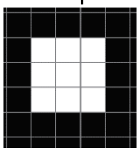
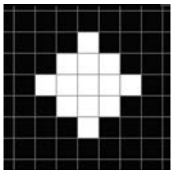
Video Signals

Date: 3 September 2019

Ex.1.[11 pts] Consider the two following images with a 3x3 square, assume that all the background pixels are black and has a value of '0' while the white pixels of the squares have a value of '1'.

We want to extract edges of both shapes. In order to do this we want to use Sobel filters:

- **[2 pts]** Define the Sobel filters for horizontal and vertical edge extraction.
- **[3 pts]** Apply these filters to each image and provide the numerical results.
- **[6 pts]** Gather the results from the previous point and define a suitable threshold value to mark edges.

Es.2. [11 pt]

Applying a JPEG encoding to an image, after the DCT transform of 4 different 8x8 blocks of the image we get the following results:

$$DCT\left(Block_1\right)=\begin{bmatrix}1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{bmatrix}\quad DCT\left(Block_2\right)=\begin{bmatrix}0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{bmatrix}$$

$$DCT\left(Block_3\right)=\begin{bmatrix}0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{bmatrix}\quad DCT\left(Block_4\right)=\begin{bmatrix}0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1\end{bmatrix}$$

- **[2pts]** What transform shall be used (provide the formula) in order to reconstruct the 8x8 Blocks?
- **[9pts]** Describe and provide a qualitative intensity representation of the Blocks.

The exam paper continues overleaf

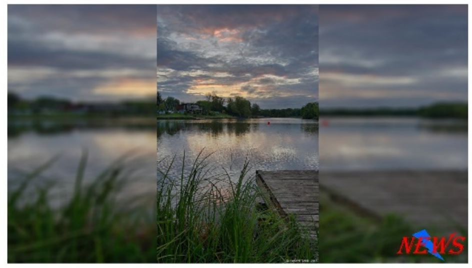
Es.3. [11 pt MATLAB Exercise]

You are working for a tv news program and you want to broadcast some footage recorded by a smartphone that unfortunately was shot in portrait mode instead of landscape (i.e. it was shot vertically). Write a MATLAB script able to take as an input a video frame (stored in a file called '*image.jpg*') and generate a 720p (1280x720) version with blurred background superimposing also your logo (stored in *'logo.png'*) in the bottom right corner.

Matlab List of possible functions

`figure`
`rgb2ind`
`im2double`
`imread`
`imclose`
`zeros`
`rgb2gray`
`imcrop`
`ones`
`imopen`
`imshow`
`find`
`fspecial`
`min`
`max`
`strel`
`imnoise`
`imfilter`
`round`
`sum`
`size`
`imresize`
`norm`

logo.png



output

a) Read the input 8-bit color image, convert it into an image of class double. Save its vertical and horizontal sizes in the variables *h* and *w* respectively.

b) In order to create the landscape blurred version follow these steps:
   I. Obtain *I_middle* by resizing the original image so the output height would be 720 (the width must scale accordingly).
   II. Initialize *I_out* as a stretched version of *I_middle* having a width of 1280.
   III. Substitute each channel of *I_out* with a blurred version of them obtained applying a gaussian filter with 20 as size and 10 as standard deviation.
   IV. Substitute the central part of *I_out* with *I_middle.*

c) Add the logo in the bottom right part of the image with the following steps:
   I. Read the logo 8-bit color image, convert it into an image of class double.
   II. Resize it obtaining a 100x200 image choosing an algorithm that do not blur the edges between the logo and the green background.
   III. Obtain a binary image that has true values where the logo is not pure green.
   IV. Superimpose the resized logo in the bottom right part of the image (do not copy the green background)

Solutions

Ex.1

The Sobel vertical edge extractor filter is: $\mathbf{G}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ and the horizontal one is:

$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$; convolving the vertical filter with the two images, assuming an infinite background of black pixels we get for the first image I_{1x}:

0	0	0	0	0	0	0
0	-1	-1	0	1	1	0
0	-3	-3	0	3	3	0
0	-4	-4	0	4	4	0
0	-3	-3	0	3	3	0
0	-1	-1	0	1	1	0
0	0	0	0	0	0	0

Filtering the second image with the vertical filter, we get I_{2x}:

0	0	0	0	0	0	0	0	0
0	0	0	-1	0	1	0	0	0
0	0	-1	-3	0	3	1	0	0
0	-1	-3	-3	0	3	3	1	0
0	-2	-4	-2	0	2	4	2	0
0	-1	-3	-3	0	3	3	1	0
0	0	-1	-3	0	3	1	0	0
0	0	0	-1	0	1	0	0	0
0	0	0	0	0	0	0	0	0

The convolutions with the horizontal filter will simply be the transpose of these results.
Combining the two filters, $I^2{}_{filtered} = I_x{}^2 + I_y{}^2$

And we get: $I_1{}^2{}_{filtered} =$

0	0	0	0	0	0	0
0	2	10	16	10	2	0
0	10	18	16	18	10	0
0	16	16	0	16	16	0
0	10	18	16	18	10	0
0	2	10	16	10	2	0
0	0	0	0	0	0	0

$I_{2\ filtered}^{2} =$

0	0	2	4	2	0	0	0
0	2	18	16	18	2	0	0
2	18	18	4	18	18	2	0
4	16	4	0	4	16	4	0
2	18	18	4	18	18	2	0
0	2	18	16	18	2	0	0
0	0	2	4	2	0	0	0
0	0	0	0	0	0	0	0

A possible threshold on the squared filtered images could be 16.

Ex.2

In order to recover the original blocks we have to use the Inverse Discrete Cosine Transform, i.e.

$$A_{mn} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \alpha_p \alpha_q B_{pq} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}, \quad \begin{matrix} 0 \le m \le M-1 \\ 0 \le n \le N-1 \end{matrix},$$

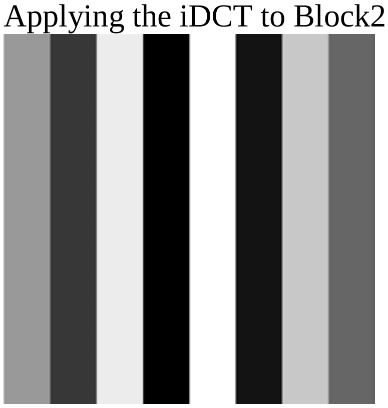
where

$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}}, & p = 0 \\ \sqrt{\frac{2}{M}}, & 1 \le p \le M-1 \end{cases}$$

and

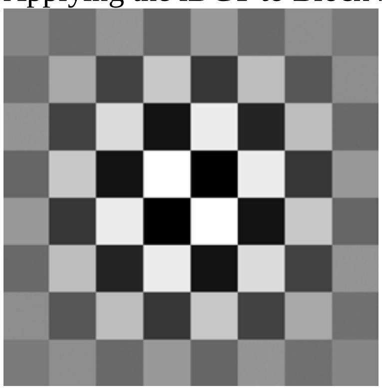
$$\alpha_q = \begin{cases} \frac{1}{\sqrt{N}}, & q = 0 \\ \sqrt{\frac{2}{N}}, & 1 \le q \le N-1 \end{cases}.$$

Applying the iDCT to Block1 we get a constant value (uniform 8x8 region).
Applying the iDCT to Block2 we get an image with the highest horizontal frequencies:



Applying the iDCT to Block3 we an image with a low frequency vertical sinusoid:

Applying the iDCT to Block4 we get an image with the highest horizontal and vertical frequencies:



Ex.3

```
close all; clear all;

%a)
I = imread('image.jpg');
I = im2double(I);
w = size(I,2); h = size(I,1);
t_h = 720; t_w = 1280;

%b1)
I_middle = imresize(I,t_h/h);
%b2)
I_out = imresize(I,[t_h,t_w]);
s_w = (t_w-size(I_middle,2))/2;
for i=1:3
    H = fspecial('gaussian',20,10);
    I_out(:,:,i) = imfilter(I_out(:,:,i),H,'symmetric');
end

%b3)
s_w = (t_w-size(I_middle,2))/2;
I_out(:,(s_w+1):(t_w - s_w),:) = I_middle;

%c1)
logo = imread('logo.png');
logo = im2double(logo);

%c2)
logo = imresize(logo,[100 200],'nearest');

%c3)
M = (logo(:,:,1) == 0 & logo(:,:,2) == 1 & logo(:,:,3) == 0);
M = ~M;

%c4)
for i = 1:size(M,1)
    for j = 1:size(M,2)
```

```
        if(M(i,j))
            I_out(620+i,1080+j,:) = logo(i,j,:);
        end
    end
end

figure(); imshow(I_out)
```