

```

import ij.*;
import ij.process.*;
import ij.gui.*;
import Libreria_01_Gabriel.*;
import ij.plugin.filter.*;
import ij.plugin.Plugin;
import ij.measure.*;
//
// VAMOS A CAMBIAR LA MÁSCARA DE BÚSQUEDA A UN CÍRCULO
// CONSTRUIDO SOBRE R_AUTOCORREGIDO, QUE AÑADÍA MULTIÁNGULO Y CORRECCIÓN DE FORMA AUTOMÁTICA,
// ES DECIR, QUE MARCA EN BLANCO AQUEL ROMBO EN EL QUE EL ALGORITMO HA ACERTADO EN SU PREVISIÓN, SIMPLEMENTE COMPARÁNDOLO
// EN LOS DATOS ALMACENADOS EN LIBRO.CSV
// ADEMÁS, EN ESTE PLUGIN VAMOS A PROCEDIMENTAR EN PAQUETES LAS LLAMADAS A FUNCIONES EXTERNAS.
//
public class Evaluacion_R_a_fichero implements PlugInFilter {

    ImagePlus imp;

    int i, j, contador, x, y, a, b, c, d, pointer, sum, apoyo_int, alto, ancho;
    double apoyo_double;
    short array_imagen[];
    int array_esquinas_rombos[];
    int array_diagonal[];
    boolean numera_esquinas;
    double diagonal_dcha_arriba, diagonal_izda_arriba;
    boolean show_table;
    double array_celdas_datos[][];

    public int setup(String arg, ImagePlus imp) {
        this.imp = imp;
        return DOES_ALL;
    }

    public void run(ImageProcessor ip) {
        //
        // DIALOG BOX VARIABLES AND INITIAL TEST
        //
        String title_1, title_2, message_1, message_2;
        int filter_length, image_height, image_width, image_dimension, bits_per_pixel_1;

```

```

//
// NEXT WE CAN FIND THE EXPONENTS FOR THE DIFFERENT SCALES OF MS-SSIM. PROPOSED BY ZHOU WANG IN HIS INITIAL PAPER OF MS-SSIM
//
double luminance_exponent [] = { 1, 1, 1, 1, 1, 1, 1, 1, 0.1333};
double contrast_exponent [] = { 1, 0.0448, 0.2856, 0.3001, 0.2363, 0.1333};
double structure_exponent []= { 1, 0.0448, 0.2856, 0.3001, 0.2363, 0.1333};

int[] wList = WindowManager.getIDList();
if (wList==null) {
    IJ.error("There is no image open");
    return;
}
a = WindowManager.getImageCount();
if (a!=1) {
    IJ.error("There must be only one image open to calculate any index");
    return;
}
bits_per_pixel_1=imp.getBitDepth();
if (bits_per_pixel_1 == 24){
    IJ.error("RGB images are not supported!");
    return;
}
}
//
// THIS DIALOG BOX SHOWS DIFFERENT OPTIONS
//
double margen_disco_central =3, border=5;
numera_esquinas=true;

GenericDialog gd = new GenericDialog ("R* Index calculation");
gd.addNumericField ("Margin central disc (in pixels):", margen_disco_central, 0); //INDICA EL COLLAR ALREDEDOR DEL DISCO
gd.addNumericField ("Border around disc:", border, 0); // INDICA EL ÁREA DE BÚSQUEDA
gd.addCheckbox ("Show grid:", true);
gd.addNumericField ("Diagonal_dcha_arriba:", diagonal_dcha_arriba, 0);
gd.addNumericField ("Diagonal_izda_arriba:", diagonal_izda_arriba, 0);
gd.addCheckbox ("Show table:", true);
gd.showDialog();
if (gd.wasCanceled()) return;
margen_disco_central = gd.getNextNumber();

```

```

border = gd.getNextNumber();
numera_esquinas = gd.getNextBoolean(); // THIS VARIABLE SHOWS THE GRID
diagonal_dcha_arriba = gd.getNextNumber();
diagonal_izda_arriba = gd.getNextNumber();
show_table = gd.getNextBoolean();
//
// NOW, WE CREATE THE FILTER (COULD BE ANOTHER KIND OF FILTER, AS MEDIAN FILTER)
//
int lpf_width = 9, filter_width=11;
float window_weights [] = new float [(filter_width*filter_width)];
float [] lpf = new float [(lpf_width*lpf_width)];

Libreria_01_Gabriel.filtros.crea (window_weights, lpf, filter_width, lpf_width);
//
// CALCULO DE LA REJILLA
//
alto = ip.getHeight();
ancho = ip.getWidth();
String entrada = "Detecting disks";
ImagePlus nuevo = NewImage.createShortImage (entrada, ancho, alto, 1, NewImage.FILL_BLACK);
ImageProcessor nuevo_ip = nuevo.getProcessor();
nuevo_ip.copyBits(ip,0,0,Blitter.COPY);
array_imagen = (short []) nuevo_ip.getPixels();
array_esquinas_rombos = new int[224]; //3240 ANTES, DEBÍA SER UN RESTO DE PROGRAMACIÓN. EL NÚMERO DE ESQUINAS ES DE 223 (DE 0 A 222).
array_diagonal = new int[2];

Libreria_01_Gabriel.grid.detection ( array_esquinas_rombos, array_imagen, array_diagonal, alto, ancho, numera_esquinas, diagonal_dcha_arriba, diagonal_izda_arriba); // DETECCION DE ESQUINAS
//
// LECTURA DEL FICHERO CON LOS DATOS DE LAS ESQUINAS
//
array_celdas_datos = new double [205][17];
Libreria_01_Gabriel.csvfile.reading_disc_data (array_celdas_datos);
//
// VAMOS A CALCULAR LOS CUADRADOS DE IMÁGENES QUE SE COMPARARÁN CON MS-SSIM
//
int distancia_esq_disco = (int) (array_diagonal[0] + array_diagonal[1])/10;
int lado_cuadrado_busqueda =0, lado_cuadrado_central=0, apoyo=0;

```

```

ImageProcessor image_1_p, image_2_p;
int iterations = (int) ((border + 1)*(border + 1)); // ES EL NÚMERO DE BÚSQUEDAS QUE HAY EN CADA ZONA, EL NÚMERO DE CUADRADOS QUE EXPLORAMOS POR ZONA

int centro [] = new int[5]; // PUNTO CENTRAL DE LOS CUADRADOS QUE SE COPIAN. El último es el centro de la celda
int esquina [] = new int [5]; // ESQUINA SUPERIOR DERECHA DE LOS CUADRADOS QUE SE COPIAN, incluido el del centro

double reserva_margen = margen_disco_central;

ResultsTable informe = ResultsTable.getResultsTable(); // TABLA PARA MOSTRAR LOS RESULTADOS DE ANÁLISIS DE R*
informe.reset();

double diametro_anterior =60; //PRIMER DIAMETRO
int results_apoyo=0;

for (i=0; i<205;i++) { // DEJAR EN 205. SE PONE ESE VALOR SI NO SE VEN LAS DOS ESQUINAS DE ABAJO. ES EL NÚMERO DE CELDAS QUE TIENE EL CDMAM

    if (array_celdas_datos[i][0] != array_celdas_datos[i][1]) { // EVITAMOS EVALUAR CELDAS NO EVALUABLES

        centro[0]= (array_esquinas_rombos [ (int) array_celdas_datos[i][0] ]) + distancia_esq_disco*ancho;
        centro[1]= (array_esquinas_rombos [ (int) array_celdas_datos[i][1] ]) - distancia_esq_disco;
        centro[2]= (array_esquinas_rombos [ (int) array_celdas_datos[i][2] ]) - distancia_esq_disco*ancho;
        centro[3]= (array_esquinas_rombos [ (int) array_celdas_datos[i][3] ]) + distancia_esq_disco;
        centro[4]= (array_esquinas_rombos [ (int) array_celdas_datos[i][0] ]) + (distancia_esq_disco*ancho*5)/2;

    } // END-IF

    else break; // SALIMOS DEL FOR DE 205 CELDAS

//
// MAIN ALGORITHM MS-SSIM INDEX SEARCH
//

int zone =0; // EL CUADRADO EN EL QUE ESTAMOS, 0 ARRIBA-IZQ, 1 ARRIBA-DCHA, 2 ABAJO-IZDA, 3 ABAJO-DCHA
int search =0; // LA ITERACIÓN DENTRO DEL CUADRADO, ES DECIR, LA ZONA QUE BARREMOS PARA ENCONTRAR EL MSSIM MÁXIMO
int first_corner =0; // EL INICIO DE CADA ZONA, LA ESQUINA SUPERIOR IZQUIERDA
double position =0; // USADO PARA CONOCER LA POSICIÓN DEL MÁXIMO
int disc_diameter=0; //USADO PARA ALMACENAR VALORES INTERMEDIOS DEL DIAMETRO DEL DISCO

double structure_zone [] = {0, 0, 0, 0}; // EL ARRAY DE VALORES DE ESTRUCTURA
double contrast_zone [] = {0, 0, 0, 0}; // EL ARRAY DE VALORES DE CONTRASTE
double luminance_zone [] = {0, 0, 0, 0}; // EL ARRAY DE VALORES DE LUMINANCIA

```

```

double mssim_zone [] = {0, 0, 0, 0}; // EL ARRAY DE VALORES DE MSSIM*
//
// CREACIÓN DE LA IMAGEN QUE TENDRÁ EL CUADRADO Y CON LA QUE BARREMOS
//

double ratio_100micras_pixel = ancho;
ratio_100micras_pixel = ratio_100micras_pixel/1500; // UN ANCHO DE 1500 PÍXELES ES APROXIMADAMENTE 100 MICRAS POR PÍXEL
double number_of_levels = 5;

if (ratio_100micras_pixel < 1.2) ratio_100micras_pixel = 1;
if (ratio_100micras_pixel > 1.2 & (ratio_100micras_pixel < 1.8)) ratio_100micras_pixel = 1.5;
if (ratio_100micras_pixel > 1.8) ratio_100micras_pixel = 2;
if (ratio_100micras_pixel > 1.2) number_of_levels=6;

disc_diameter = (int) array_celdas_datos[i][5];
margen_disco_central = reserva_margen + (number_of_levels-5)*2;
if (disc_diameter > 700) {margen_disco_central = reserva_margen + (number_of_levels-5)*2+ (number_of_levels-4)*2;}
//
// SI SE ENTRA UN 3 EN MARGEN, QUEDA 3-5 EN 100 MICRAS Y 5-9 EN 50 MICRAS
//

apoyo_double = (double) disc_diameter*ratio_100micras_pixel; //(number_of_levels - 4);
apoyo_double = apoyo_double + 50*ratio_100micras_pixel; // (number_of_levels - 4); // AYUDA AL REDONDEO, ASÍ, DIÁMETROS DE 160 MICRAS DAN DIÁMETROS DE DISCO
disc_diameter = (int) (apoyo_double*1.06/100); //MAGNIFICACIÓN DE 1.06 QUE SUCEDE EN MAMOGRAFÍA. SE NOTA EN DIÁMETROS GRANDES

if (disc_diameter <1) disc_diameter=1;

switch (disc_diameter) { // PARA DISCOS PEQUEÑOS NO TIENE SENTIDO REESCALAR
    case 1: number_of_levels=1; break;
    case 2: number_of_levels=2; break; // antes a 1
    case 3: number_of_levels=2; break; // nuevo. antes estaba junto a nivel=1, en la época de Redondoc_CDMAM, cuando se hizo el trabajo de El Cairo
    case 4:
    case 5:
    case 6: number_of_levels=3; break;
    case 7:
    case 8:
    case 9:
    case 10:
    case 11:

```

```

case 12: number_of_levels=4; break;
case 13:
case 14:
case 15:
case 16:
case 17:
case 18:
case 19:
case 20:
case 21:
case 22:
case 24: number_of_levels=5; break;
default: number_of_levels = 6;
}
lado_cuadrado_central = (int) (margen_disco_central*2+disc_diameter);
lado_cuadrado_busqueda = (int) (lado_cuadrado_central + border); // ES DECIR, EL LADO DEL AREA TOTAL DE BÚSQUEDA
apoyo = (int) lado_cuadrado_central/2;
apoyo = apoyo*(ancho+1);
esquina [4] = centro[4] - apoyo;

for (a=0; a<4; a++) {
    apoyo = (int) lado_cuadrado_busqueda/2;
    apoyo = apoyo*(ancho+1);
    esquina [a] = centro[a] - apoyo; // AHORA ESQUINA APUNTA AL INICIO DE CADA ZONA DE ACUERDO CON SU DIÁMETRO
}
int image_original_dimension = lado_cuadrado_central* lado_cuadrado_central;
int image_original_height = lado_cuadrado_central;
int image_original_width = lado_cuadrado_central;

image_width = image_original_width;
image_height = image_original_height;

ImageProcessor image_2_original_p = new ShortProcessor (image_original_height, image_original_width);
short [] array_image_2_original_p = (short []) image_2_original_p.getPixels();

ImagePlus image_1_original_imp = NewImage.createShortImage ("área pequeña", image_original_height, image_original_width, 1, NewImage.FILL_BLACK);
ImageProcessor image_1_original_p = image_1_original_imp.getProcessor();

```

```

Libreria_01_Gabriel.impreferencia.crea (image_1_original_p, lado_cuadrado_central, margen_disco_central, ancho, esquina, disc_diameter );

/**
nuevo.show();
nuevo.updateAndDraw();
*/
for (zone=0;zone<=3; zone++) {

for (search=0; search<iterations;search++) {

apoyo = (int) border+1; // UNA ITERACIÓN MÁS QUE BORDER. BASTA VER EL DIBUJO. REPASAR EN FIND_GRID_CDMAM
int linea = (int) (search/apoyo);
int columna = (int) (search % apoyo);
pointer = esquina[zone] + linea*ancho + columna;

for (b=0; b<lado_cuadrado_central; b++) {
for (c=0; c<lado_cuadrado_central; c++) {
array_image_2_original_p [c+lado_cuadrado_central*b] = array_imagen [pointer+c+ancho*b]; //FALLO
}
}
}

//
// LA IMAGEN 2 SIEMPRE TIENE EL CUADRADO DE BÚSQUEDA EN DISTINTO SITIO
//
image_1_p= image_1_original_p.resize (image_original_width);
image_2_p= image_2_original_p.resize (image_original_width);

//
// WE ARE GOING TO USE ARRAYS OF 7 LEVELS INSTEAD OF 6.
// WE WANT TO FORCE THAT THE INDEX OVER THE LEVEL WERE THE SAME THAN THE INDEX OVER THE ARRAY.
// REMEMBER THAT IN JAVA THE FIRST INDEX OF AN ARRAY IS THE "0" POSITION. WE WILL NEVER USE THIS POSITION IN THE FOLLOWING THREE ARRAYS.
//
// LA SIGUIENTE RUTINA DEBERÍA PROPORCIONAR LOS VALORES L, S Y C DE CUALQUIER IMAGEN QUE LE PASEMOS image_1_p e image_2_p
//
double [] comparison_elements = new double[3];
Libreria_01_Gabriel.lcs.calcula(number_of_levels, image_width, lpf, lpf_width, window_weights, filter_width, bits_per_pixel_1, image_1_p, image_2_p, comparison_elements);

if (comparison_elements[2] > structure_zone [zone] ) {
structure_zone [zone] = comparison_elements[2];
// contrast_zone [zone] = comparison_elements[1];

```

```

// luminance_zone [zone] = comparison_elements[0];
position = search;
}

} // END-FOR ITERACIÓN SEARCH, DENTRO DE CADA ZONA

double zone_d =zone;

} // END-FOR ITERACIÓN DE ZONAS

double max_structure=0, porcentaje_structure = 0, diferencia_structure = 0, second_structure=0;
int position_max_structure=0;

for (a=0; a<4;a++){
    if (structure_zone[a] > max_structure) {
        max_structure=structure_zone[a];
        position_max_structure =a;
    }
}

for (a=0; a<4;a++){
    if (a!=position_max_structure)
        if (structure_zone[a] > second_structure) second_structure=structure_zone[a];
}

diferencia_structure = max_structure-second_structure;
porcentaje_structure = diferencia_structure/max_structure;

centro[0]= (array_esquinas_rombos [ (int) array_celdas_datos[i][0] ]) + distancia_esq_disco*ancho;
centro[1]= (array_esquinas_rombos [ (int) array_celdas_datos[i][1] ]) - distancia_esq_disco;
centro[2]= (array_esquinas_rombos [ (int) array_celdas_datos[i][2] ]) - distancia_esq_disco*ancho;
centro[3]= (array_esquinas_rombos [ (int) array_celdas_datos[i][3] ]) + distancia_esq_disco;

apoyo = (int) lado_cuadrado_búsqueda/2;
apoyo = apoyo*(ancho+1);

a = (int) centro[position_max_structure] - apoyo; // AHORA ESQUINA APUNTA AL INICIO DE CADA ZONA DE ACUERDO CON SU DIÁMETRO
apoyo = (int) border + 1;

```



```

int linea = (int) (position/apoyo);
int columna = (int) (position % apoyo);
a = a + linea*ancho + columna;

for (b=0; b<lado_cuadrado_central; b++) {
    for (c=0; c<lado_cuadrado_central; c++) {
        array_imagen [a+c*ancho*b]=0; // DIBUJA EN NEGRO LOS CUADRADOS QUE CREEMOS QUE ACERTAMOS
    }
}

// FILLS IN WHITE THE CENTRE OF THE ROMBUSES WE HAVE FOUND THE CORRECT CORNER

int true_position = (int) array_celdas_datos[i][4]; // REJILLA VERDADERA
array_celdas_datos[i][13]=0;

if (true_position==position_max_structure) {
    array_celdas_datos[i][13]=1;
    apoyo = (int) (20*ratio_100micras_pixel);
    apoyo_int = (int) (40*ratio_100micras_pixel);

    for (b=0; b<apoyo; b++) {
        for (c=0; c<apoyo; c++) {
            array_imagen [centro0]+c+ancho*(b+apoyo_int)]=-1;
        }
    }

    array_celdas_datos[i][6] = structure_zone[0];
    array_celdas_datos[i][7] = structure_zone[1];
    array_celdas_datos[i][8] = structure_zone[2];
    array_celdas_datos[i][9] = structure_zone[3];
    array_celdas_datos[i][10] = max_structure;
    array_celdas_datos[i][11] = diferencia_structure;
    array_celdas_datos[i][12] = porcentaje_structure;
    array_celdas_datos[i][14] = true_position;
    array_celdas_datos[i][15] = structure_zone[true_position]; //VALOR R* DE LA ESQUINA EN LA QUE ESTÁ EL ROMBO EN REALIDAD. ESO NO SIGNIFICA QUE R* SEA AHÍ MÁX

    double mean_value_background_zone=0;

```

```

for (b=0; b<4; b++) {
    if (b!=true_position) { mean_value_background_zone = mean_value_background_zone+ structure_zone[b];}
}

array_celdas_datos[i][16] = mean_value_background_zone/3;

nuevo.show();
nuevo.updateAndDraw();

if (show_table){
    informe.incrementCounter();
    informe.addValue("Celda", i+results_apoyo);

    if (array_celdas_datos[i][5]!=diametro_anterior){ //AJUSTA
        informe.incrementCounter();
        results_apoyo++;
        informe.addValue("Celda", i+results_apoyo);
        diametro_anterior = array_celdas_datos[i][5];
    }

    if ((i+results_apoyo)>=116){ // COMIENZA A PERDER DIÁMETROS POR LA IZQUIERDA
        informe.incrementCounter();
        results_apoyo++;
        informe.addValue("Celda", i+results_apoyo);
        diametro_anterior = array_celdas_datos[i][5];
    }
}

informe.addValue("Diametro", array_celdas_datos[i][5]);
informe.addValue("Máximo", array_celdas_datos[i][10]);
informe.addValue("Diferencia", array_celdas_datos[i][11]);
informe.addValue("Porcentaje", array_celdas_datos[i][12]);

informe.addValue("Posición disco", array_celdas_datos[i][14]);
informe.addValue("Valor R* en disco", array_celdas_datos[i][15]);
informe.addValue("Media R* en background", array_celdas_datos[i][16]);

informe.addValue("R* en celda 0", array_celdas_datos[i][6]);
informe.addValue("R* en celda 1", array_celdas_datos[i][7]);
informe.addValue("R* en celda 2", array_celdas_datos[i][8]);

```

```

informe.addValue("R* en celda 3", array_celdas_datos[i][9]);

informe.addValue("Acierto", array_celdas_datos[i][13]);
}

} // END-FOR OVER ALL THE CELLS OVER VARIABLE i

if (show_table){
informe.show("Resultado lectura R*");
}

/**
double pepe_d =lpf_width;
//double otro_d=max_structure;
//pepe_d=1;
//otro_d=i;
GenericDialog results = new GenericDialog ("RESULTS: lest and right angles");

//results.addNumericField ("Diferencia absoluta", diferencia_structure, 4, 8, "");
results.addNumericField ("Porcentaje", pepe_d, 4, 8, "");
//results.addNumericField ("Valor absoluto ", otro_d, 4, 8, "");

results.showDialog();
results.dispose();
*/

for (a=0;a<10;a++) System.gc();

} //Fin del Run FIN DE PRINCIPAL
//

} // Fin de la clase

```