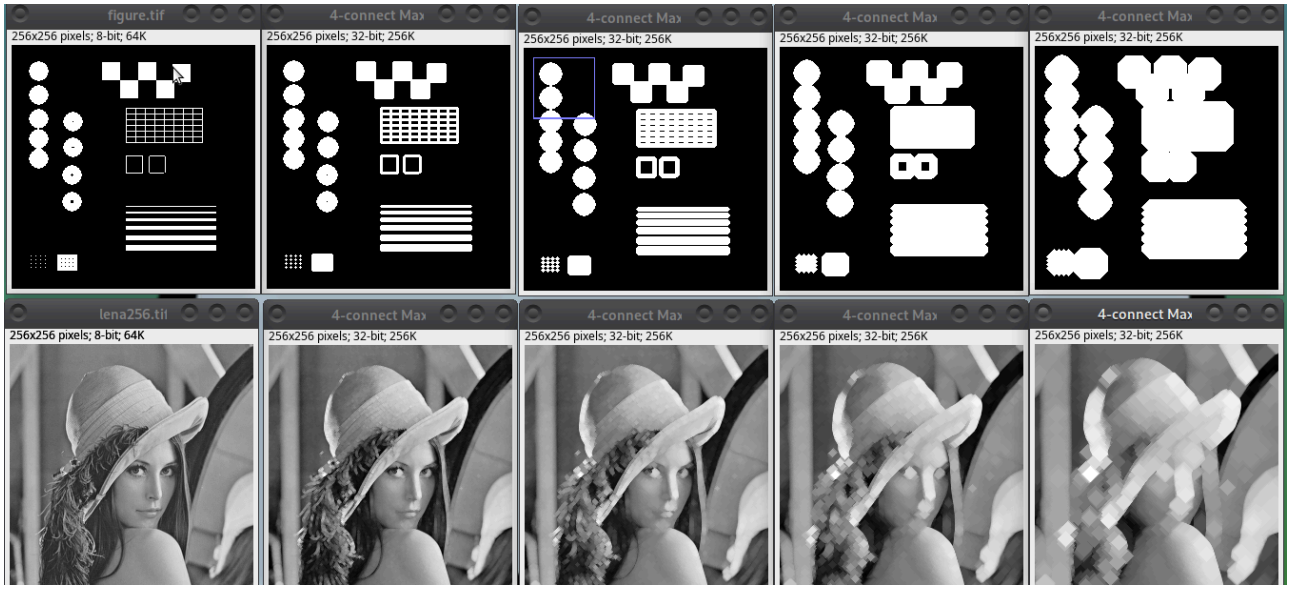
Nome: Lucas Miranda Mendonça Rezende
Nro USP: 12542838

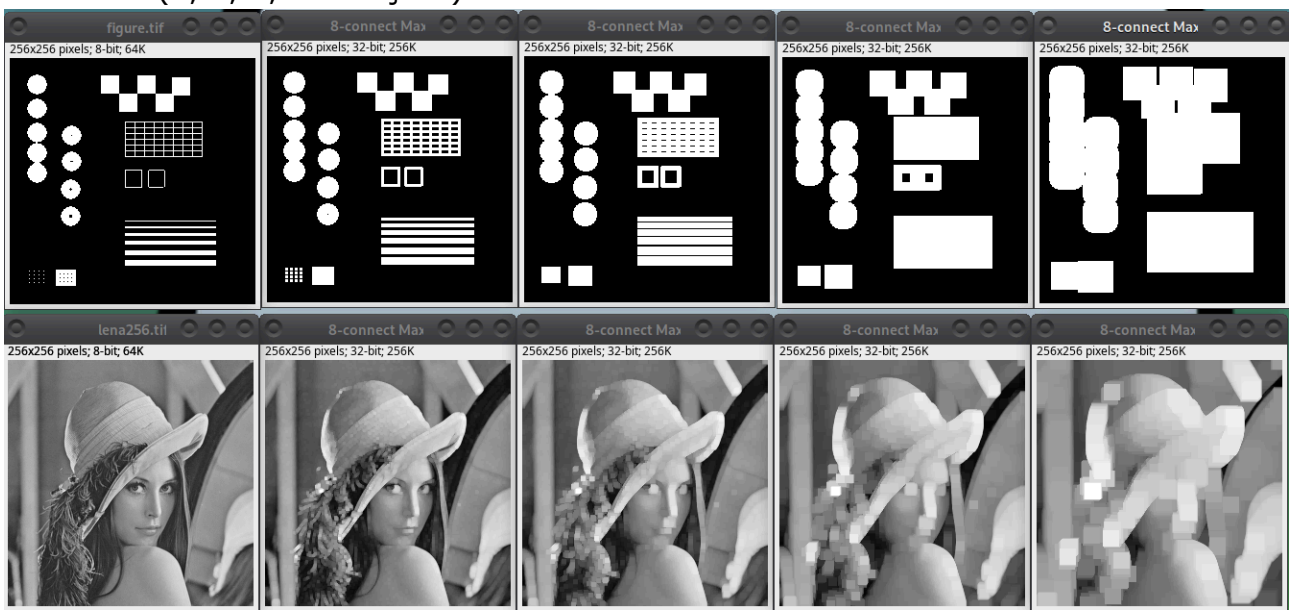
relatório.doc **Operadores Morfológicos**

Questão 1:

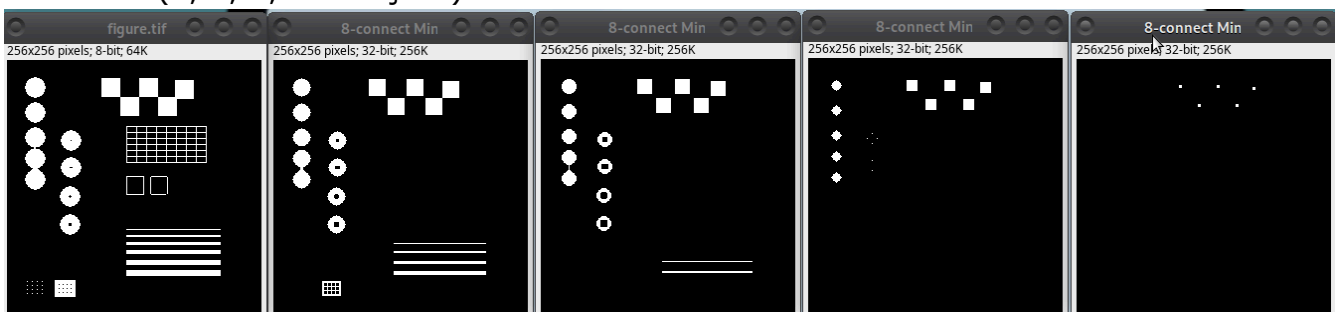
4-connect (1, 2, 4, 8 iterações):



8-connect (1, 2, 4, 8 iterações):



Questão 2.1:
8-connect (1, 2, 4, 8 iterações):

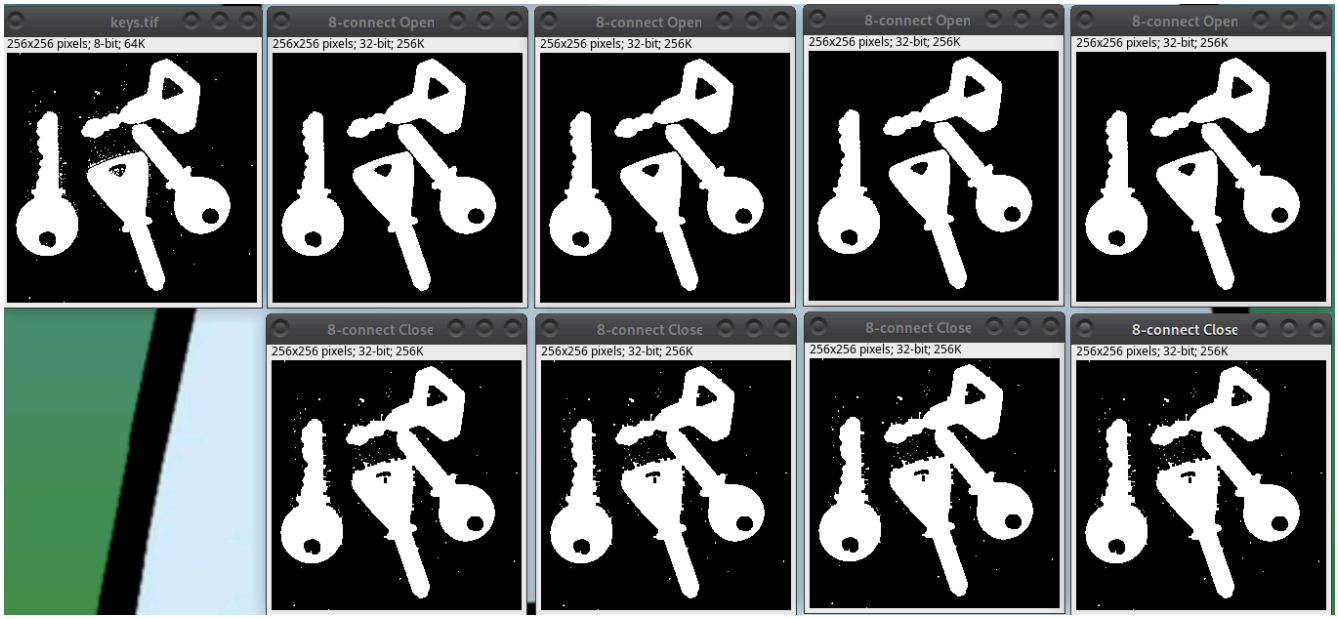


Algoritmo:

```
static public ImageAccess doErosion(ImageAccess img) {
        int nx = img.getWidth();
        int ny = img.getHeight();
        ImageAccess out = new ImageAccess(nx, ny);
        double arr[] = new double[9];
        double min;

        for (int x = 0; x < nx; x++) {
            for (int y = 0; y < ny; y++) {
                // grab the 3×3 neighborhood around (x,y)
                img.getPattern(x, y, arr, ImageAccess.PATTERN_SQUARE_3x3);

                // initialize min to the first element
                min = arr[0];
                // find the minimum value
                for (int k = 1; k < arr.length; k++) {
                    if (arr[k] < min) {
                        min = arr[k];
                    }
                }

                // write the minimum into the output pixel
                out.putPixel(x, y, min);
            }
        }

        return out;
    }
```

Questão 2.2:

8-connect (1, 2, 4, 8 iterações) (Open, Close):

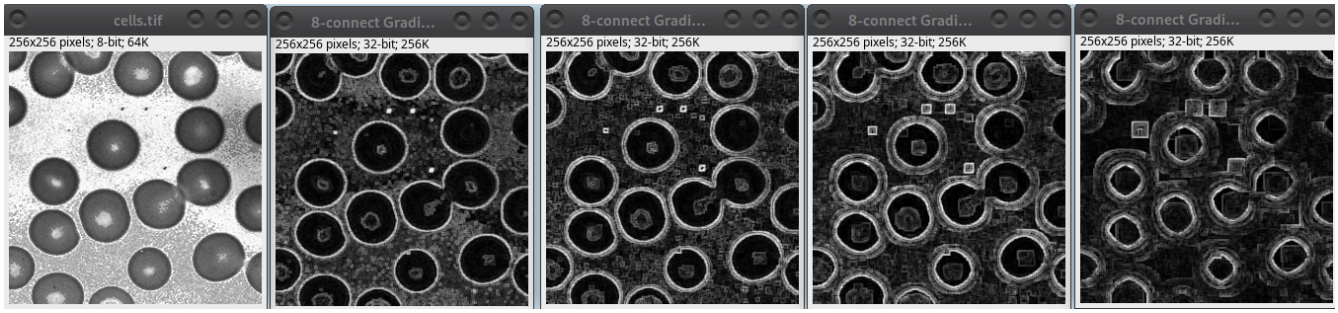


Algoritmo:

```
static public ImageAccess doOpen(ImageAccess img) {
        // First erode, then dilate the result
        ImageAccess eroded = doErosion(img);
        ImageAccess opened = doDilation(eroded);
        return opened;
    }

    static public ImageAccess doClose(ImageAccess img) {
        // First dilate, then erode the result
        ImageAccess dilated = doDilation(img);
        ImageAccess closed = doErosion(dilated);
        return closed;
    }
```

Questão 2.3:

8-connect (1, 2, 4, 8 iterações):



Algoritmo:

```
static public ImageAccess doGradient(ImageAccess img) {
        int nx = img.getWidth();
        int ny = img.getHeight();
        ImageAccess out = new ImageAccess(nx, ny);

        // Compute morphological gradient = dilation(img) - erosion(img)
        ImageAccess dil = doDilation(img);
        ImageAccess ero = doErosion(img);
        for (int x = 0; x < nx; x++) {
            for (int y = 0; y < ny; y++) {
                double val = dil.getPixel(x, y) - ero.getPixel(x, y);
                out.putPixel(x, y, val);
            }
        }
        out.normalizeContrast();
        return out;
    }
```

Questão 2.4:

8-connect (1, 2, 4, 8 iterações) (Bright, Dark):

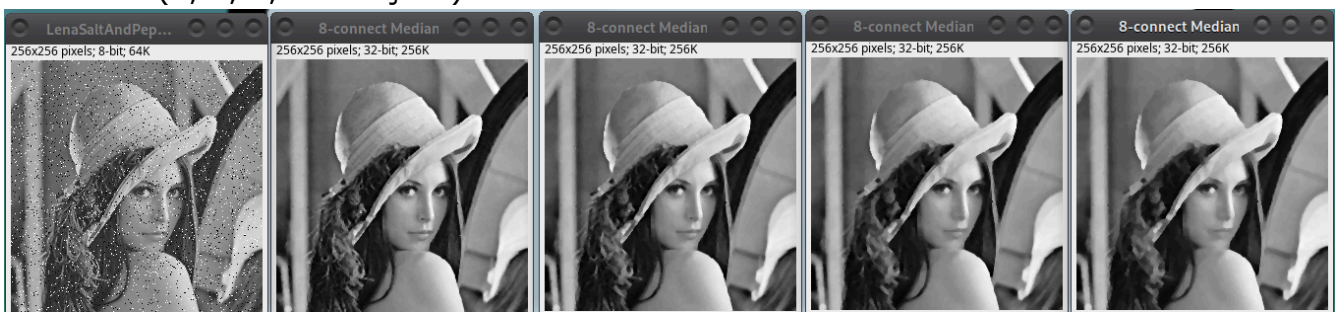
Algoritmo:

```
static public ImageAccess doTopHatBright(ImageAccess img) {
        int nx = img.getWidth();
        int ny = img.getHeight();
        ImageAccess out = new ImageAccess(nx, ny);

        // Top-hat bright = original(img) - opening(img)
        ImageAccess opened = doOpen(img);
        for (int x = 0; x < nx; x++) {
            for (int y = 0; y < ny; y++) {
                double val = img.getPixel(x, y) - opened.getPixel(x, y);
                out.putPixel(x, y, val);
            }
        }
        out.normalizeContrast();
        return out;
    }

    static public ImageAccess doTopHatDark(ImageAccess img) {
        int nx = img.getWidth();
        int ny = img.getHeight();
        ImageAccess out = new ImageAccess(nx, ny);

        // Top-hat dark = closing(img) - original(img)
        ImageAccess closed = doClose(img);
```

```
        for (int x = 0; x < nx; x++) {
            for (int y = 0; y < ny; y++) {
                double val = closed.getPixel(x, y) - img.getPixel(x, y);
                out.putPixel(x, y, val);
            }
        }
        out.normalizeContrast();
        return out;
    }
```

Questão 2.5:

8-connect (1, 2, 4, 8 iterações):



Algoritmo:

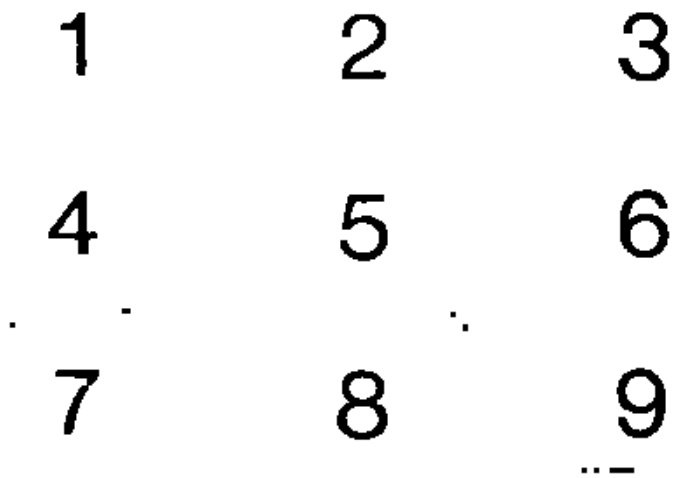
```
static public ImageAccess doMedian(ImageAccess img) {
        int nx = img.getWidth();
        int ny = img.getHeight();
        ImageAccess out = new ImageAccess(nx, ny);
        double arr[] = new double[9];

        for (int x = 0; x < nx; x++) {
            for (int y = 0; y < ny; y++) {
                // grab 3×3 neighborhood
                img.getPattern(x, y, arr, ImageAccess.PATTERN_SQUARE_3x3);
                // sort and pick middle
                sortArray(arr);
                double median = arr[arr.length / 2]; // index 4
                out.putPixel(x, y, median);
            }
        }
        return out;
    }
```

Questão 3.1:

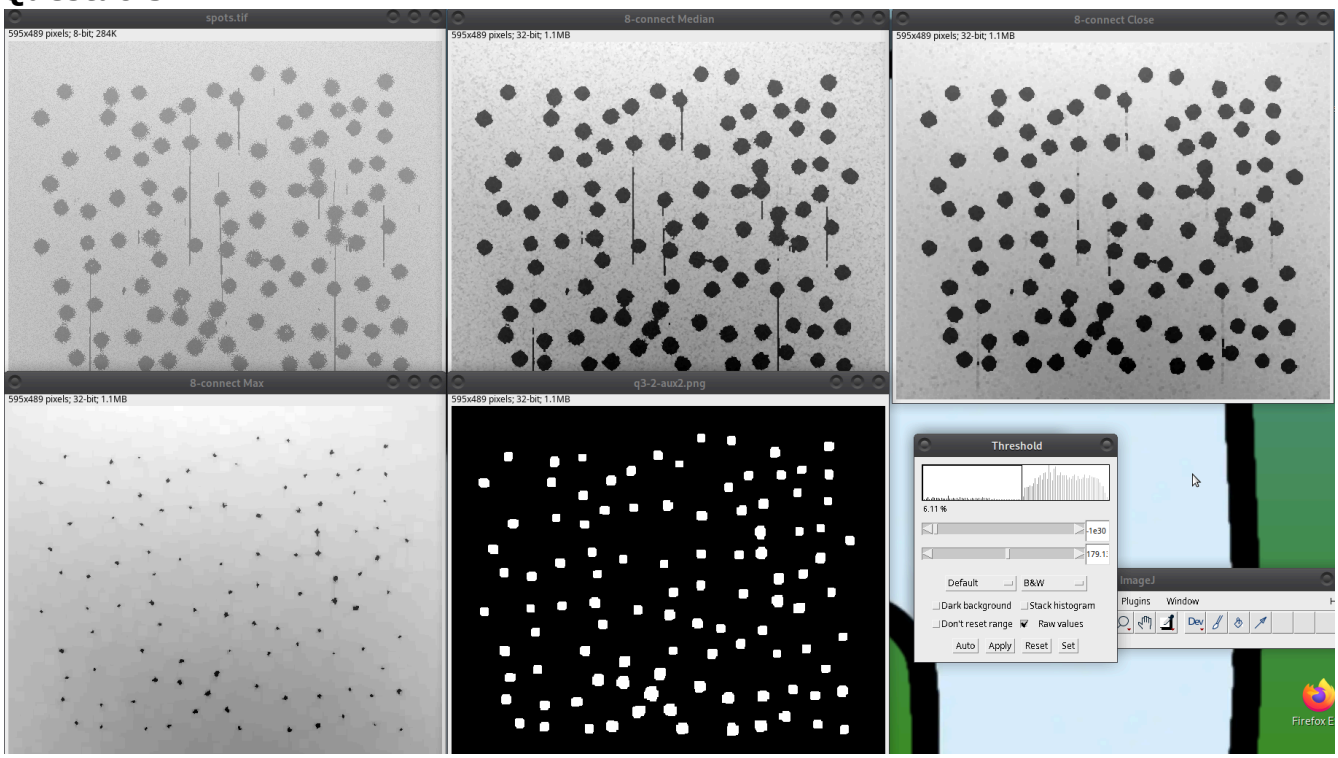
Algoritmo:

1. Open (10 iterações)
2. Threshold (139)

Resultado:



Questão 3.2:

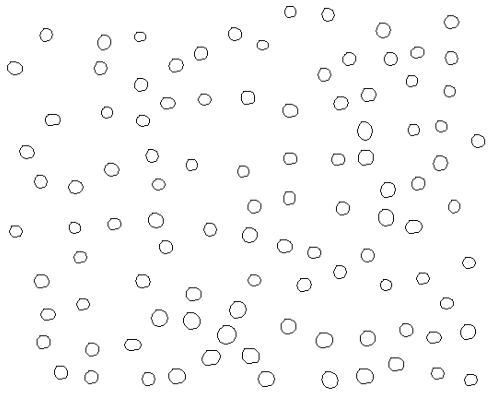


Algoritmo:

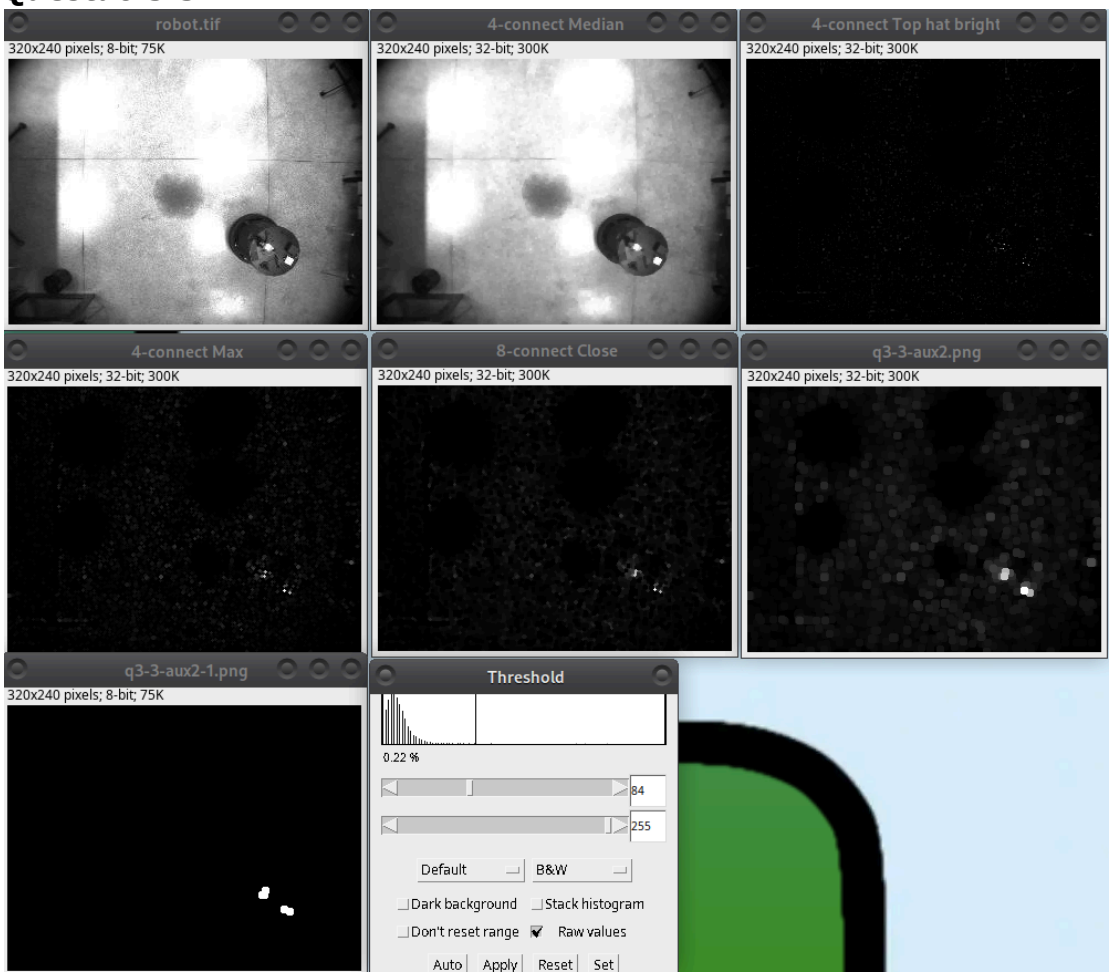
1. Median (1 iteração)
2. Close (1 iteração)
3. Max (6 iterações)
4. Min (4 iterações)
5. Threshold (179)
6. Analyze particles (min pixel size = 1)

Resultado:

- Número de spots: **102**



Questão 3.3:



Algoritmo:

1. Median (10 iterações)
2. Tophat Bright (1 iteração)
3. Max (1 iteração)
4. Close (10 iterações)
5. Max (2 iterações)
6. Threshold (84)

Resultado: