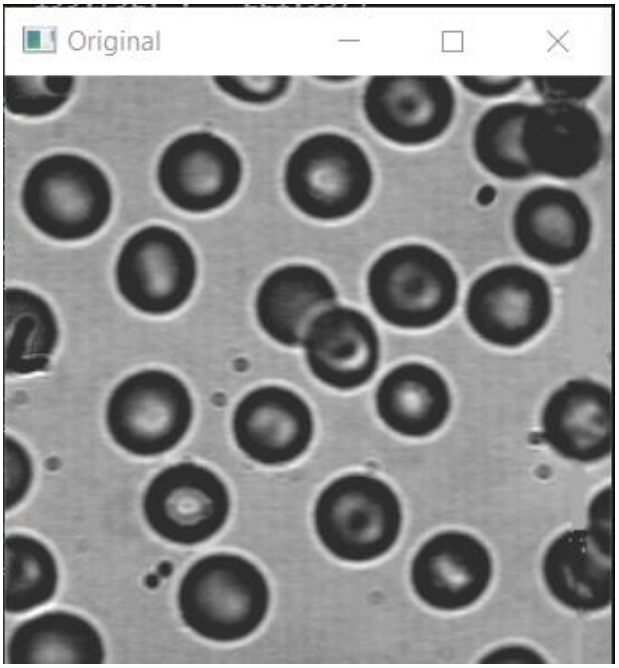
Pentru centru:

```
int row = 0, col = 0;
for (int i = 0; i < (*src).rows; i++) {
	for (int j = 0; j < (*src).cols; j++) {
		Vec3b pixel = (*src).at<Vec3b>(y, x);
		if (equalPixels(pixel, (*src).at<Vec3b>(i, j))) {
			row += i;
			col += j;
}
	}
}
int row_c = row / a;
int col_c = col / a;
```

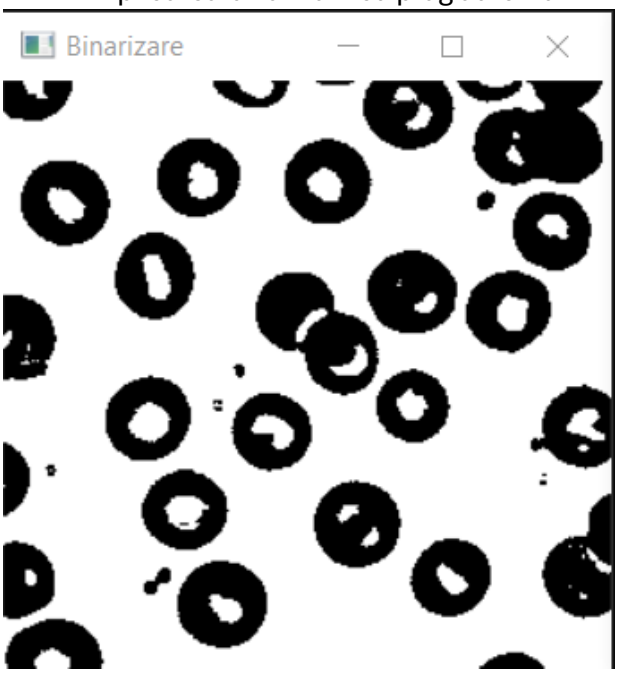
8. Modulul de afisare a rezultatelor – va itera in lista de obiecte Cell, afisand pentru fiecare aria, diametrul si centrul de masa, iar apoi va deschide o noua imagine care va consta in imaginea initiala si centrele de masa marcate.

Testare si validare

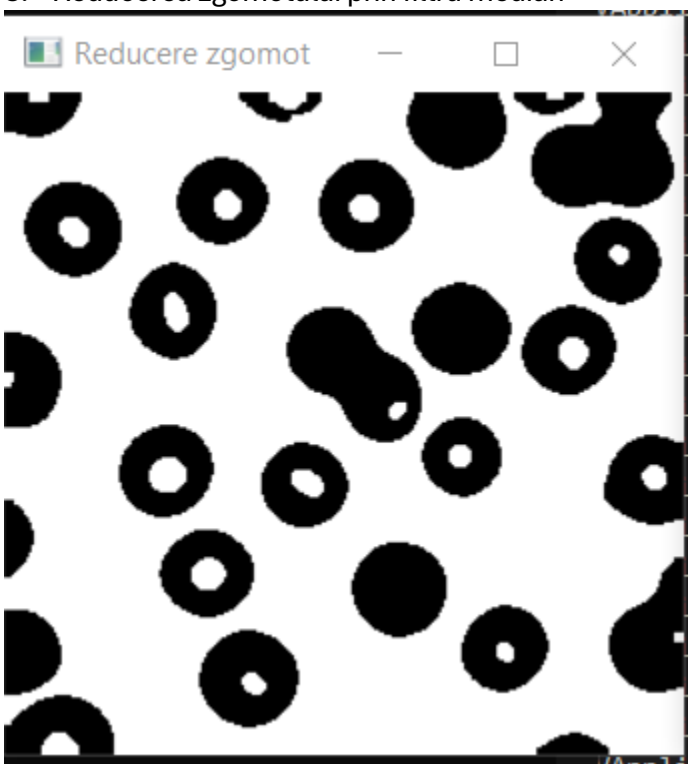
1. Imaginea originala.



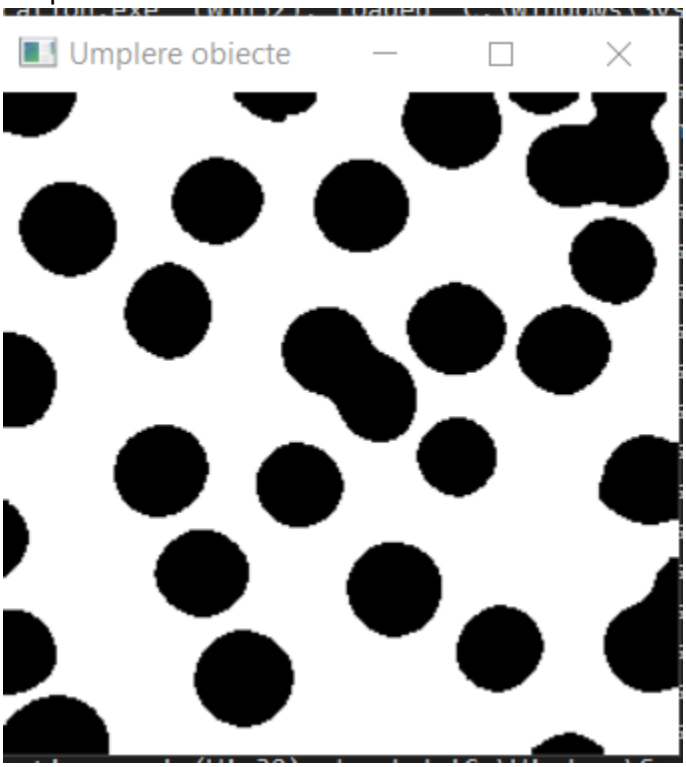
2. Aplicarea binarizarii cu prag automat



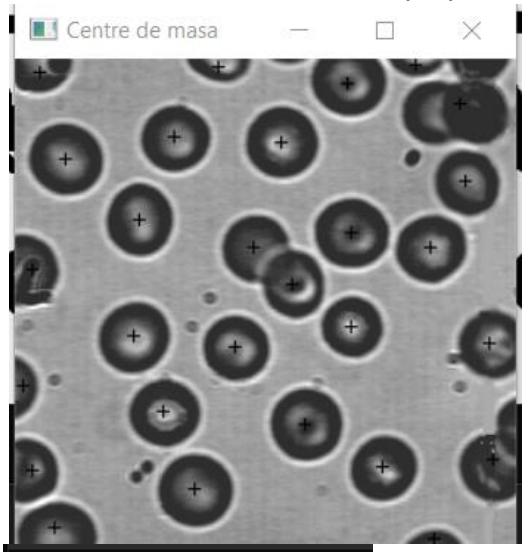
3. Reducerea zgomotului prin filtru median



4. Umplerea celulelor folosind BFS



5. Afisarea centrelor de masa si a proprietatiilor fiecarei celule




```
Cell Properties:
 Area: 465
 Diameter: 24.3325
 Center: (x = 13.4581, y = 7.43441)
Cell Properties:
 Area: 282
 Diameter: 18.9489
 Center: (x = 110.316, y = 4.17376)
Cell Properties:
 Area: 1017
 Diameter: 35.9849
 Center: (x = 180.851, y = 13.3441)
Cell Properties:
 Area: 189
 Diameter: 15.5128
 Center: (x = 217.402, y = 3.16931)
Cell Properties:
 Area: 1980
 Diameter: 50.2102
 Center: (x = 241.04, y = 24.7015)
Cell Properties:
 Area: 970
 Diameter: 35.1435
 Center: (x = 86.7784, y = 42.9608)
Cell Properties:
 Area: 1104
 Diameter: 37.4925
 Center: (x = 144.444, y = 45.0254)
Cell Properties:
 Area: 1139
 Diameter: 38.0822
 Center: (x = 27.2906, y = 54.2028)
Cell Properties:
 Area: 919
 Diameter: 34.2072
 Center: (x = 244.939, y = 66.8672)
Cell Properties:
 Area: 1022
 Diameter: 36.0732
 Center: (x = 67.2945, y = 87.1047)
Cell Properties:
 Area: 1122
 Diameter: 37.7969
 Center: (x = 182.316, y = 94.3342)
```