# Solution to the exercises for week 2

09/09/2015 (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/)
omrindal (http://inf4300.olemarius.net/author/omrindal/)
14 Comments (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/#comments)

## Contents

## Week 2 : GLCM exercise

In this exercise we will implement GLCM texture features. This code is strongly linked to the lecture about texture so you need to study this to be able to understand this code. Please let me know about any errors in the code

I have structured all the code for this week into three functions. The CLCMExercise is the "main" function calling and running the two other funtions glidingGLCM and GLCM. See the other functions for more detailed descriptions.

This code can be downloaded from https://github.com/olemarius90/INF4300

```
function GLCMExercise()
```

```
%clear all
close all
%Set this to one if you want to calculate the GLCM
% set it to zero if it has allready been calculated
calculateGLCM = 0;
```

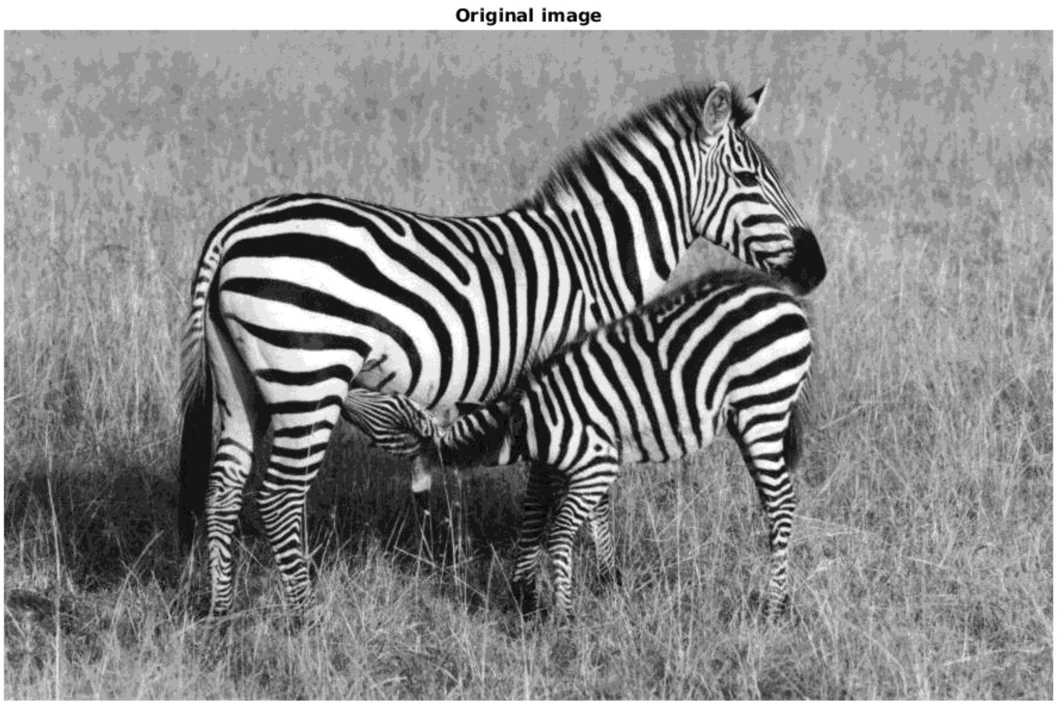
## Task 1 : GLCM implementation

This task is to implement code Testing my GLCM implementation

```
testImage = [0 0 1 1; 0 0 1 1; 0 2 2 2; 2 2 3 3]
disp('NB! You have to scroll down to the end of the page to see the output');
GLCM1 = GLCM(testImage,4,1,0,0,0);
GLCM2 = GLCM(testImage,4,0,1,0,0);
disp('Parameters: dx = 1, dy = 0, gives this result:')
disp(GLCM1)
disp('Parameters: dx = 0, dy = 1, gives this result:')
disp(GLCM2)

% Read the input image
img = imread('zebra_2.tif');

% All zebra images have 2^4 = 16 possible intensities. 'zebra_1.tif' stores
% these using an 8 bits intensity range (the other images uses 4 bits as
% they should do). To convert the read 'zebra_1.tif' to use a 4 bits
% intensity range, use e.g.:

%img = uint8(floor(double(img)/16)); % only when using 'zebra_1.tif'
figure(1);
imshow(img, []); % notice the black-and-white frame in 'zebra_1.tif'
title('Original image');

G = 8; % We just want to use G gray levels

% Make the histogram (approx.) uniform with G grey levels.
% See the curriculum for INF2310 if you do not know histogram equalization
img_std = histeq(img,G);
img_std = uint8(round(double(img_std) * (G-1) / double(max(img_std(:)))));

figure(2);
imshow(img_std, [0 G-1]);
title('Image after histogram equalization');
% Lets look at the histograms of the original image and the image after
% histogramequalization
figure(3);
subplot(211)
h = imhist(img);
plot([0:255],h/sum(h),'linewidth',2);
xlim([0 15])
title('Normalized histogram of original image');
subplot(212)
h = imhist(img_std);
plot([0:255],h/sum(h),'linewidth',2);
xlim([0 15])
title('Normalized histogram of image after histeq');

disp('Original image consists of these values:')
unique(img)'
disp('Image after histeq and normalization has these values:');
unique(img_std)'

% Define GLCM-parameters.
windowSize = 31;
dx = 0;
dy = 2;

% Call the function to calculate the feature images with gliding GLCM
if calculateGLCM == 1
    [glcmVar,glcmCtr,glcmEnt] = glidingGLCM(img_std,G,dx,dy,windowSize);
    save('Data.mat','glcmVar','glcmCtr','glcmEnt','windowSize','G');
```

```
else
    load Data.mat;
end
```

```
testImage =

     0     0     1     1
     0     0     1     1
     0     2     2     2
     2     2     3     3

NB! You have to scroll down to the end of the page to see the output
```


Original image

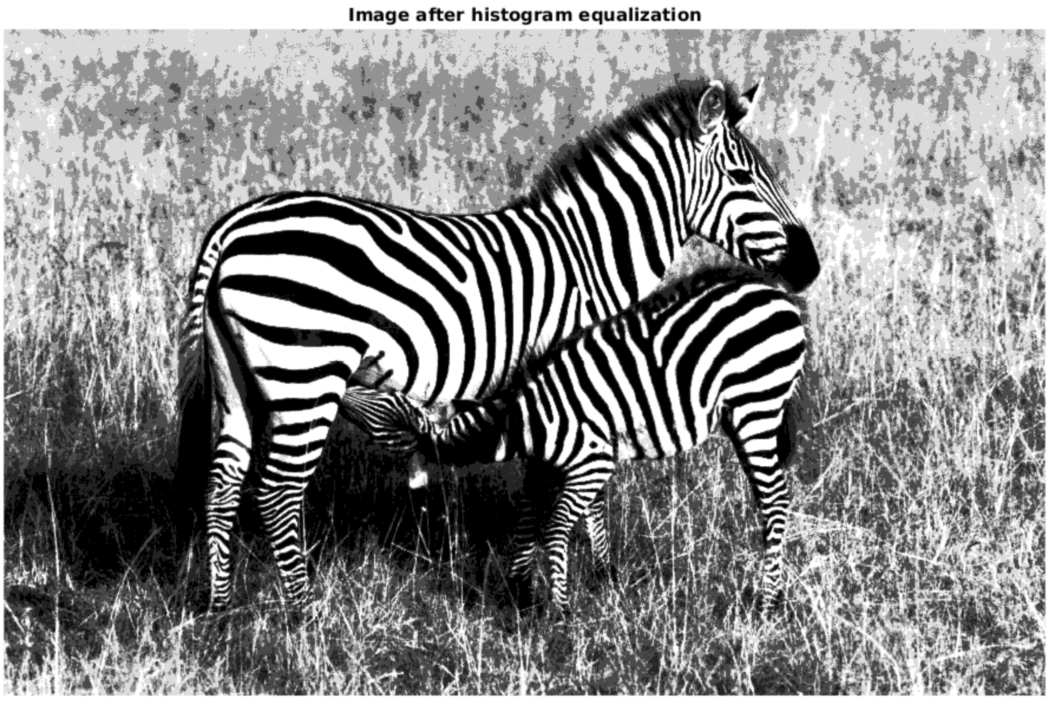
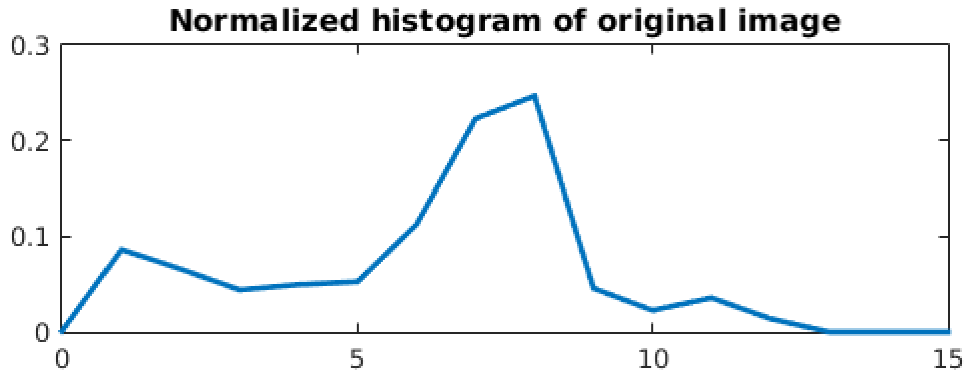
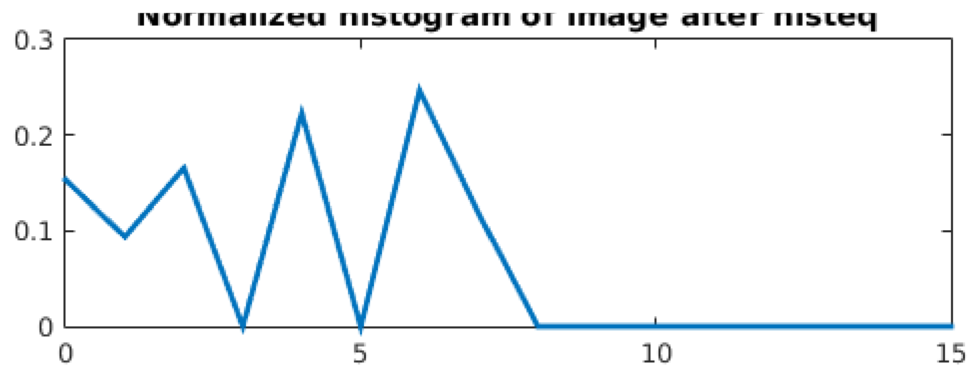


Image after histogram equalization


Normalized histogram of original image
0.3
0.2
0.1
0
0
5
10
15



## Task 2 : Treshold the GLCM feature images

We want to treshold the resulting GLCM variance, GLCM contrast and GLCM entropy

```
% Display the results
figure(4);clf
subplot(211)
imshow(glcmVar, []); title('GLCM Variance');
subplot(212)
imshow(img.*uint8(glcmVar > (max(glcmVar(:)) * 0.5)),[]);
title('Image thresholded with GLCM Variance');

figure(5);clf
subplot(211)
imshow(glcmCtr, []); title('GLCM Contrast');
subplot(212)
imshow(img.*uint8(glcmCtr > (max(glcmCtr(:)) * 0.2)),[]);
title('GLCM Contrast thresholded');

figure(6);clf
subplot(211)
imshow(glcmEnt, []); title('GLCM Entropy' );
subplot(212)
imshow(img.*uint8(abs(glcmEnt) < (max(abs(glcmEnt(:))) * 0.7)),[]);
title('GLCM Entropy tresholded');
```

GLCM Variance

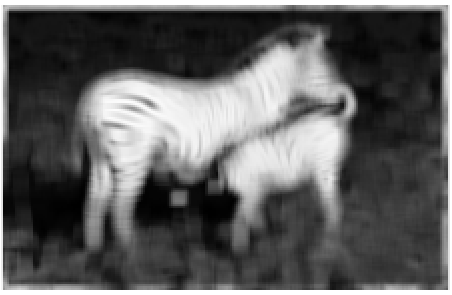
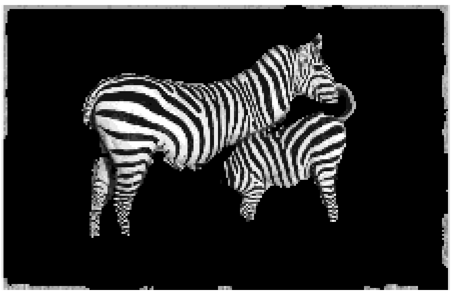
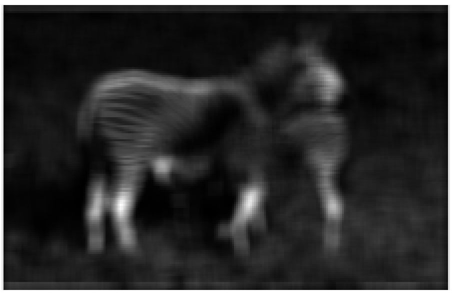


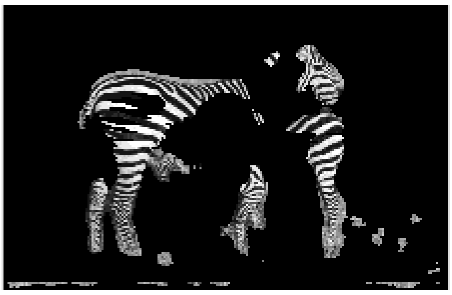
Image thresholded with GLCM Variance



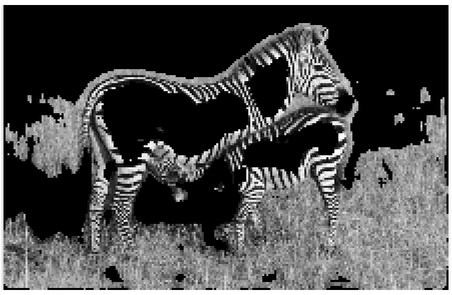
GLCM Contrast



GLCM Contrast thresholded



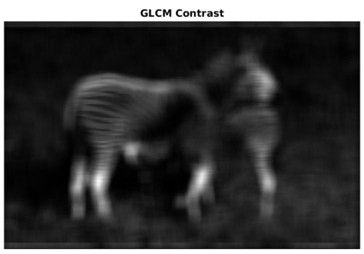
GLCM Entropy

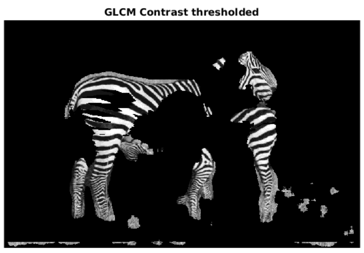


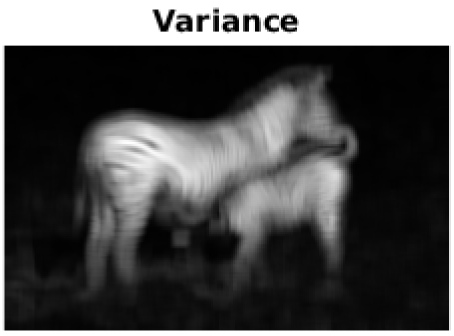
## Task 3: Compare with first order texture measurements

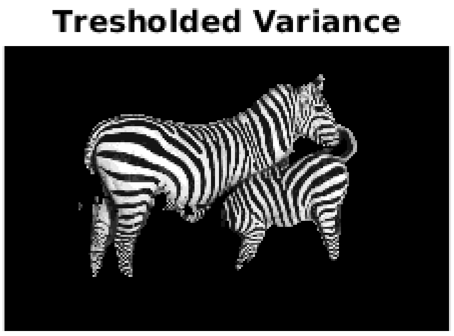
Here we want to use the first order texture measures variance and entropy using the built in matlab functions stdfilt and entropyfilt

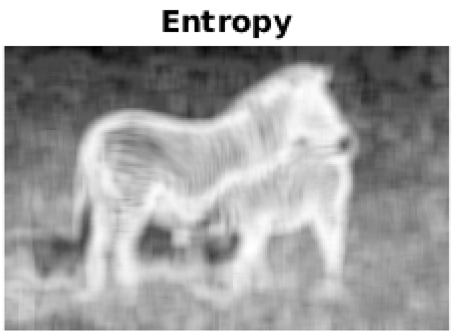
```
% Remember that the variance is the square of the standard deviation
std_var = stdfilt(img, ones(windowSize)).^2;
figure(7); subplot(211); imshow(std_var, []); title('Variance');
subplot(212), imshow(img.*uint8(std_var > (max(std_var(:)) * 0.25)),[]);
title('Tresholded Variance')

% See page 532 in Gonzales and woods for a definition on entropy ("average
% information")
std_ent = entropyfilt(img, ones(windowSize));
figure(8); subplot(211); imshow(std_ent, []); title('Entropy' );
subplot(212), imshow(img.*uint8(std_ent > (max(std_ent(:)) * 0.6)),[])
title('Tresholded entropy')

% Variance on image after histeq
std_var_of_img_std = stdfilt(img_std, ones(windowSize)).^2;
figure(9), subplot(211), imshow(glcmVar, []), title('GLCM Variance');
subplot(212), imshow(std_var_of_img_std, []), title('Variance');
```


GLCM Contrast

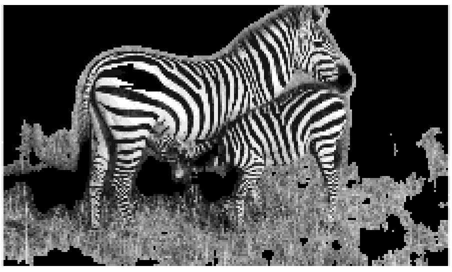

GLCM Contrast thresholded


Variance

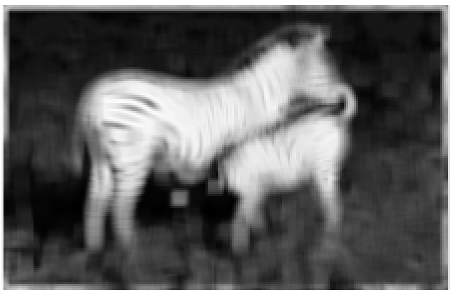

Tresholded Variance


Entropy

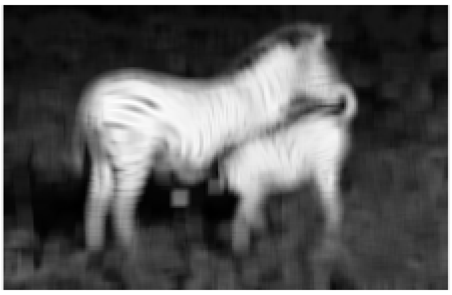
Tresholded entropy



GLCM Variance



Variance



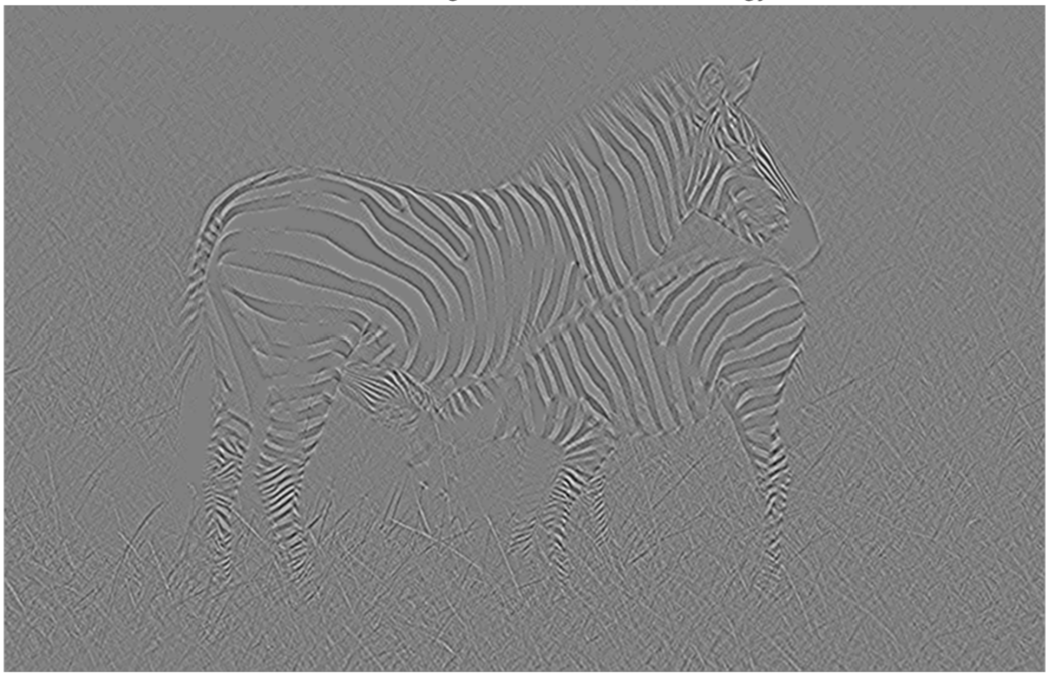
## Task 4: Using Laws texture mask to find texture features

See the lecture foils for more info on Laws texture masks

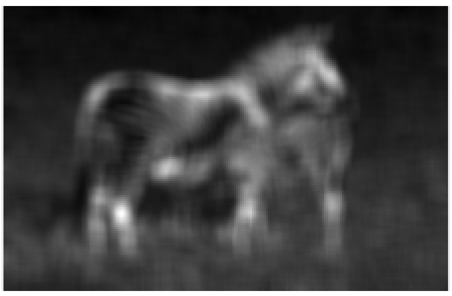
```
% "Building blocks" for Laws 3x3 texture masks
L3 = [ 1  2  1];
E3 = [-1  0  1];
S3 = [-1  2 -1];

% "Building blocks" for Laws 5x5 texture masks are made from the X3 masks
L5 = conv2(L3, L3, 'full')
E5 = conv2(L3, E3, 'full');
S5 = conv2(L3, S3, 'full');
W5 = conv2(E3, S3, 'full'); % not mentioned in the lecture notes
R5 = conv2(S3, S3, 'full');

% We are desiding to use this 5x5 mask
E5E5 = conv2(E5', E5, 'full');

% Filter the image with the 5x5 mask
laws_energy = imfilter(double(img), E5E5, 'symmetric', 'conv');

figure(10);clf
imshow(laws_energy, []);
title('Result after using a 5x5 E5E5 mask, laws energy');

% Using a mean filter on laws energy
laws_mean_abs_feat = imfilter(abs(laws_energy), ones(35), 'symmetric');

figure(11);clf
subplot(211)
imshow(laws_mean_abs_feat, []);
title('Mean of "Laws energy"');
subplot(212),
imshow(img.*uint8(laws_mean_abs_feat > (max(laws_mean_abs_feat(:)) * 0.3)),[]);
title('Tresholded mean of Laws energy');

% Finding the standard deviation of Laws energy
laws_std_feat  = stdfilt(laws_energy, ones(35));
figure(12);clf
subplot(211)
imshow(laws_std_feat,[]);
title('Standard deviation of Laws Energy');
subplot(212)
imshow(img.*uint8(laws_std_feat > (max(laws_std_feat(:))* 0.3)),[])
title('Tresholded with standard deviation of Laws Energy');
```

```
L5 =

     1     4     6     4     1
```

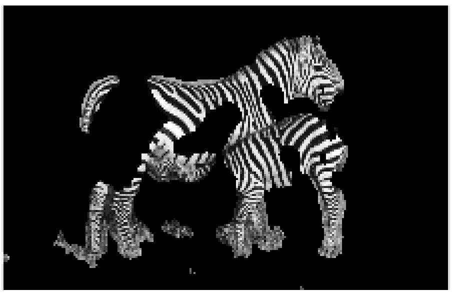
Result after using a 5x5 E5E5 mask, laws energy



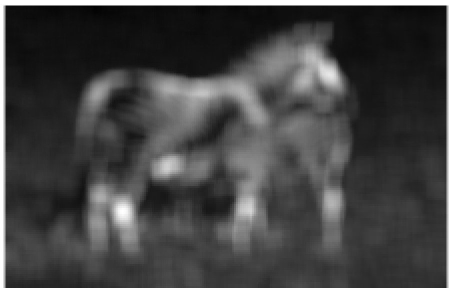
Mean of "Laws energy"

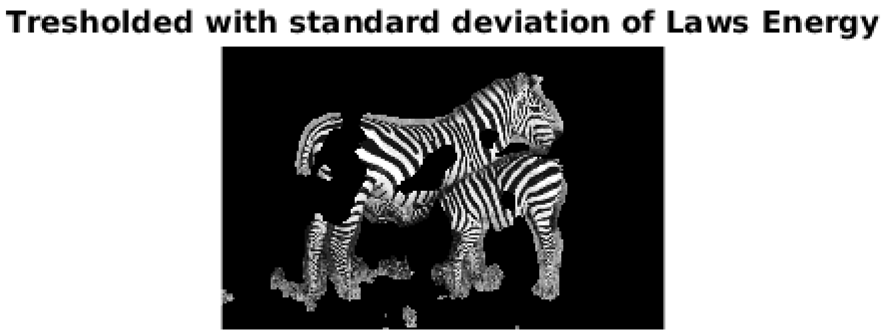


Tresholded mean of Laws energy



Standard deviation of Laws Energy





```
end
```

## Function for Task 1 : Gliding window GLCM

The function glidingGLCM takes as input a gray level image, the window size, the number of graylevels, the GLCM parameters dx and dy.

```
function[glcmVar,glcmCtr,glcmEnt] = glidingGLCM(img,G,dx,dy,windowSize)
%GlidingGLCM
% Calulates the GLCM and the feature in fun for
% every gliding window in img. It is first added
% a frame of ones around img to make sure that the
% resulting feature images is the same size as img.

[Mo,No] = size(img);            %Size of original image
halfSize = floor(windowSize/2); %Size of "half" the filter

% Expand the original image with a zero border
imgWithBorders = zeros(Mo+windowSize-1, No+windowSize-1);
imgWithBorders(halfSize:end-halfSize-1,halfSize:end-halfSize-1) = img;

figure(100)
imagesc(imgWithBorders);
colormap gray
title('The resuting image with borders')

% Size of the image with borders
[M,N] = size(imgWithBorders);

% Index matrices. These are needed for the online implementation of the
% GLCM features
i = repmat((0:(G-1))', 1, G);
j = repmat( 0:(G-1)  , G, 1);

% Defining buffers for the resulting glcm feature images
glcmVar = zeros(Mo,No);
glcmCtr = zeros(Mo,No);
glcmEnt = zeros(Mo,No);


% Go through the image
for m = 1+halfSize:M-halfSize-1
    for n = 1+halfSize:N-halfSize-1

        % Extracting the wanted window
        window = imgWithBorders(m-halfSize:m+halfSize,n-halfSize:n+halfSize);
        % Calculate the
        p = GLCM(window,G, dx,dy,0,0);

        % Compute GLCM's variance feature.
        mu = mean(window(:));
        glcmVar(m-halfSize,n-halfSize) = sum(sum(p .* ((i-mu).^2)));
        % Computed using for-loops:
        % for i = 0:(G-1)
        %     for j = 0:(G-1)
        %         glcm_var(m,n) = glcm_var(m,n) + p(i+1,j+1) * (i-mu)^2;
        %     end
        % end

        % Compute GLCM's contrast feature.
        glcmCtr(m-halfSize,n-halfSize) = sum(sum(p .* ((i - j).^2)));
        % Computed using for-loops:
        % for i = 0:(G-1)
        %     for j = 0:(G-1)
        %         glcm_ctr(m,n) = glcm_ctr(m,n) + p(i+1,j+1) * (i-j)^2;
        %     end
        % end
        % Compute GLCM's entropy feature (with base 2).
        glcmEnt(m-halfSize,n-halfSize) = -sum(p(p>0) .* log2(p(p>0)));
    end
end
end
```

## Function for Task 1 : The GLCM function

```
function [glcm] = GLCM(img,G,dx,dy,normalize,symmetric)
%GLCM calculates the GLCM (Gray Level Coocurrence
%Matrices) of an image
%   img         : the input image
%   G           : number of gray levels
%   dx          : distance in x-direction
%   dy          : distance in y-direction
%   normalize   : if 1 normalize by pixel pairs
%                 so sum(glcm(:)) = 1
%   symmetric   : if 1 make GLCM matrix symmetric
[N,M] = size(img);
glcm = zeros(G);

% For the image go through the entire image and count how many transitions
% from the first to the second graylevel
for i = 1:N
    for j = 1:M
        %Some idexing details, could proably be simlified 😉
        if i+dy > N || i+dy < 1 || j+dx > M || ...
                j+dy < 1 || i + dx < 1 || j + dx < 1
           continue
        end
        firstGLevel = img(i,j);
        secondGLevel = img(i+dy,j+dx);
        glcm(firstGLevel+1, secondGLevel+1) = ...
                glcm(firstGLevel+1, secondGLevel+1) + 1;
    end
end

% If we want a symmetric GLCM add the transpose. Read about the symmetric
% GLCM in the book or the lecture foils. It is basically counting both 2,1
% and 1,2 graylevel. Thus "counting both ways".
if symmetric
    glcm = glcm+glcm';
end
% If we want to normalize the GLCM
if normalize
    glcm = glcm/sum(sum(glcm));
end
end
% post_id = 286; %delete this line to force new post;
% permaLink = http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/;
```

```
Parameters: dx = 1, dy = 0, gives this result:
     2     2     1     0
     0     2     0     0
     0     0     3     1
     0     0     0     1

Parameters: dx = 0, dy = 1, gives this result:
     3     0     2     0
     0     2     2     0
     0     0     1     2
     0     0     0     0

Original image consists of these values:

ans =

    0    1    2    3    4    5    6    7    8    9   10   11   12

Image after histeq and normalization has these values:

ans =

    0    1    2    4    6    7
```

Published with MATLAB® R2015a (http://www.mathworks.com/products/matlab/)

## Comments

**Paul Magnus** says:

11/09/2015 at 12:03 (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/#comment-61)

You do this in your code:
p = GLCM(window,G, dx,dy,0,0);
mu = mean(window(:));
glcmVar(m-halfSize,n-halfSize) = sum(sum(p .* ((i-mu).^2)));
So you take the variance of p with respect to the original image?
From the lecture notes:
"This feature puts relatively high weihts on the elements that differ from the average value of P(i,j)"
So should it not be mu=mean(p)?

Reply (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/?replytocom=61#respond)

**omrindal** says:

12/09/2015 at 14:34 (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/#comment-63)

Hello. This is actually a really good question and I'm quite confused myself. Let me check this up with Fritz on Monday and I'll come back with an answer.

Reply (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/?replytocom=63#respond)

**omrindal** says:

14/09/2015 at 13:29 (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/#comment-72)

I have received an answer from Fritz, and I am sorry that I was mistaken in my code. The $\mu$ is supposed to be the mean value of $P$. See heim.ifi.uio.no/~in384/info/glcm.ps

I'll correct my code, and another bug I have been made aware of.

Cheers

Reply (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/?replytocom=72#respond)

**Cameron Lowell Palmer** says:

15/09/2015 at 19:53 (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/#comment-73)

Ole Marius and I worked on this some more today and it seems mu comes from 832-833 in the text. It is never referred to as mu, but if you unfold the variance formula you'll find it. mu = sum_i( i * sum_j( p_ij ) ). Which in terms of MatLab code I believe is:

```
G = glcm(slidingWindow, ...);
p = G ./ sum(G(:));
mu = sum(sum(p .* (i + 1)));
variance(row, col) = sum(sum(p .* ((i - mu).^2)));
```

Reply (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/?replytocom=73#respond)

**omrindal** says:

16/09/2015 at 10:06 (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/#comment-74)

I totally agree with this, so disregard my earlier message!

Reply (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/?replytocom=74#respond)

**NI HUNGCHIH** says:

11/09/2015 at 22:42 (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/#comment-62)

Hi:
Can you explain how these two lines work again ?

i = repmat((0:(G-1))', 1, G);
glcmVar(m-halfSize,n-halfSize) = sum(sum(p .* ((i-mu).^2)));

thanks~~

Reply (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/?replytocom=62#respond)

---

**omrindal** says:

12/09/2015 at 14:41 (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/#comment-64)

Hello,

The definition on the GLCM variance feature is $VAR = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} (i-\mu)^2 P(i,j)$. Buy the way, I'll check up what mean $\mu$ actually is, see the previous question. So we want to take the sum of the entire GLCM matrix, $P$, where we are weighting each matrix element by $(i-\mu)^2$. The two code lines `i = repmat((0:(G-1))’, 1, G);` `glcmVar(m-halfSize,n-halfSize) = sum(sum(p .* ((i-mu).^2)));` takes the entire matrix P and find the dot product with a "weight" matrix $(i-u)^2$. Then we have done all the weighting in the dot multiplication and can simply sum the result. Hope this makes sense.

Reply (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/?replytocom=64#respond)

---

**Cameron Lowell Palmer** says:

13/09/2015 at 13:28 (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/#comment-65)

"Variance calculated using i or j give the same result, since the GLCM is symmetrical."

I take that to mean:

variance_i(row, col) = sum(sum(glcm .* ((i - mu).^2)));
variance_j(row, col) = sum(sum(glcm .* ((j - mu).^2)));

should be equivalent. Is that correct?

Reply (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/?replytocom=65#respond)

---

**omrindal** says:

13/09/2015 at 19:47 (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/#comment-68)

Yes, if the GLCM is symmetrical. But that is not be the case in my code, but you can easily change it by changing the argument you pointed out earlier. Did I write that it is symmetrical? Or is that from the lecture foils?

Reply (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/?replytocom=68#respond)

---

**Cameron Lowell Palmer** says:

13/09/2015 at 20:57 (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/#comment-69)

From the lecture foils. Slide 43, third bullet point.

Reply (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/?replytocom=69#respond)

---

**Cameron Lowell Palmer** says:

13/09/2015 at 13:34 (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/#comment-66)

If the GLCM is symmetrical, shouldn't your call to GLCM have a 1 in place of the symmetrical option?

p = GLCM(window,G, dx,dy,0,1);

Reply (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/?replytocom=66#respond)

---

**omrindal** says:

13/09/2015 at 19:43 (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/#comment-67)

Yes, if I wanted it to be symmetrical. But I just wanted the simplest non-symmetrical 🙂

Reply (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/?replytocom=67#respond)

---

**Cameron Lowell Palmer** says:

13/09/2015 at 21:04 (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/#comment-70)

Isn't Pij supposed to be equal to Gij / n where Gij is the current GLCM position and n is the sum of values in the GLCM? I'm assuming mu was defined somewhere as the average of the current window, but I can't find it.

```
I = [0 0 1 1; 0 0 1 1; 0 2 2 2; 2 2 3 3];


gray_levels = 4;
gray_limits = [min(I(:)), max(I(:))];

var = zeros(gray_levels);


 window_size = 3;
[rows, cols] = size(I);
half_size = floor(window_size / 2);
ZI = ZeroPadImage(I, half_size);
for row = 1:rows
for col = 1:cols
row2 = row + (half_size * 2);
col2 = col + (half_size * 2);
sliding_window = ZI(row:row2, col:col2);
G = graycomatrix(sliding_window, 'NumLevels', gray_levels, 'GrayLimits', gray_limits,
'Symmetric', true);
mu = mean(sliding_window(:));
for i = 1:gray_levels
for j = 1:gray_levels
Pij = G(i, j) / sum(G(:));
var(i, j) = (i - mu)^2 * Pij;
end
end
end
end
```

Reply (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/?replytocom=70#respond)

**Cameron Lowell Palmer** says:

13/09/2015 at 21:05 (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/#comment-71)

Sorry, I meant to say this is the most 'naive' version I could come up with to try and nail down my understanding of the mechanics.

Reply (http://inf4300.olemarius.net/2015/09/09/glcmexercises-m/?replytocom=71#respond)

## Leave a Reply

Your email address will not be published. Required fields are marked *

**Name ***

**Email ***

**Website**

**Comment**

Post Comment

« Previous Post (http://inf4300.olemarius.net/2015/09/02/solution-m/)

Next Post » (http://inf4300.olemarius.net/2015/09/17/omhr_solution-m/)

This page is not to be confused with the official course web site (http://www.uio.no/studier/emner/matnat/ifi/INF4300/h15/index.html). You need to pay attention to the course web page since all official information will be posted there.

At this website I will post the solutions to the weekly exercises (http://www.uio.no/studier/emner/matnat/ifi/INF4300/h15/undervisningsplan/), and you have the possibility to leave a comment or two on these solutions. You are encouraged to do so. I prefer open questions on this site, but you may also contact me on my email omrindal@ifi.uio.no or you may stop by my office in room 4466.

All the code presented on this site will also be available as a git repository (https://github.com/olemarius90/INF4300). You are all ecouraged to learn to use Git. A nice introduction to Git can be found here (https://try.github.io/).

## Posts

Solution to exam 2013 (http://inf4300.olemarius.net/2015/11/30/solution-to-exam-2013/)

## Recent Comments

matlabi (http://www.matlabi.ir) on Week 1 : Introduction to MATLAB (http://inf4300.olemarius.net/2015/09/02/week-1-introduction-to-matlab-this-weeks-assignment-is-a-quick-introduction-to-matlab-for-image-analysis-the-original-file-is-found-under-the-undervisningsplan-on-the-course-web-site-the-s/#comment-94)

omrindal on Solution to exercises week 8, The Multivariate Gaussian Classifier (http://inf4300.olemarius.net/2015/10/27/exercise-m-2/#comment-93)

Ni, Hung Chih on Solution to exercises week 8, The Multivariate Gaussian Classifier (http://inf4300.olemarius.net/2015/10/27/exercise-m-2/#comment-92)

omrindal on Solution to exercises week 8, The Multivariate Gaussian Classifier (http://inf4300.olemarius.net/2015/10/27/exercise-m-2/#comment-91)

Ni Hubg Chih on Solution to exercises week 8, The Multivariate Gaussian Classifier (http://inf4300.olemarius.net/2015/10/27/exercise-m-2/#comment-90)

## Search

Search

Search