IN5520 - Mandatory Assignment 1

Ada Weinert Ravn, adalw@ulrik.uio.no

October 15, 2019

1 Introduction

In this assignment we will distinguish between textures in a mosaic using GLCMs (Gray Level Co-occurrence Matrices), and feature weights applied to them with given parameters. Two images are provided with textures featuring some similarity but also differences.

2 Analyzing the textures

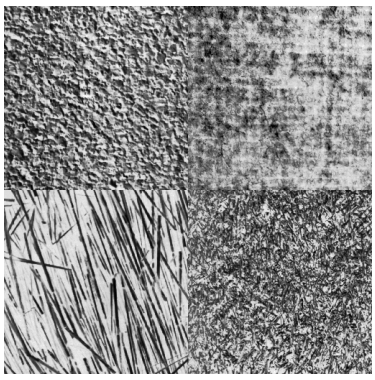

Figure 1: Mosaic 1, alphabetically from left to right, top to bottom.

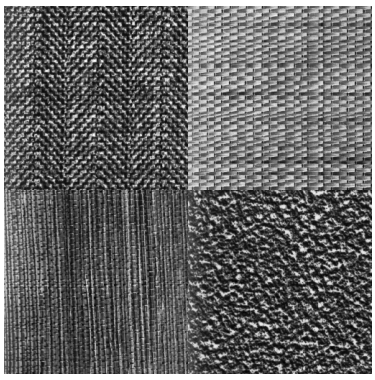

Figure 2: Mosaic 2, alphabetically from left to right, top to bottom.

The goal of this analysis is to compare the textures in the two mosaics for the purpose of discovering the shortest possible list of distinctive defining features to distinguish them by. We'll be using five parameters to describe the different textures: texture direction, frequency, variance, homogeneity and element size.

2.1 Mosaic 1

a) Coarse pattern, with unclear direction, large element size. High contrast and variance and low homogeneity.

b) A grid pattern, with lines stacked horizontally and vertically, large element size. Low contrast resulting in some homogeneity, and low variance.

c) Long line elements, most pointing in same direction. Large element size with nonuniform frequency. Very high contrast and variance with varying homogeneity.

d) Coarse pattern, with uniform direction and small element size. High variance and contrast and low homogeneity.

At the first glance textures a and d look very similar and might be difficult to distinguish, with texture a having a slightly larger element size and pointing at about 45 degree angle, while texture d has a smaller element size and no clear direction. The only clear structure can be seen in texture b easily separating it from the others. Texture c is non uniform, but is characterized by very strong contrast which should aid segmentation.

2.2 Mosaic 2

a) Strong zig-zag pattern at about 45 degrees stacked horizontally, small element size. High contrast and variance.

b) Strong checked pattern, with lines stacked horizontally and vertically, large element size, with high frequency. Lower contrast and variance.

c) Strong vertical line pattern with thinner horizontal lines crossing them. High contrast with lower variance.

d) Coarse pattern with uniform direction. Low homogeneity, and variance, medium contrast.

Here as well the textures a and d look very similar, but there is a structure in texture a at +/- 45 degree angle to aid segmentation. Textures b and c also contain strong structures, so choosing the correct angle will help differentiate between them.

3 Visualizing GLCM matrices

3.1 Methods

First the contrast in each image is enhanced using histeq, and both are normalized to $G = 2^4$ gray levels. This is to make the features more distinguishable and consistent. Afterwards the images are manually split into the different textures using index operations for the preliminary visualization of textures.

The matrices are visualised using GLCM (Gray Level Co-occurrence Matrix), using both isometric and regular version, each defined in their own function. The function counts the number of pixel pairs between each gray level at a distance d, and angle θ for the whole image, and inserts the number in the corresponding (i, j) position in the GLCM matrix which starts initialized with zeros. The analysis of the resulting matrix can reveal patterns and structures within the image based on its parameters.

Instead of taking a continuous θ, I've approximated it to 4 values [0, 45, 90, -45], which cover both diagonals and axis of the image. Translating the d and θ parameters into dx and dy takes place next. For instance $\theta = 0$ will result in $dx = d$ and $dy = 0$. For $\theta = -45$ the input image is flipped to simplify future indexing.

Finally going through double loops we're counting each pixel pair for the provided parameters. Here some indexing was necessary. The resulting GLCM is made symmetrical, counting for both pair (i, j) and (j, i) and then normalized.

Isotropic GLCM calculates the GLCM for each of the 4 theta values, and divides the values in the matrix by the amount of angles tested. This helps compare similar textures which are slightly differently angled.

3.2 Parameter selection

The prior analysis of textures allows for a shorter list of test variables. To account for the direction of the texture different θ values will be used, and to account for element size, frequency and contrast we choose different values of d.

For Mosaic 1 to distinguish between texture a and d a θ of -45 will be used. Values for d tested have been 2, 3 and 4 with 3 giving the biggest difference in the GLCMs.

For Mosaic 2 it is best to use isotropic GLCM to distinguish between textures a and d due to the +/- 45 degree structure elements in texture a, and because of comparable element size in both textures. the d values tested between 2 and 5 show that the value 4 gives the most distinguishable GLCMs.

3.3 Resulting GLCM matrices

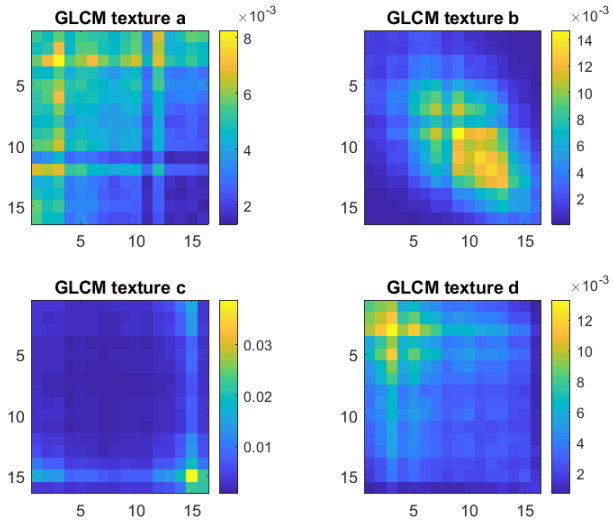

Figure 3: Mosaic 1: $d = 3$, $\theta = -45$.

Figures 3 and 4 show the GLCMs of manually split textures in mosaics (Figure 1 and 2).

The GLCMs in Figure 3 show a good distribution of peak values and distribution which will aid in segmentation. Textures a and d look the most similar, but texture a has more high peak values, while the values in texture d are distributed more evenly.

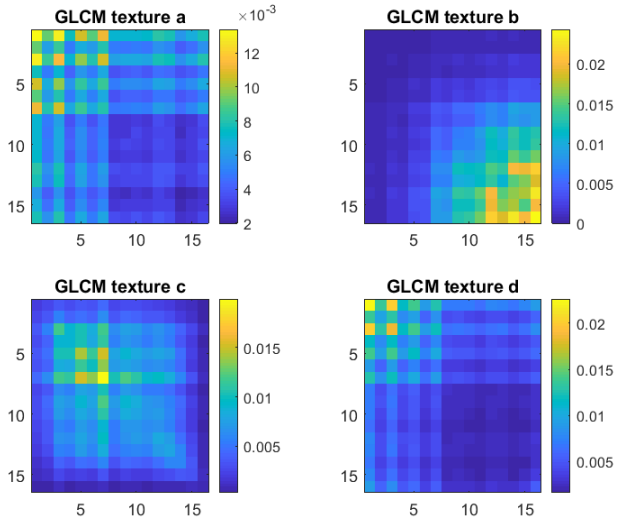

Figure 4: Mosaic 2: $d = 4$, isotropic.

The GLCMs of textures b and c in Figure 4 distinguish themselves quite well from each other and the remaining textures. The GLCM of texture a and d unfortunately looks very similar, even when using the isotropic calculation.

4 Computing GLCM feature images in local windows

4.1 Methods

First the contrast in each image is enhanced using histeq, and both are normalized to $G = 2^4$ gray levels. Next the image is 0-padded with 'half' the window size. A larger window will give a more precise estimation of features and therefore a window size of 31 has been used. A gliding GLCM window is implemented, gliding over the whole image and calculating a GLCM along with the values for the feature weights at each index. The feature windows are

calculated using the equations suggested in the exercise text. Index matrices using repmat have been applied for simpler notation.

4.2 Resulting GLCM feature images

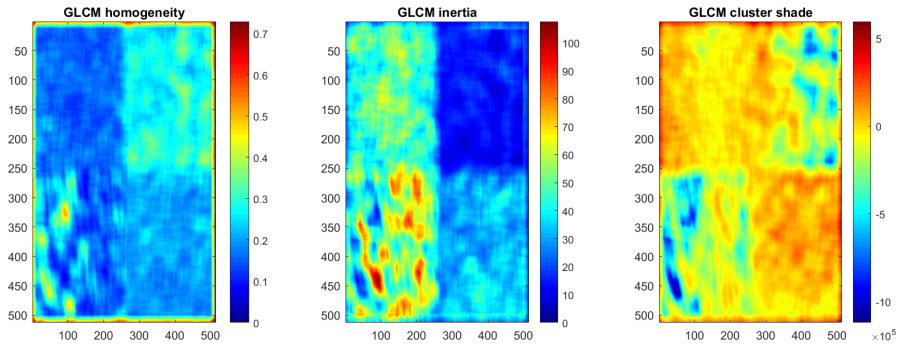

Figure 5: Feature images for Mosaic 1.

As expected from assumptions the texture b in Mosaic 1 is easily distinguishable from the others in the feature images, particularly with regards to inertia, and to a lesser degree homogeneity, as seen in Figure 5. Textures a and d remain similar although marginally distinguishable, especially with regards to inertia, which is as expected.

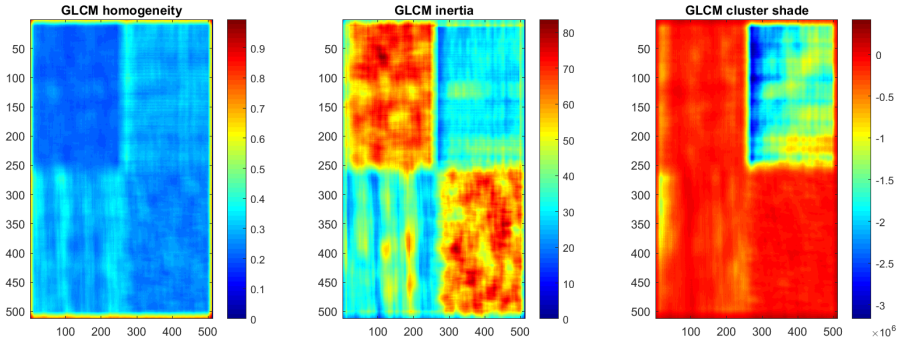

Figure 6: Feature images for Mosaic 2.

In Figure 6 we can see that the textures in Mosaic 2 are more cleanly divided using the feature images compared to Mosaic 1. The cluster shade clearly distinguishes texture b from all the others, just as expected. The textures b and c can be distinguished by their directions: horizontal and vertical respectively.

Textures a and d remain similar, but can likely be distinguished using homogeneity.

5 Threshold of the GLCM feature images

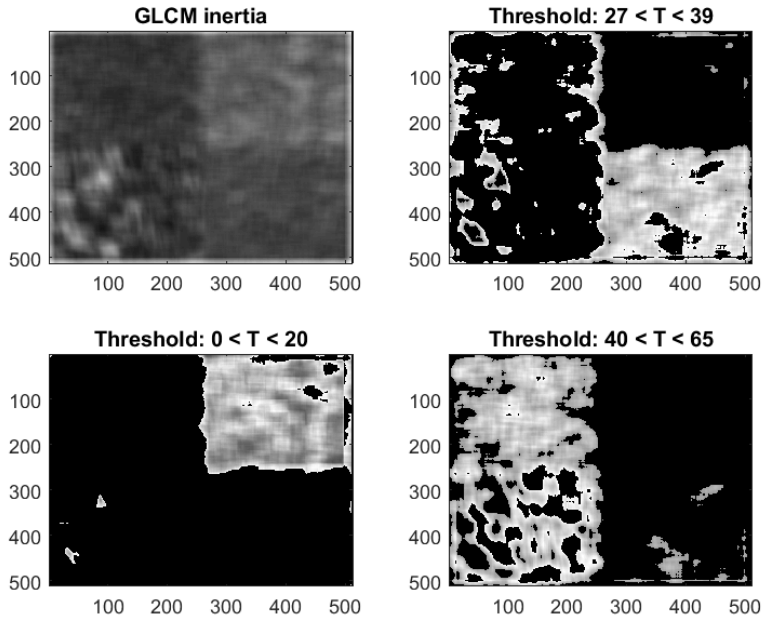

Figure 7: Thresholds for Mosaic 1.

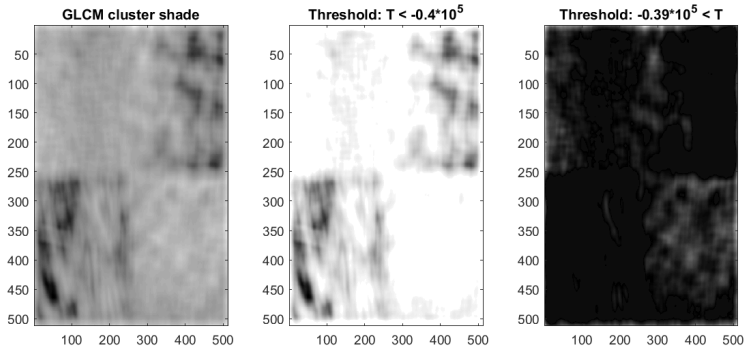

Figure 8: Thresholds for Mosaic 1.

In Figures 7 and 8 we can see the examples of thresholds used to separate the textures in Mosaic 1. The cleanest threshold can be attributed to texture b by threshold of inertia. It overlaps the texture c in two spots and contains some holes which should be covered, the borders are uneven but approximately correct. Threshold of homogeneity can also be used to find texture b, but the former results in a cleaner image. Threshold of inertia also allows for distinguishing texture d although it contains spots within and borders of textures a and c, the reverse also applies. Threshold of cluster shade can somehow divide textures a and d from b and c although very messily. The borders aren't even, big chunks of textures are missing and there is some overlap with texture a and d.

Texture c remains the most difficult to distinguish but can potentially be achieved by for instance lowest and highest values in homogeneity, highest in inertia and lower values of cluster shade not contained in texture b.

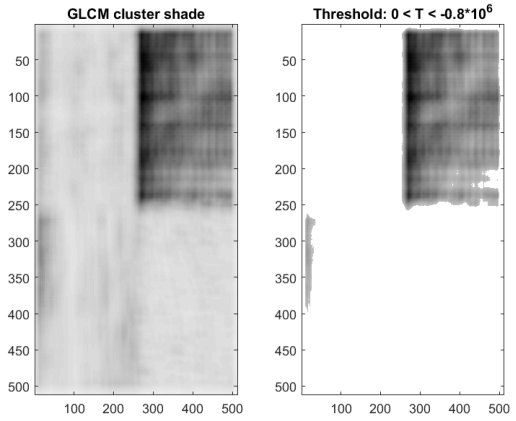

Figure 9: Threshold of cluster shade for Mosaic 2.

In Figures 9, 10 and 11 we can see the examples of thresholds used to separate the textures in Mosaic 1. The cleanest of those can be seen for texture b in cluster shade (Figure 9), the left border is split very cleanly, but some of the texture is missing on the bottom border. A bit of texture c is also caught within the threshold. Textures a and d can be for the most part separated from the others apart from three spots in texture c bu threshold of inertia as seen in Figure 11. By extension the opposite also applies as seen in the same figure. While texture a can be separated from others by threshold of homogeneity, it does not capture the whole texture, and captures specs of texture d as well as seen in Figure 10.

Not a single feature can distinguish between all textures and not even all of

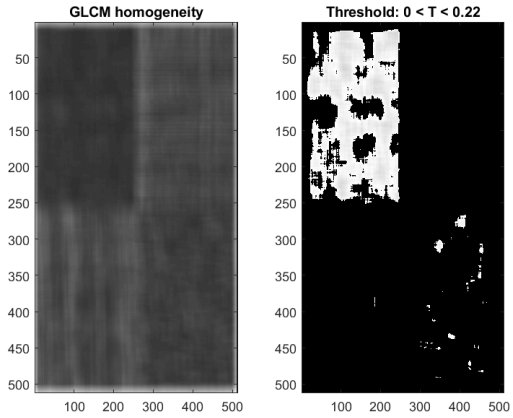

Figure 10: Threshold of homogeneity for Mosaic 2.

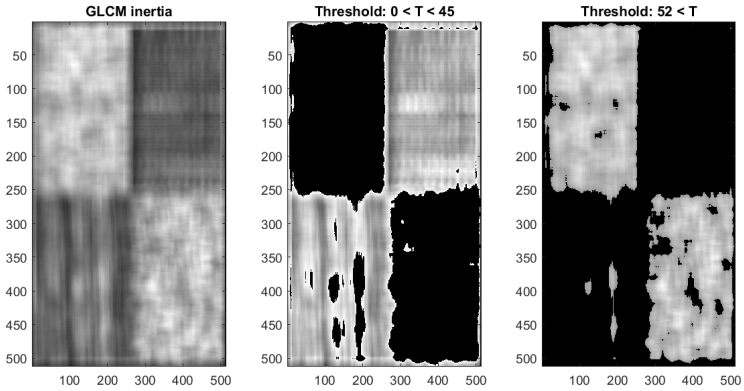

Figure 11: Threshold of inertia for Mosaic 2.

them each on their own. Some of the textures can be threshold easier and help the others be found by exclusion, or partially though different features. As an example in Mosaic 2 we cannot clearly find texture c by itself but we can find a threshold for texture c and b together in inertia as well as just texture b in cluster shade. We can then apply a filter for a texture in inertia to apply for the whole are not contained in the mask from Figure 9. AN example of that can be seen in Figures 12, 13, 14 and 15 below.

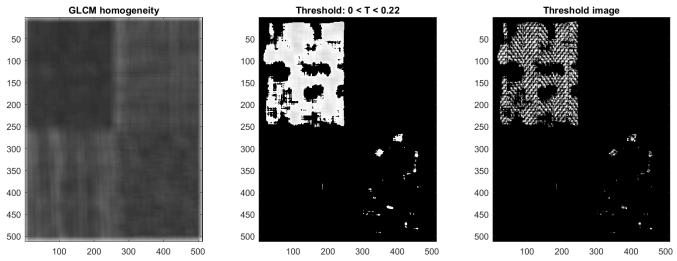

Figure 12: Texture a for Mosaic 2.

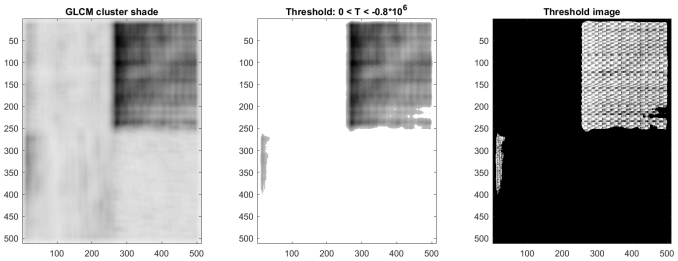

Figure 13: Texture b for Mosaic 2.

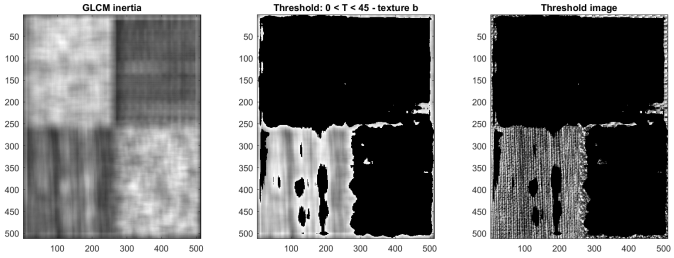

Figure 14: Texture c for Mosaic 2.

6 Conclusion

Given a finite set of variables, clearly defined problem and target images obtained from controlled environment clearly distinguishing between the limited textures would be possible to automate with careful choice of parameters.

For a general problem though it is not viable. The angle of the texture matters, some textures are very similar to each other and tedious to find correct

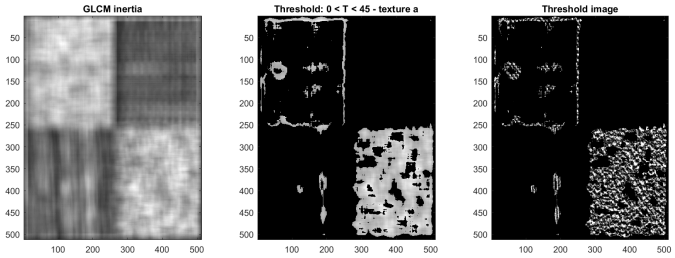

Figure 15: Texture d for Mosaic 2.

parameters for, or straight up indistinguishable using the given method. Scale will affect how the GLCM of a texture looks as well.

7 Appendix

```
Mandatory1.m
clear all;
close all;

% Reading images
mosaic1 = imread('mosaic1.png');
mosaic2 = imread('mosaic2.png');

% Normalizing the images
G = 2^4; % grayscale levels
mosaic1 = histeq(mosaic1, G);
mosaic1 = uint8(round(double(mosaic1)*(G - ...
    1)/double(max(mosaic1(:)))));
mosaic2 = histeq(mosaic2, G);
mosaic2 = uint8(round(double(mosaic2)*(G - ...
    1)/double(max(mosaic2(:)))));

% Splitting mosaics into separate textures
[N,M] = size(mosaic1);
t1a = mosaic1(1:N/2, 1:M/2);
t1b = mosaic1(1:N/2, M/2+1:M);
t1c = mosaic1(N/2+1:N, 1:M/2);
t1d = mosaic1(N/2+1:N, M/2+1:M);
t2a = mosaic2(1:N/2, 1:M/2);
t2b = mosaic2(1:N/2, M/2+1:M);
t2c = mosaic2(N/2+1:N, 1:M/2);
t2d = mosaic2(N/2+1:N, M/2+1:M);

% Visualizing the textures for mosaic 1
```

```
d = 3; % Δ
theta = −45; % angle
v1a = GLCM(t1a, G, d, theta);
v1b = GLCM(t1b, G, d, theta);
v1c = GLCM(t1c, G, d, theta);
v1d = GLCM(t1d, G, d, theta);

figure(1);
colormap parula
subplot(2, 2, 1);
imagesc(v1a), colorbar, title('GLCM texture a');
subplot(2, 2, 2);
imagesc(v1b), colorbar, title('GLCM texture b');
subplot(2, 2, 3);
imagesc(v1c), colorbar, title('GLCM texture c');
subplot(2, 2, 4);
imagesc(v1d), colorbar, title('GLCM texture d');

% Visualizing the textures for mosaic 2
d = 4; % Δ
theta = 90; % angle
v2a = isoGLCM(t2a, G, d);
v2b = isoGLCM(t2b, G, d);
v2c = isoGLCM(t2c, G, d);
v2d = isoGLCM(t2d, G, d);

figure(2);
colormap parula
subplot(2, 2, 1);
imagesc(v2a), colorbar, title('GLCM texture a');
subplot(2, 2, 2);
imagesc(v2b), colorbar, title('GLCM texture b');
subplot(2, 2, 3);
imagesc(v2c), colorbar, title('GLCM texture c');
subplot(2, 2, 4);
imagesc(v2d), colorbar, title('GLCM texture d');

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

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

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

```
subplot(1,3,3)
imagesc(SHD2), colorbar, title('GLCM cluster shade');

% Applying global threshold to the feature images
figure(5)
colormap gray
subplot(2,2,1)
imagesc(IDM1), title('GLCM inertia');
subplot(2,2,2)
imagesc(INR1.*(INR1 ≤ 39 & INR1 ≥ 23)), title('Threshold: 27 < T ...
    < 39');
subplot(2,2,3)
imagesc(INR1.*(INR1 ≤ 20)), title('Threshold: 0 < T < 20');
subplot(2,2,4)
imagesc(INR1.*(INR1 ≤ 65 & INR1 ≥ 40)), title('Threshold: 40 < T ...
    < 65');

figure(6)
colormap gray
subplot(1,3,1)
imagesc(SHD1), title('GLCM cluster shade');
subplot(1,3,2)
imagesc(SHD1.*(SHD1 ≤ −0.4*10^5)), title('Threshold: T < ...
    −0.4*10^5');
subplot(1,3,3)
imagesc(SHD1.*(SHD1 ≥ −0.38*10^5)), title('Threshold: −0.39*10^5 ...
    < T');


figure(12)
colormap gray
subplot(1,3,1)
imagesc(SHD2), title('GLCM cluster shade');
subplot(1,3,2)
imagesc(SHD2.*uint8(SHD2 ≤ −0.8*10^6)), title('Threshold: 0 < T ...
    < −0.8*10^6')
subplot(1,3,3)
imagesc(mosaic2.*uint8(SHD2 ≤ −0.8*10^6)), title('Threshold image')

figure(11)
colormap gray
subplot(1,3,1)
imagesc(IDM2), title('GLCM homogeneity');
subplot(1,3,2)
imagesc(IDM2.*(IDM2 ≤ 0.22)), title('Threshold: 0 < T < 0.22');
subplot(1,3,3)
imagesc(mosaic2.*uint8(IDM2 ≤ 0.22)), title('Threshold image');

figure(14)
colormap gray
subplot(1,3,1)
imagesc(INR2), title('GLCM inertia');
subplot(1,3,2)
imagesc(INR2.*(INR2 ≥ 52 & IDM2 ≥ 0.23)), title('Threshold: 0 < ...
    T < 45 − texture a');
subplot(1,3,3)
imagesc(mosaic2.*uint8(INR2 ≥ 52 & IDM2 ≥ 0.23)), ...
```

```
        title('Threshold image');

figure(13)
colormap gray
subplot(1,3,1)
imagesc(INR2), title('GLCM inertia');
subplot(1,3,2)
imagesc(INR2.*(INR2 ≤ 45 & SHD2 ≥ −0.79*10^6)), ...
        title('Threshold: 0 < T < 45 − texture b');
subplot(1,3,3)
imagesc(mosaic2.*uint8(INR2 ≤ 45 & SHD2 ≥ −0.79*10^6)), ...
        title('Threshold image');
```

```
GLCM.m
function [glcm] = GLCM(img, G, d, theta)
% GLCM calculates the GLCM (Gray Level Coocurrence Matrices) of ...
    an image.
% The result is normalized and symmetric.

[N,M] = size(img);
glcm = zeros(G);

% 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;
    img = flipud(img);
end

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

% Make symmetric
```

```
glcm = glcm + glcm';

% Normalize
glcm = glcm/sum(sum(glcm));
end
```

```
function [isoglcm] = isoGLCM(img, G, d)
% isoGLCM calculates the isometric GLCM (Gray Level Coocurrence ...
    Matrices)
% of an image. The result is normalized and symmetric.

isoglcm = zeros(G);
thetas = [0, 45, 90, -45];

for t = 1:length(thetas)
    isoglcm = isoglcm + GLCM(img, G, d, thetas(t));
end

isoglcm = isoglcm/3;
end
```

```
function [IDM, INR, SHD] = glidingGLCM(img, G, d, theta, w_s, iso)
% Calculate the GLCM for every glading window in an image. It ...
    adds a frame
% to the image first so that the resulting image is the same ...
    size as input.
% iso : 1 for isometric GLCM, 0 otherwise

[M_o, N_o] = size(img); % Original image size
h_s = floor(w_s/2); % Size of half the filter

% Apply the zero-padding to the original image
imgPadded = zeros(M_o + w_s - 1, N_o + w_s - 1);
imgPadded(h_s:end - h_s - 1, h_s:end - h_s - 1) = img;

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

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

% Buffers for resulting images
IDM = zeros(M_o, N_o);
INR = zeros(M_o, N_o);
SHD = zeros(M_o, N_o);

% Go through the image
for m = (h_s + 1):(M - h_s - 1)
    for n = (h_s + 1):(N - h_s - 1)

```

```
28          % Extracting the window
29          window = imgPadded(m - h_s:m + h_s, ...
30              n - h_s:n + h_s);
31
32          % Calculating the GLCM
33          if iso == 1
34              p = isoGLCM(window, G, d);
35          else
36              p = GLCM(window, G, d, theta);
37          end
38
39          % Calculating the homogeneity, IDM (Inverse Difference ...
                Moment)
40          IDM(m - h_s, n - h_s) = sum(sum((1./(1 + (i - j).^2).*p)));
41
42          % Calculating the inertia, INR
43          INR(m - h_s, n - h_s) = sum(sum(((i - j).^2).*p));
44
45          % Calculating the cluster shade, SHD
46          ux = sum(sum(p.*(i + 1)));
47          uy = sum(sum(p.*(j + 1)));
48          SHD(m - h_s, n - h_s) = sum(sum((i + j - ux - uy).^3));
49      end
50  end
51  end
```