IN5520 – Digital Image Analysis

Mandatory 1

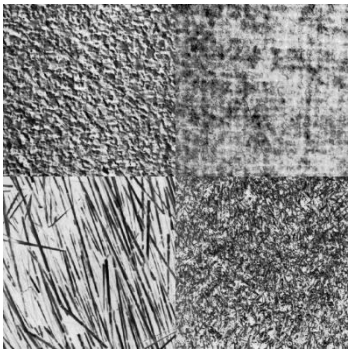
Paul Arquier – paularq@uio.no

Description of the problem

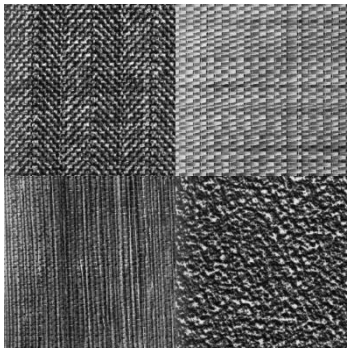
The objective of this assignment is to analyse two images that are mosaics using the GLCM method.

We will apply different parameters in the GLCM analysis in order to discuss the similarities and differences between the different textures of the mosaics.

Here are the 2 images to analyse :



Mosaic 1



Mosaic 2

The area on the top left is area number 1
The area on the top right is area number 2
The bottom left-hand zone is zone number 3
The bottom right-hand zone is zone number 4

A. Analyzing the textures

Mosaic 1

Area 1 : There is no clear direction, hight variance and contrast. The homogeneity is low and the element size is large.

Area 2 : We can see a grid representing the directions to the horizontal and to the vertical. The variance is low. The homogeneity is higher than for the first zone and the element size is large.

Area 3 : Here the image is represented by lines that are mostly pointing in the same direction (which is neither horizontal nor vertical). The variance of the image is very high, the homogeneity is varied and we can see a large element size.

Area 4 : The direction is uniform. The contrast and the variance are hight. We distinguish a low homogeneity and the element size is small

We can see several differences between the four parts of this mosaic. Part 1 and 4 are similar but differ in the size of the elements. Part 3 has a very high variance while part 2 is more easily recognisable.

Mosaic 2

Area 1 : The direction can be defined as a horizontal wave movement. The contrast and variance are high. The homogeneity is low and the element size is small

Area 2 : Several rectangular shapes stacked on top of each other horizontally and vertically. The variance is low. The homogeneity is medium and the size of the elements is large.

Area 3 : We can see vertical lines indicating the direction. The variance is low. High homogeneity and the size of the elements is large.

Area 4 : There is no clear direction, the variance is low. The homogeneity is low and the element size is large.

In mosaic 2, we can differentiate part 1 by its "wave" direction. Parts 2 and 3 are similar but the directions are opposite. Part 4 is composed of coarse patterns and can be easily differentiated from the other parts.

B. Visualizing GLCM matrices

We will use the gray level co-occurrence matrix method in this analysis. The function counts the number of pairs of pixels between each grey level with the distance d and θ as parameters. It inserts the total number into the corresponding position (i,j) of the GLCM matrix. The analysis of the resulting matrix can reveal patterns and structures within the image based on its given parameters, d, θ, size and gray level of image.

Rather than taking continuous value for d at given θ, discrete values, {−45◦ , 0 ◦ , 45◦ , 90◦} are chosen, which cover both diagonals and axis of the image.
Translating the d and θ parameters into dx and dy takes place next. For example θ = 0 will result in dx = d and dy = 0. For θ = −45 the input image is flipped to simplify future indexing.

GLCM is implemented in two nested for-loops, going through all the pixel pair for its parameters, incrementing corresponding index for its pixel pair. Then it is normalized by number of pixel pairs. The resulting GLCM is made symmetrical, counting for both pair (i, j) and (j, i) and then normalized.
In computation, it is achieved by calculating a simple GLCM then adding its transposed pair.

We will now test different values of θ and d to determine the best parameters.
The value of d will directly affect the element size and the contrast.
The objective of this test phase is to find the values that best highlight the differences between the textures.

For Mosaic 1, we will use a θ = -45.
We have tested several values of d and the best results are with d=3

For Mosaic 2, we can choose an θ that is equal to 0 or 90. We will clearly see the differences between the first 3 textures. We will also use the GLCM isotropy to clearly differentiate between textures 1 and 4.
After trying several d-values, the d=4 value gives us the best result to see the differences.

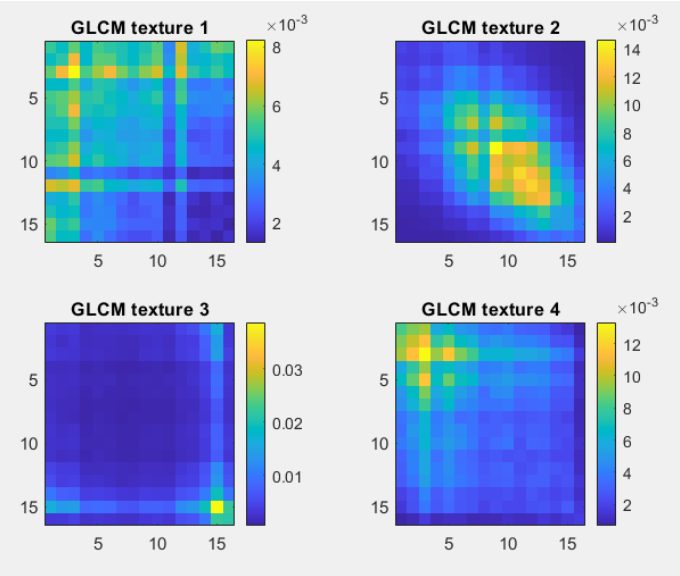


Figure 1 : d = 3, θ = -45 for Mosaic 1

We see in the results above the distribution of the different peaks of each texture. We can notice that texture 3 is different from the other 3

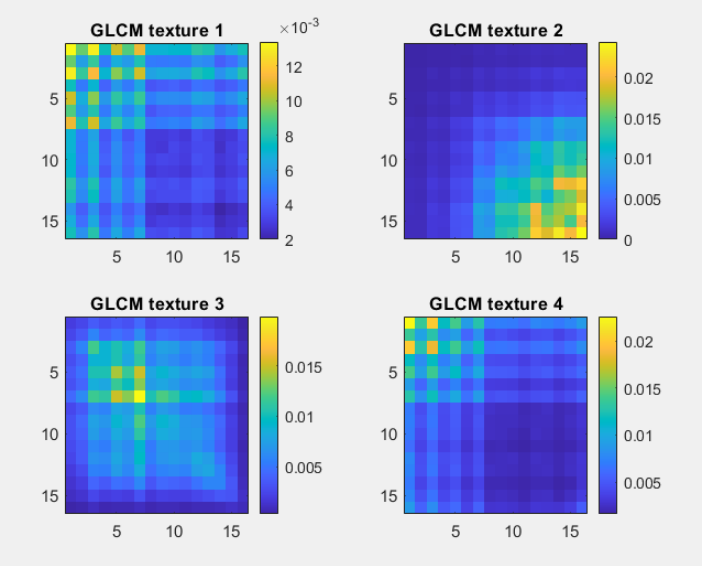


Figure 2 : d = 4, θ = 90 for Mosaic 2

Above, we have the results of the peak distribution for Mosaic 2. We can start to differentiate them from each other, even if texture 1 and 4 remain very similar

C. Computing GLCM feature images in local windows

We will use a sliding window here to run the GLCM at each index. We will then use the functions in the characteristic window statement.
We need to choose a large enough window size to accurately represent the features and give a more uniform shape for each texture.

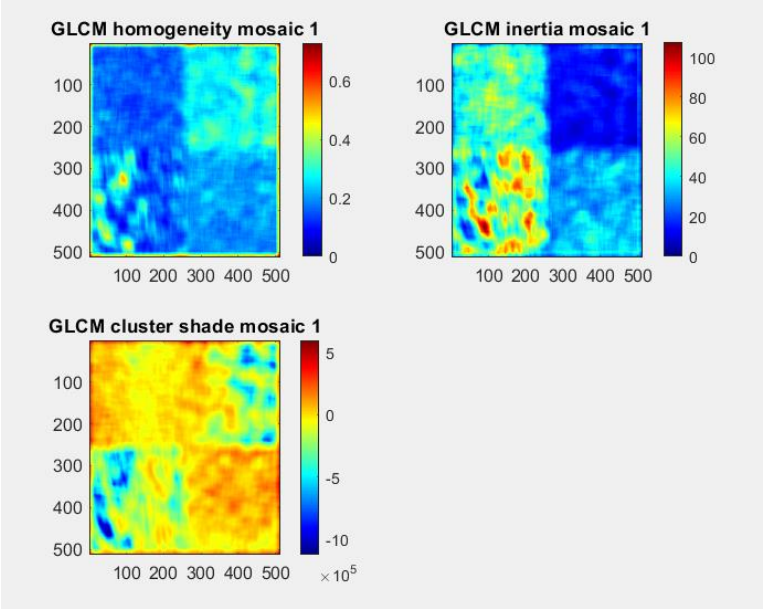


Figure 3 : Features images for Mosaic 1

We see in the images above that textures 1 and 4 are similar in terms of homogeneity and cluster shade. Nevertheless, we can more easily differentiate them thanks to the inertia index. Texture 3 varies and seems difficult to extract with global treshold. As for texture 2, the inertia clearly allows us to differentiate it from the others.

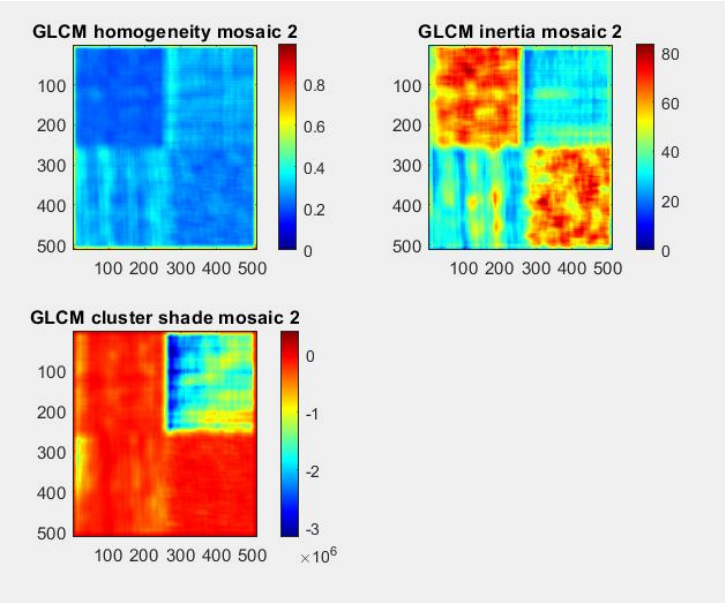


Figure 4 : Features images for Mosaic 2

D. Segment the GLCM feature images and describe how well the texture regions are separated

Thresholding is a type of image segmentation, where we change the pixels of an image to make the image easier to analyze. In thresholding, we convert an image from colour or grayscale into a binary image, i.e., one that is simply black and white.

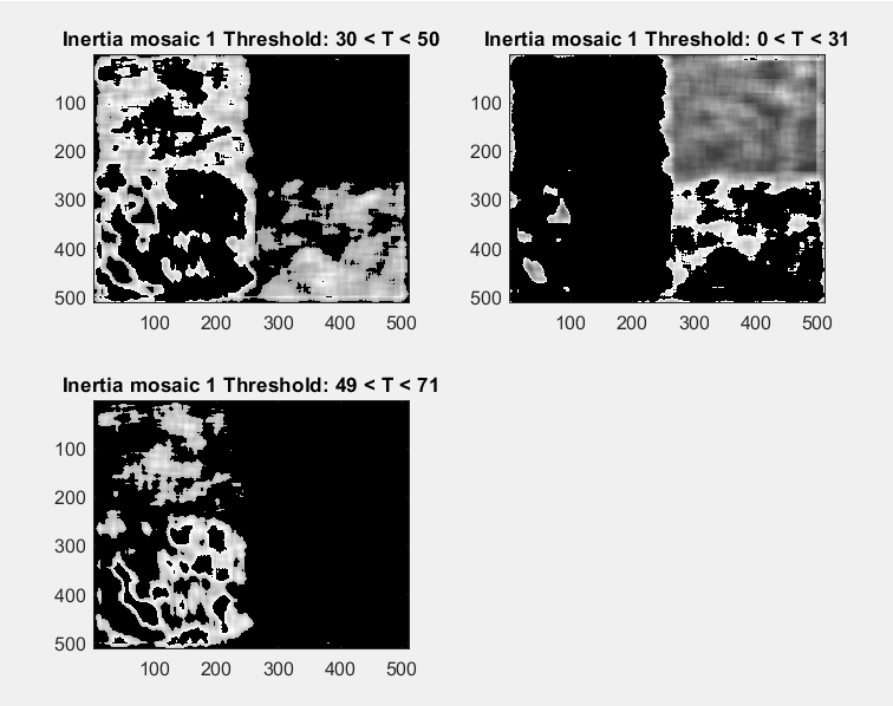


Figure 5 : Thresholds for GLCM Inertia for Mosaic 1

We can see here that the use of treshold allows texture 2 to be separated from the others. It is difficult to differentiate between textures 1 and 3, which remain very similar.

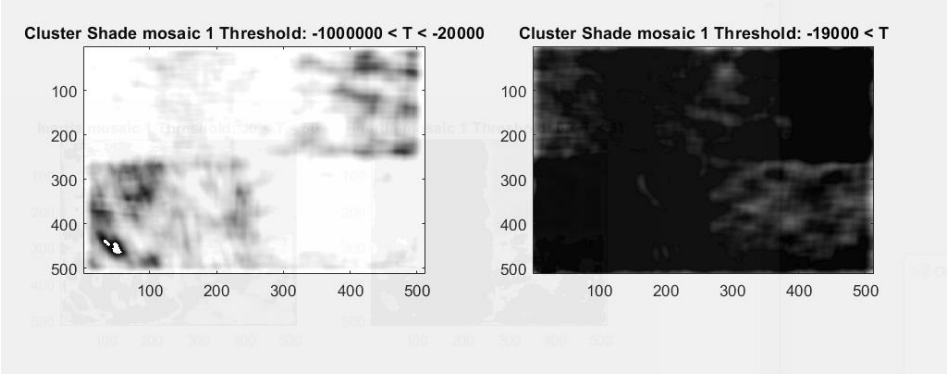


Figure 6 : Thresholds for GLCM cluster shade for Mosaic 1

With the cluster threshold shade, we can group textures 1 and 4 together. Textures 2 and 3 are also similar but this is difficult to exploit.

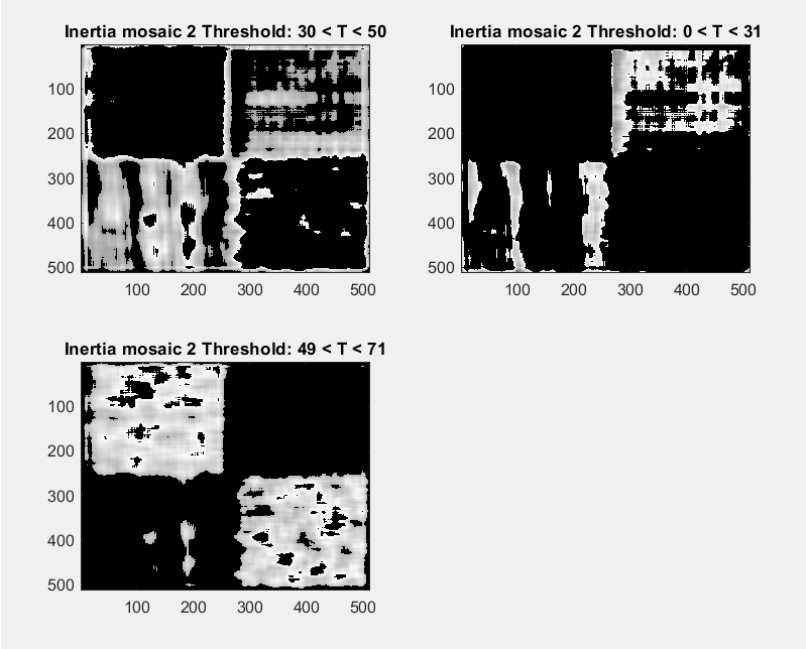


Figure 7 : Thresholds for GLCM Inertia for Mosaic 2

With the Inertia Treshold, we can see that textures 1 and 4 can be differentiated from each other.

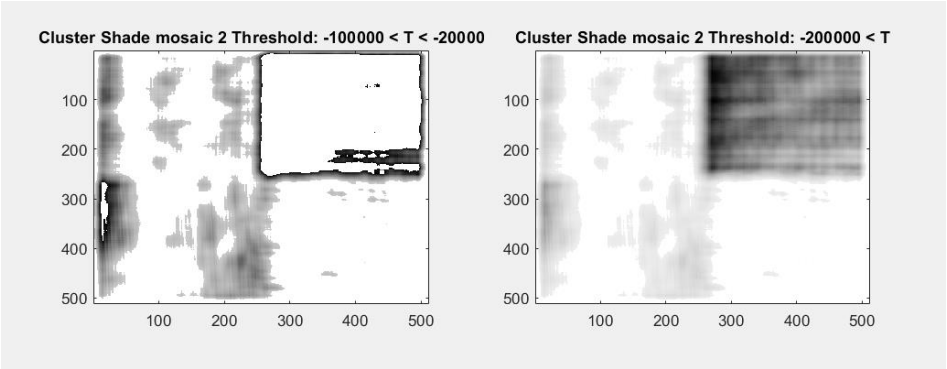


Figure 8 : Thresholds for GLCM cluster shade for Mosaic 2

With the Treshold of cluster shade, we manage to separate texture 2 from the others with a very clear border.

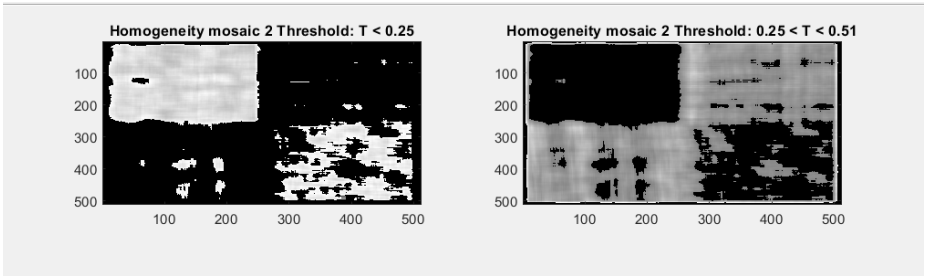


Figure 9 : Thresholds for GLCM homogeneity for Mosaic 2

With the homogeneity threshold, we manage to separate texture 1 from the others with a very clear border. Textures 2 and 3 remain similar and different from the others.

E. Conclusion

We used the GLCM method to analyse different textures of 2 mosaics. We were able to obtain satisfactory results, allowing us to differentiate the textures in some cases. Nevertheless, we can point out some negative aspects of this method, notably the fact that we have to choose very precise parameters to highlight the differences between the textures. In other cases where the textures would be very similar, the parameters would have to be chosen with great precision.

Code

Main.m

```
clc;   % Clear the command window.
clear; % Erase all existing variables.
close all; % Close all figures.

% Here we read the 2 mosaics
mosaic1 = imread('mosaic1.png');
mosaic2 = imread('mosaic2.png');

% Normalize the images
grayscale = 16; % grayscale levels
mosaic1 = histeq(mosaic1, grayscale); % Improve the contrast
% Round each element to the nearest integer
mosaic1 = uint8(round(double(mosaic1)*(grayscale - 1)/double(max(mosaic1(:)))));
mosaic2 = histeq(mosaic2, grayscale); % Improve the contrast
% Round each element to the nearest integer
mosaic2 = uint8(round(double(mosaic2)*(grayscale - 1)/double(max(mosaic2(:)))));

% Divide the mosaic into different area that will represent the differents
% textures
[N,M] = size(mosaic1);
M1Area1 = mosaic1(1:N/2, 1:M/2);
M1Area2 = mosaic1(1:N/2, M/2+1:M);
M1Area3 = mosaic1(N/2+1:N, 1:M/2);
M1Area4 = mosaic1(N/2+1:N, M/2+1:M);
M2Area1 = mosaic2(1:N/2, 1:M/2);
M2Area2 = mosaic2(1:N/2, M/2+1:M);
M2Area3 = mosaic2(N/2+1:N, 1:M/2);
M2Area4 = mosaic2(N/2+1:N, M/2+1:M);

% Display of textures for mosaic 1
d = 3; % delta
theta = -45; % angle
m1t1 = GLCM(M1Area1, grayscale, d, theta);
m1t2 = GLCM(M1Area2, grayscale, d, theta);
```

```
m1t3 = GLCM(M1Area3, grayscale, d, theta);
m1t4 = GLCM(M1Area4, grayscale, d, theta);

figure(1);
subplot(2, 2, 1);
imagesc(m1t1), colorbar, title('GLCM texture 1');
subplot(2, 2, 2);
imagesc(m1t2), colorbar, title('GLCM texture 2');
subplot(2, 2, 3);
imagesc(m1t3), colorbar, title('GLCM texture 3');
subplot(2, 2, 4);
imagesc(m1t4), colorbar, title('GLCM texture 4');

% Display of textures for mosaic 2
d = 4; % delta
theta = 90; % angle
m2t1 = isoGLCM(M2Area1, grayscale, d);
m2t2 = isoGLCM(M2Area2, grayscale, d);
m2t3 = isoGLCM(M2Area3, grayscale, d);
m2t4 = isoGLCM(M2Area4, grayscale, d);

figure(2);
subplot(2, 2, 1);
imagesc(m2t1), colorbar, title('GLCM texture 1');
subplot(2, 2, 2);
imagesc(m2t2), colorbar, title('GLCM texture 2');
subplot(2, 2, 3);
imagesc(m2t3), colorbar, title('GLCM texture 3');
subplot(2, 2, 4);
imagesc(m2t4), colorbar, title('GLCM texture 4');

% Getting the feature images
windowSize = 31;
d=3;
theta=-45;
[IDM1, INR1, SHD1] = glidingGLCM(mosaic1, grayscale, d, theta, windowSize, 0);
d=4;
theta=90;
[IDM2, INR2, SHD2] = glidingGLCM(mosaic2, grayscale, d, theta, windowSize, 1);

figure(3)
colormap jet
subplot(2,2,1)
imagesc(IDM1), colorbar, title('GLCM homogeneity mosaic 1');
subplot(2,2,2)
imagesc(INR1), colorbar, title('GLCM inertia mosaic 1');
subplot(2,2,3)
imagesc(SHD1), colorbar, title('GLCM cluster shade mosaic 1');

figure(4)
colormap jet
subplot(2,2,1)
imagesc(IDM2), colorbar, title('GLCM homogeneity mosaic 2');
subplot(2,2,2)
imagesc(INR2), colorbar, title('GLCM inertia mosaic 2');
subplot(2,2,3)
imagesc(SHD2), colorbar, title('GLCM cluster shade mosaic 2');

% Applying global threshold to the GLCM feature images
% Values chosen manually in order to have the best separation between the textures
% Mosaic 1
figure(5)
colormap gray
subplot(2,2,1)
imagesc(INR1.*(INR1 <= 49 & INR1 >= 31)), title('Inertia mosaic 1 Threshold: 30 < T < 50');
subplot(2,2,2)
```

```
imagesc(INR1.*(INR1 <= 30)), title('Inertia mosaic 1 Threshold: 0 < T < 31');
subplot(2,2,3)
imagesc(INR1.*(INR1 <= 70 & INR1 >= 50)), title('Inertia mosaic 1 Threshold: 49 < T < 71');

figure(6)
colormap gray
subplot(2,2,1)
imagesc(SHD1.*(SHD1 <= -20000 & SHD1 >= -1000000)), title('Cluster Shade mosaic 1 Threshold: -1000000 < T < -20000');
subplot(2,2,2)
imagesc(SHD1.*(SHD1 >= -19000)), title('Cluster Shade mosaic 1 Threshold: -19000 < T');
% Mosaic 2
figure(7)
colormap gray
subplot(2,2,1)
imagesc(INR2.*(INR2 <= 49 & INR2 >= 31)), title('Inertia mosaic 2 Threshold: 30 < T < 50');
subplot(2,2,2)
imagesc(INR2.*(INR2 <= 30)), title('Inertia mosaic 2 Threshold: 0 < T < 31');
subplot(2,2,3)
imagesc(INR2.*(INR2 <= 70 & INR2 >= 50)), title('Inertia mosaic 2 Threshold: 49 < T < 71');

figure(8)
colormap gray
subplot(2,2,1)
imagesc(SHD2.*(SHD2 <= -200000 & SHD2 >= -1000000)), title('Cluster Shade mosaic 2 Threshold: -100000 < T < -20000');
subplot(2,2,2)
imagesc(SHD2.*(SHD2 <= -200000)), title('Cluster Shade mosaic 2 Threshold: -200000 < T');

figure(9)
colormap gray
subplot(2,2,1)
imagesc(IDM2.*(IDM2 <= 0.25)), title('Homogeneity mosaic 2 Threshold: T < 0.25');
subplot(2,2,2)
imagesc(IDM2.*(IDM2 >= 0.26 & IDM2 <= 0.5)), title('Homogeneity mosaic 2 Threshold: 0.25 < T < 0.51');
```

GLCM.m

```
function [glcm] = GLCM(window, grayscale, d, theta)
% GLCM function calculates the GLCM of an image and the result is
% normalized and symmetric.

[N,M] = size(window);
glcm = zeros(grayscale);

% Translating input
if theta == 0
    dx = d;
    dy = 0;
elseif theta == 45
    dx = d;
    dy = d;
elseif theta == 90
    dx = 0;
    dy = d;
elseif theta == -45
    dx = d;
    dy = d;
    %Flip the rows of window
    window = flipud(window);
end

% Counting transitions for indexing
for i = 1:N
    for j = 1:M
        if i + dy > N || i + dy < 1 || i + dx < 1 || ...
           j + dx > M || j + dy < 1 || j + dx < 1
           continue
        end
        first = window(i,j);
        second = window(i + dy, j + dx);
        glcm(first + 1, second + 1) = glcm(first + 1, second + 1) + 1;
    end
```

```
  end

  % Make symmetric and normalize
  glcm = glcm + glcm';
  glcm = glcm/sum(sum(glcm));
  end
```

GlidingGLCM.m

```
function [IDM, INR, SHD] = glidingGLCM(window, grayscale, d, theta, windowSize, iso)
% Calculate the GLCM for every glading window in an image.

[MOriginal, NOriginal] = size(window); % Original image size
HalfSize = floor(windowSize/2); % Size of half the filter

% Apply to the original image the zero-padding
padded = zeros(MOriginal + windowSize - 1, NOriginal + windowSize - 1);
padded(HalfSize:end - HalfSize - 1, HalfSize:end - HalfSize - 1) = window;

[M, N] = size(padded); % Padded image size

% Index matrices
i = repmat((0:(grayscale - 1))', 1, grayscale);
j = repmat(0:(grayscale - 1), grayscale, 1);

IDM = zeros(MOriginal, NOriginal);
INR = zeros(MOriginal, NOriginal);
SHD = zeros(MOriginal, NOriginal);

% Browse image
for m = (HalfSize + 1):(M - HalfSize - 1)
    for n = (HalfSize + 1):(N - HalfSize - 1)
        window = padded(m - HalfSize:m + HalfSize, ...
            n - HalfSize:n + HalfSize);

        % Calculating the GLCM
        if iso == 1
            p = isoGLCM(window, grayscale, d);
        else
            p = GLCM(window, grayscale, d, theta);
        end

        % Calculating the IDM
        IDM(m - HalfSize, n - HalfSize) = sum(sum((1./(1 + (i - j).^2).*p)));

         % Calculating the INR
         INR(m - HalfSize, n - HalfSize) = sum(sum(((i - j).^2).*p));

         % Calculating the SHD
         ux = sum(sum(p.*(i + 1)));
         uy = sum(sum(p.*(j + 1)));
         SHD(m - HalfSize, n - HalfSize) = sum(sum((i + j - ux - uy).^3));
     end
 end
 end
```

isoGLCM.m

```
function [isoGLCM] = isoGLCM(window, grayscale, d)
% isoGLCM calculates the isometric GLCM of an image and the
% result is normalized and symmetric.

isoGLCM = zeros(grayscale);
theta = [0, 45, 90, -45];

for t = 1:length(theta)
    isoGLCM = isoGLCM + GLCM(window, grayscale, d, theta(t));
end

isoGLCM = isoGLCM/3;
end
```