**Nome: Lucas Miranda Mendonça Rezende**

Nro. USP: 12542838

**relatorio.doc Detecção de Bordas**

**Questão 1**

Apenas introdutória.

**Questão 2.2**

static public ImageAccess blurring(ImageAccess input, double sigma) {

int nx = input.getWidth();

int ny = input.getHeight();

ImageAccess out = new ImageAccess(nx, ny);

// Número de filtros em cascata

int N = 3;

// Calcula a variância (sigma^2)

double variance = sigma \* sigma;

// Calcular o valor do pólo 'a' com base na relação:

// a = ((sigma^2 + N) - sqrt(N\*(2\*sigma^2 + N)))/sigma^2

double a = ((variance + N) - Math.sqrt(N \* (2 \* variance + N))) / variance;

// Definir os pólos do filtro IIR

double[] poles = new double[N];

for (int i = 0; i < N; i++) {

poles[i] = a;

}

// Passo 1: Convolução ao longo das linhas

for (int j = 0; j < ny; j++) {

double[] row = new double[nx];

for (int i = 0; i < nx; i++) {

row[i] = input.getPixel(i, j);

}

// Aplicar convolução 1D na linha

double[] blurredRow = Convolver.convolveIIR(row, poles);

for (int i = 0; i < nx; i++) {

out.putPixel(i, j, blurredRow[i]);

}

}

// Passo 2: Convolução ao longo das colunas

for (int i = 0; i < nx; i++) {

double[] col = new double[ny];

for (int j = 0; j < ny; j++) {

col[j] = out.getPixel(i, j);

}

// Aplicar convolução 1D na coluna

double[] blurredCol = Convolver.convolveIIR(col, poles);

for (int j = 0; j < ny; j++) {

out.putPixel(i, j, blurredCol[j]);

}

}

// Normalização: compensar o ganho DC da cascata dos filtros.

// Para cada direção (linhas e colunas) o ganho é 1/(1-a)^N, portanto o ganho total 2D é 1/(1-a)^(2N)

// Multiplicando por scale = (1-a)^(2N) garantimos que o ganho DC seja 1.

double scale = Math.pow((1 - a), 2 \* N);

for (int i = 0; i < nx; i++) {

for (int j = 0; j < ny; j++) {

out.putPixel(i, j, out.getPixel(i, j) \* scale);

}

}

return out;

}

**Questão 3.1**

**static public ImageAccess[] gradient(ImageAccess input) {**

**int nx = input.getWidth();**

**int ny = input.getHeight();**

**// grad[0] = magnitude, grad[1] = gx, grad[2] = gy**

**ImageAccess[] grad = new ImageAccess[3];**

**grad[0] = new ImageAccess(nx, ny);**

**grad[1] = new ImageAccess(nx, ny);**

**grad[2] = new ImageAccess(nx, ny);**

**// Filtros (3x3), cada elemento já multiplicado por 1/12**

**double[][] hx = {**

**{ -1.0/12, 0.0, 1.0/12 },**

**{ -4.0/12, 0.0, 4.0/12 },**

**{ -1.0/12, 0.0, 1.0/12 }**

**};**

**double[][] hy = {**

**{ -1.0/12, -4.0/12, -1.0/12 },**

**{ 0.0, 0.0, 0.0 },**

**{ 1.0/12, 4.0/12, 1.0/12 }**

**};**

**// Passo 1: Convolução 2D para gx e gy (ignorando a borda de 1 pixel)**

**for (int y = 1; y < ny - 1; y++) {**

**for (int x = 1; x < nx - 1; x++) {**

**double sumX = 0.0;**

**double sumY = 0.0;**

**// Aplicar cada núcleo 3x3 ao redor do pixel (x,y)**

**for (int ky = -1; ky <= 1; ky++) {**

**for (int kx = -1; kx <= 1; kx++) {**

**double val = input.getPixel(x + kx, y + ky);**

**sumX += val \* hx[ky + 1][kx + 1];**

**sumY += val \* hy[ky + 1][kx + 1];**

**}**

**}**

**// Armazenar resultados em grad[1] e grad[2]**

**grad[1].putPixel(x, y, sumX); // gx**

**grad[2].putPixel(x, y, sumY); // gy**

**}**

**}**

**// Passo 2: Calcular o módulo do gradiente (grad[0] = sqrt(gx^2 + gy^2))**

**for (int y = 1; y < ny - 1; y++) {**

**for (int x = 1; x < nx - 1; x++) {**

**double gx = grad[1].getPixel(x, y);**

**double gy = grad[2].getPixel(x, y);**

**double magnitude = Math.sqrt(gx \* gx + gy \* gy);**

**grad[0].putPixel(x, y, magnitude);**

**}**

**}**

**return grad;**

**}**

**Questão 4.2**

**static public ImageAccess suppressNonMaximum(ImageAccess grad[]) {**

**if (grad.length != 3) {**

**return null;**

**}**

**int nx = grad[0].getWidth();**

**int ny = grad[0].getHeight();**

**ImageAccess suppressed = new ImageAccess(nx, ny);**

**// Percorrer cada pixel da imagem**

**for (int y = 1; y < ny - 1; y++) {**

**for (int x = 1; x < nx - 1; x++) {**

**// Obter o valor do gradiente em x e y**

**double gx = grad[1].getPixel(x, y);**

**double gy = grad[2].getPixel(x, y);**

**// Calcular a magnitude do gradiente G(A)**

**double magnitude = Math.sqrt(gx \* gx + gy \* gy);**

**// Se a magnitude for 0, supressão imediata**

**if (magnitude == 0) {**

**suppressed.putPixel(x, y, 0);**

**continue;**

**}**

**// Calcular o vetor unitário na direção do gradiente**

**double norm = Math.sqrt(gx \* gx + gy \* gy);**

**double ux = gx / norm;**

**double uy = gy / norm;**

**// Calcular as posições A1 e A2**

**double xa1 = x + ux;**

**double ya1 = y + uy;**

**double xa2 = x - ux;**

**double ya2 = y - uy;**

**// Obter os valores de G(A1) e G(A2) por interpolação**

**double ga1 = grad[0].getInterpolatedPixel(xa1, ya1);**

**double ga2 = grad[0].getInterpolatedPixel(xa2, ya2);**

**// Se G(A) for maior que G(A1) e G(A2), manter o valor, caso contrário, suprimir**

**if (magnitude >= ga1 && magnitude >= ga2) {**

**suppressed.putPixel(x, y, magnitude);**

**} else {**

**suppressed.putPixel(x, y, 0);**

**}**

**}**

**}**

**return suppressed;**

**}**

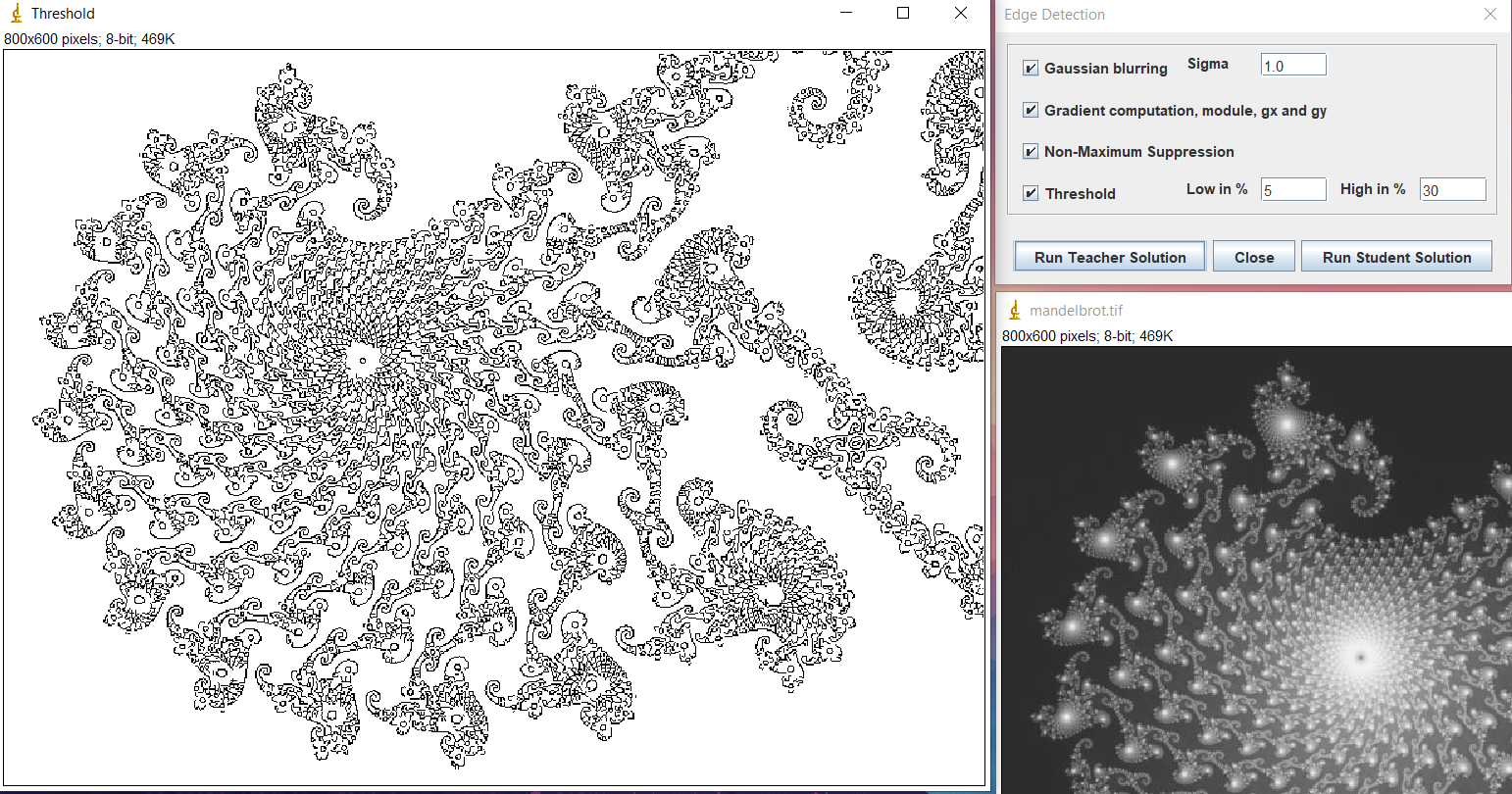
**Questão 5**

Insira aqui uma imagem (Detalhe de Madelbrot.tif)

Sigma: 1

Threshold low: 5

Threshold high: 30

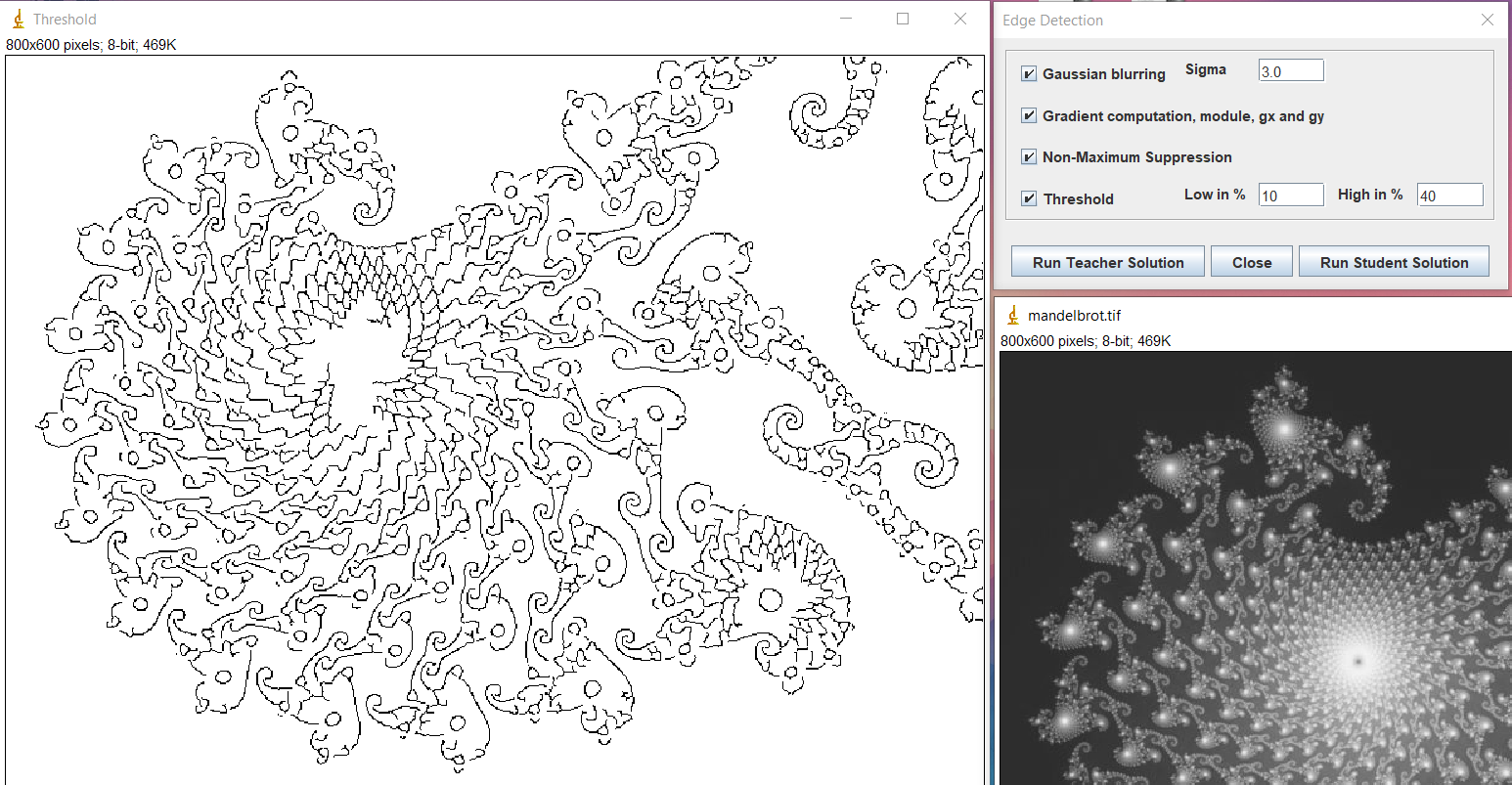


Insira aqui uma imagem (Visão global de mandelbrot.tif)

Sigma: 3

Threshold low: 10

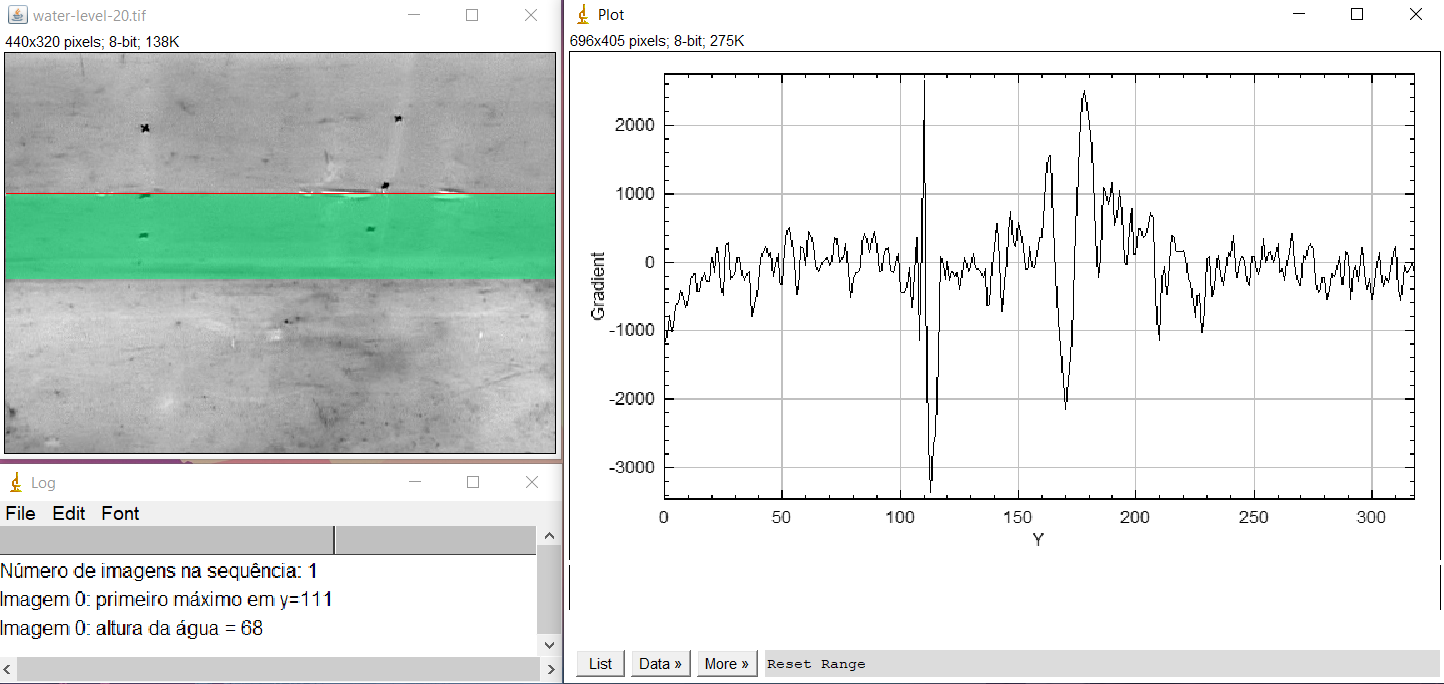
Threshold high: 40



**Questão 6.1**

y1 = 111

y2 = 179



static public double[] computeXProjectionGradient(ImageAccess image) {

int ny = image.getHeight();

int nx = image.getWidth();

double[] proj = new double[ny];

for (int y = 0; y < ny; y++) {

double sum = 0;

for (int x = 0; x < nx; x++) {

sum += image.getPixel(x, y);

}

proj[y] = sum;

}

double[] grad = new double[ny - 1];

for (int y = 0; y < ny - 1; y++) {

grad[y] = proj[y + 1] - proj[y];

}

DisplayTools.plot(grad, "Y", "Gradient");

return grad;

}

**Questão 6.2 e Questão 6.3**

Imprima as posições de ymax ou insira o gráfico ymax(t)

Imprima as posições de altura ou insira o gráfico altura (t)

Número de imagens na sequência: 40

Imagem 0: primeiro máximo em y=140

Imagem 0: altura da água = 49

Imagem 1: primeiro máximo em y=137

Imagem 1: altura da água = 51

Imagem 2: primeiro máximo em y=186

Imagem 2: altura da água = -15

Imagem 3: primeiro máximo em y=187

Imagem 3: altura da água = -16

Imagem 4: primeiro máximo em y=185

Imagem 4: altura da água = -15

Imagem 5: primeiro máximo em y=184

Imagem 5: altura da água = -15

Imagem 6: primeiro máximo em y=183

Imagem 6: altura da água = -15

Imagem 7: primeiro máximo em y=182

Imagem 7: altura da água = -59

Imagem 8: primeiro máximo em y=181

Imagem 8: altura da água = -61

Imagem 9: primeiro máximo em y=181

Imagem 9: altura da água = -63

Imagem 10: primeiro máximo em y=117

Imagem 10: altura da água = 63

Imagem 11: primeiro máximo em y=114

Imagem 11: altura da água = 66

Imagem 12: primeiro máximo em y=180

Imagem 12: altura da água = -68

Imagem 13: primeiro máximo em y=111

Imagem 13: altura da água = 68

Imagem 14: primeiro máximo em y=109

Imagem 14: altura da água = 69

Imagem 15: primeiro máximo em y=108

Imagem 15: altura da água = 70

Imagem 16: primeiro máximo em y=177

Imagem 16: altura da água = -70

Imagem 17: primeiro máximo em y=177

Imagem 17: altura da água = -15

Imagem 18: primeiro máximo em y=103

Imagem 18: altura da água = 74

Imagem 19: primeiro máximo em y=177

Imagem 19: altura da água = -75

Imagem 20: primeiro máximo em y=100

Imagem 20: altura da água = 76

Imagem 21: primeiro máximo em y=99

Imagem 21: altura da água = 78

Imagem 22: primeiro máximo em y=175

Imagem 22: altura da água = -78

Imagem 23: primeiro máximo em y=173

Imagem 23: altura da água = -77

Imagem 24: primeiro máximo em y=94

Imagem 24: altura da água = 79

Imagem 25: primeiro máximo em y=93

Imagem 25: altura da água = 80

Imagem 26: primeiro máximo em y=91

Imagem 26: altura da água = 81

Imagem 27: primeiro máximo em y=89

Imagem 27: altura da água = 83

Imagem 28: primeiro máximo em y=88

Imagem 28: altura da água = 85

Imagem 29: primeiro máximo em y=87

Imagem 29: altura da água = 86

Imagem 30: primeiro máximo em y=84

Imagem 30: altura da água = 89

Imagem 31: primeiro máximo em y=171

Imagem 31: altura da água = -91

Imagem 32: primeiro máximo em y=78

Imagem 32: altura da água = 91

Imagem 33: primeiro máximo em y=168

Imagem 33: altura da água = -94

Imagem 34: primeiro máximo em y=168

Imagem 34: altura da água = -16

Imagem 35: primeiro máximo em y=168

Imagem 35: altura da água = -95

Imagem 36: primeiro máximo em y=76

Imagem 36: altura da água = 93

Imagem 37: primeiro máximo em y=169

Imagem 37: altura da água = -90

Imagem 38: primeiro máximo em y=82

Imagem 38: altura da água = 89

Imagem 39: primeiro máximo em y=85

Imagem 39: altura da água = 86

static public void measureLevel(ImageAccess[] sequence) {

int nt = sequence.length;

IJ.log("Número de imagens na sequência: " + nt);

// Iterate over each image in the sequence.

for (int t = 0; t < nt; t++) {

ImageAccess image = sequence[t];

// (1) Compute the gradient of the horizontal projection.

double[] gradient = computeXProjectionGradient(image);

// (2) Detect the first maximum in the gradient.

// Add +1 because the gradient array is offset relative to the original image rows.

int yMax = findMax(gradient) + 1;

IJ.log("Imagem " + t + ": primeiro máximo em y=" + yMax);

DisplayTools.drawLine(t, yMax);

// (3) Detect two maxima that are at least 10 pixels apart.

int[] twoMax = findTwoMaxima(gradient);

int y1 = twoMax[0] + 1; // adjust index offset

int y2 = twoMax[1] + 1; // adjust index offset

if (y1 >= 0 && y2 >= 0) {

// Draw a rectangle between y1 and y2.

DisplayTools.drawLevels(t, y1, y2);

// Calculate and log the water level height.

int h = y2 - y1;

IJ.log("Imagem " + t + ": altura da água = " + h);

} else {

IJ.log("Imagem " + t + ": não foi possível encontrar dois máximos a >=10 px de distância.");

}

// (4) Plot the gradient for further diagnosis.

DisplayTools.plot(gradient, "Y", "Gradient");

}

}

static private int findMax(double[] data) {

int indexMax = 0;

double maxVal = data[0];

for (int i = 1; i < data.length; i++) {

if (data[i] > maxVal) {

maxVal = data[i];

indexMax = i;

}

}

return indexMax;

}

static private int[] findTwoMaxima(double[] data) {

int[] maxima = new int[2];

maxima[0] = findMax(data);

// Search for a second maximum with a separation of at least 10 pixels.

int secondMax = -1;

for (int i = 0; i < data.length; i++) {

if (i != maxima[0] && Math.abs(i - maxima[0]) >= 10) {

if (secondMax == -1 || data[i] > data[secondMax]) {

secondMax = i;

}

}

}

maxima[1] = secondMax;

return maxima;

}