Oblig2 INF4300

mathiaki

13. november 2017

Innhold

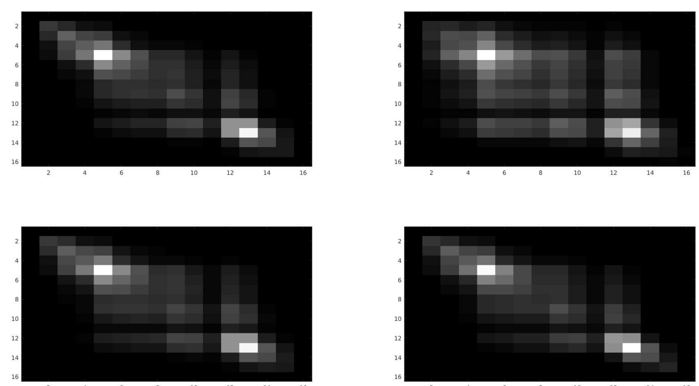
1 Texture description

The first thing to do in this mandatory assignment was to find the best GLCM for all 3 of the textures. In the last assignment our job was to find these matrices, but this time we already have the finished glcm matrices.

With the finished GLCM matices, we can now get the feature images for the different orientations.

1.1 Matrix data

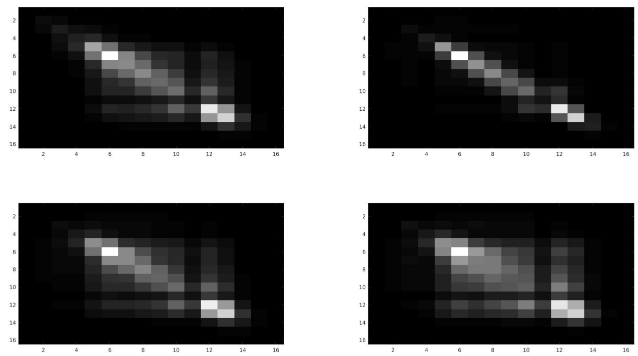
from the files included we get this result for the 4 different textures:



(a) Texture 1

Figur 1: Texture 1 GLCM

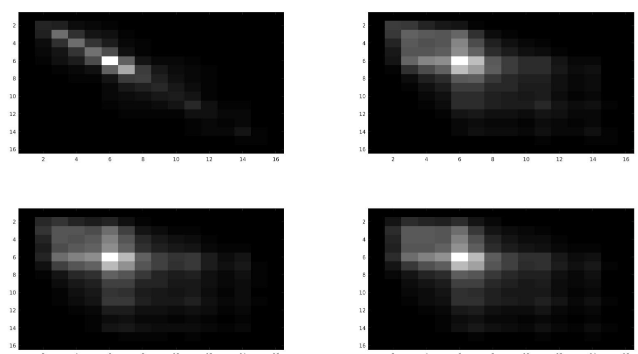
From this first texture we choose the first of the 4 images.
This corresponds to dx=0 dy=1



(a) Texture 2

Figur 2: Texture 2 GLCM

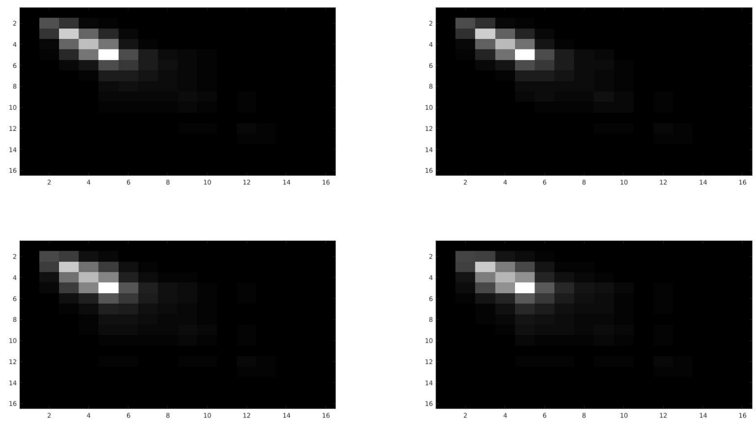
From this first texture we choose the first of the 4 images.
This corresponds to dx=1 dy=0



(a) Texture 3

Figur 3: Texture 3 GLCM

From this first texture we choose the first of the 4 images.
This corresponds to dx=0 dy=-1



(a) Texture 4

Figur 4: Texture 4 GLCM

From this first texture we choose the first of the 4 images.
This corresponds to dx=0 dy=-1
Now that we have all the necessary dxy values, we can now start with the sliding GLCM part of the program.

2 Quadrant and sliding

With the GLCM matrices from assignment 1 we can now divide the each of the GLCM matrices in to 4 parts:

```
GLCM Gliding window
|---------------|
|Q11|Q12|       |
|---|---|  Q2   |
|Q13|Q14|       |
|-------|-------|
|       |       |
|  Q3   |  Q4   |
|       |       |
|---------------|
```

Figur 5: GLCM Gliding matrix

When we now run the gliding GLCM the result of the different quadrants are stored in the respective variables. As shown in 5.

As the Q1 quadrant has the most difference between the different textures, it is natural to spit the quadrant up in to 4 subquadrants.

```
function [Q1,Q2,Q3,Q4,Q11,Q12,Q13,Q14] = gGLCM(img, G, dx, dy, window)
% at this pont in the process this function was not self-made.
% Heavy inspiration for Kristoffer Hoiseter, since he did the first
% obligatory assignment in MATLAB, and I made my gliding window in python.


[M,N] = size(img);
halfW = floor(window/2);

%expanding the image with a border
imgBorder = zeros(M+window-1, N+window-1);
imgBorder(halfW+1:end-halfW,halfW+1:end-halfW) = img;

%size of the new image
[Mborder, Nborder] = size(imgBorder);

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


Q1 = zeros(M,N);
Q2 = zeros(M,N);
Q3 = zeros(M,N);
Q4 = zeros(M,N);

%spitting the top left quadrant in 4, because the action is happening here
```

```
Q11 = zeros(M,N);
Q12 = zeros(M,N);
Q13 = zeros(M,N);
Q14 = zeros(M,N);


%going through the image
for m = 1+halfW:Mborder-halfW-1
    for n = 1+halfW:Nborder-halfW-1

        win = imgBorder(m-halfW:m+halfW, n-halfW:n+halfW);

        p = GLCM(win,G,dx,dy);

        Q1(m-halfW,n-halfW)=sum(sum(p(1:G/2,1:G/2)))/sum(sum(p));
        Q2(m-halfW,n-halfW)=sum(sum(p(1:G/2,G/2:G)))/sum(sum(p));
        Q3(m-halfW,n-halfW)=sum(sum(p(G/2:G,1:G/2)))/sum(sum(p));
        Q4(m-halfW,n-halfW)=sum(sum(p(G/2:G,G/2:G)))/sum(sum(p));

        Q11(m-halfW,n-halfW)=sum(sum(p(1:G/4,1:G/4)))/sum(sum(p));
        Q12(m-halfW,n-halfW)=sum(sum(p(1:G/4,1+G/4:G/2)))/sum(sum(p));
        Q13(m-halfW,n-halfW)=sum(sum(p(1+G/4:G/2,1:G/4)))/sum(sum(p));
        Q14(m-halfW,n-halfW)=sum(sum(p(1+G/4:G/2,1+G/4:G/2)))/sum(sum(p));


    end


end
end
```