EENG 428 Laboratory --- Lab Session 4

Description:

In this session, the fundamentals of digital image processing and image preprocessing techniques that are commonly used in computer and machine vision applications are to be discussed.

Prerequisites:

Attending students are expected to know:

- Basic Matlab commands.

Contents:

1- What is a digital image (binary images, gray-scale images, color images).

2- 2D filtering operations (filter mask).

3- Histogram of an image, and histogram equalization.

4- Conversion between image types.

5- Noise reduction (Gaussian noise, impulsive noise)

6- Edge detection.

7- Morphological operations (dilation, erosion, opening and closing).

1- Digital images:

A digital image is a single or multi-dimensional matrix, which contains numerical values (pixels) corresponding to sampled information from a specific scene.

Basic image types:

1.1- Binary images:

A digital binary image (black and white image) is a set (or a matrix), with only two different pixel values, (1 is white and 0 is black).

```
Ibw=imread('B.png');
imshow(Ibw)
imshow(1-Ibw) %negative
```

Look at the image in the work space!

There exist a value of “0” for each black pixel and a value of “1” for each white pixel. (Logical)

Working with binary images is considered to be the fastest among all type of images, and this is due to the fact that logical operations (anding, oring … etc) are much faster than multiplying and summing.

The most famous operations that are usually done on binary images are known as “Morphological Operations”. (we will see dilation, erosion, opening and closing).

1.2- Gray-scale images:

A digital gray-scale image is a set (or a matrix), with pixel values in the range {0-255} (255 is white and 0 is black).
Each pixel is represented in 8-bits (2^8 = 256) !!

```
Igs=imread('G.jpg');
imshow(Igs)
imshow(255-Igs) %negative
```

You may see that each bright pixel has value around 255 and each dark pixel has value around 0.

These type of images are known as the luminance component of an image. They contain the basic structure of an image.

Working with gray-scale images is usually done using “filtering” operations (linear or nonlinear filtering) or point operations.
The linear filtering operation requires 2D convolution, non-linear operations usually combine order-statistics (such as finding the median), while point operations deals with each pixel value separately.

See: Filtering, Histogram Equalization

1.3- Color Images:

A usual color image consist of 3 channels (3 layers) of gray-scale images.
In a typical digital image (the images we take with normal cameras), the image consists of 3-channels, namely Red, Green and Blue channels.
These type of images are known as RGB images.

```
Icol=imread('RGB.jpg');
figure
imshow(Icol)
figure
imshow(Icol(:,:,1))% the R channel
figure
imshow(Icol(:,:,2))% the G channel
figure
imshow(Icol(:,:,3))% the B channel
```

-Each pixel has 3 components (Red, Green and Blue).
-The mixture of these components is generating the pixel color. (Recall the color mixing theory from school).
-If the pixel is white, all of the components are in the maximum level (255).
-If the pixel is black, all of the components are in the minimum level (0).

Dealing with color images in the RGB color space is not easy (or not efficient in most of the times). Because both of the luminance component and chrominance component are integrated in the RGB channels.

Usually, we tend to convert a color image to other color spaces (such as HIS (Hue, Saturation and Intensity) to process it, then we return back to RGB to save or display.

2- 2D filter operations:

2.1- Linear filtering:

Linear filtering refers for the 2D convolution operation (fold, shift, multiply and add).

According to the type of filter, different operation can be done. Such as:

- Smoothing (blurring)
- Sharpening (taking the edges).

- Smoothing operation (low pass filtering) is usually used to reduce noise (noise has high frequency). Or maybe to omit some details (removing rankles from the face).

  The following example filters an image with a 3*3 box filter.

  Box filter:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

```
H=[1 1 1; 1 1 1; 1 1 1]/9;
Ifilt=imfilter(Igs,H);
imshow(Ifilt)
```

See that the image was blurred!!

Try to increase the size of the filter (5*5), and see that the image will be more blurred.

The following example filters an image with a 3*3 Sobel filter:

Sobel filter:

-1	0	1
-1	0	1
-1	0	1

```
H=[-1 0 1; -1 0 1; -1 0 1];
Ifilt=imfilter(Igs,H);
imshow(Ifilt)
imshow(255-Ifilt)
```

See that only the edges were preserved!!

This operations is high-pass filtering, and keeps only the pixels that separates between two different (components) of the image, which are edges.

Try having another Sobel filter as:

-1	-1	-1
0	0	0
1	1	1

And see the results.

(Do you have any idea about the difference in the outputs?!)

2.2- Non-Linear filtering:

For some applications (such as impulsive noise removal), we may need to use non-linear filters to process the image.

The most common non-linear filters are “order-statistics filters”, which usually work as follows:

-For each pixel, take a neighborhood (say 3*3)
-Order the neighborhood values (sort)
-Do the operation (take the median, variance …)

The following example filters an image with a 3*3 Median filter:

```
Ifilt=medfilt2(Igs);
imshow(Ifilt)
```

See that the image has changed, but no new pixel values were added to the image (not getting darker or brighter).

Order statistics filter exhibit massive superiority in removing impulsive noise compared with linear filters!! (see noise reduction).

3- Histogram of an image, and histogram equalization:

The histogram of an image is defined as the pmf (probability mass function) of the image values. Which is simply the number of times each value (from 0-255) occurred in the image.

Example:

```
imhist(Igs)
```

You may see that most of the pixels are bright.
Humans can see the details more when we have more difference between the colors.

This can be achieved using "Histogram Equalization", which is defined as the process of multiplying each pixel value by its cdf (cumulative distribution function).

Example:

```
Ieq= histeq(Igs);
imshow(Ieq)
```

Better?! No.   Reason?!

Another example for a more natural image:

```
I=imread('image.jpg');
imshow(I)
Ieq=histeq(I);
imshow(Ieq)
```

4- Conversion between image types:

4.1- Color image to gray scale image:

Simply, for each pixel, take the average between the 3 components to obtain a gray scale image.

Gs = (R+G+B)/3

(This usually fails when the colors are artificial).

Example:

```
Icolor=imread('dt.jpg');
Igrayscale=(Icolor(:,:,1)+Icolor(:,:,2)+Icolor(:,:,3
))/3;
Igrayscale=uint8(Igrayscale);
imshow(Igrayscale)
```

No good yet?!

There are other method to change color images to gray scale images, which combine more complicated (such as sparse representation and high dimensional projections), Matlab offers such a conversion:

Example:

```
Icolor=imread('dt.jpg');
Igrayscale=rgb2gray(Icolor);
imshow(Igrayscale)
```

Better?!

4.2- Gray scale to binary conversion:

This operation is very simple, simply by thresholding.

If the pixel value is below S (say S is 127), make it zero. Otherwise 1.

Example:

```
Ibinary=im2bw(Igrayscale);
imshow(Ibinary)
```

The only option is to modify the value of S.

Type “help im2bw” for details.

5- Noise reduction:

There are many type of noise that might degrade an image, each type of noise has its own statistical distribution which usually allows us to reduce (noise removal is impossible practically).
Depending on the type of noise, we may apply a suitable filter to reduce its effect on the image.

5.1- Gaussian Random Noise:

This type of noise is very common, and it has random values between -255 to 255, with Gaussian distribution.

Reducing the effect of this type of noise can be simply be done by averaging (low pass filtering).

Example:

```
IGauss=imnoise(Igs,'gaussian');
imshow(IGauss)
```

Averaging with a box filter:

```
H=ones(3,3)/9;
Ifilt=imfilter(IGauss,H);
imshow(Ifilt)
```

Better!?

See other types of filters:

```
H=[1 2 1; 2 4 2; 1 2 1]/16;
Ifilt=imfilter(IGauss,H);
imshow(Ifilt)
```

5.2- Impulsive Noise Reduction:

Common impulsive noise have values (-inf and inf), which usually correspond to (0 and 255) because of the 8bit representation of each pixel.

Reducing the effect of this noise is usually done by order statistics filters (medial filter for instance).

Example:

```
I_SaltPepper = imnoise(Igs,'salt & pepper');
imshow(I_SaltPepper)
```

Applying median filter:

```
Ifilt=medfilt2(I_SaltPepper);
imshow(Ifilt)
```

Try to filter again with median filter and see the result.

Try to filter with box filter!! See the difference?

6- Edge detection:

This operation is high pass filtering (remember Sobel operators), and is not useful to remove the noise (noise is already high frequency).

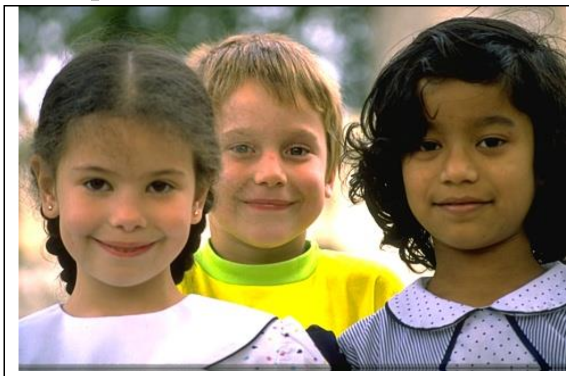
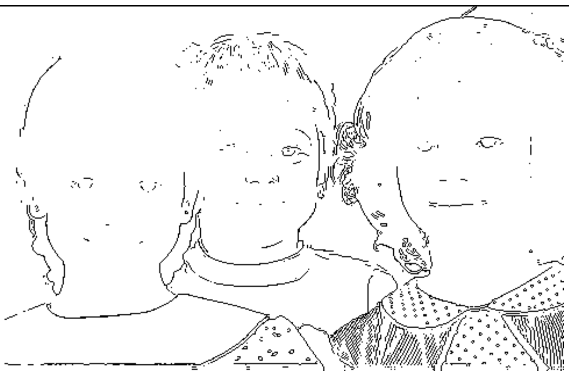
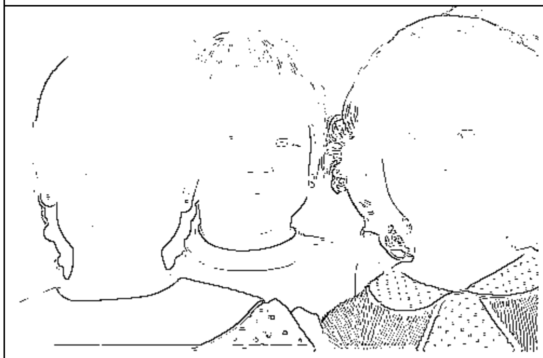
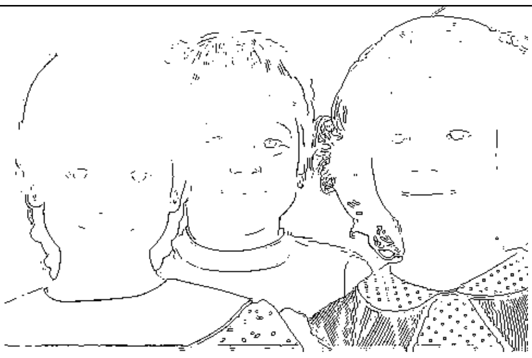
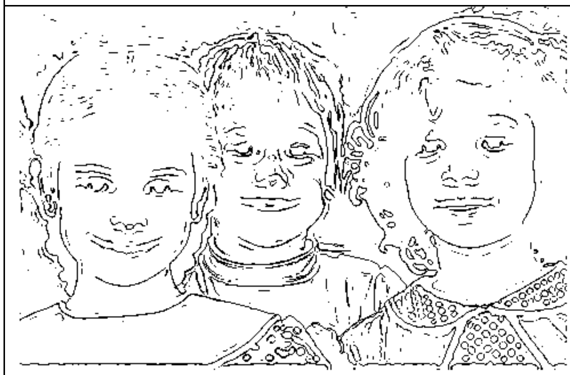
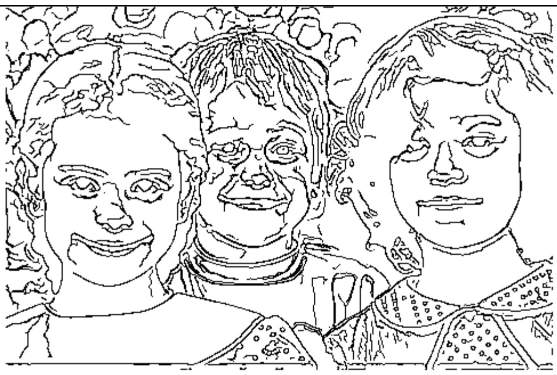
There are many edge detection filters such as:

- Sobel
- Roberts
- Prewitt
- Laplacian
- Canny

The best is Canny (there are many other methods depends on learning trees but they are very advanced for this introduction).

“The Canny method finds edges by looking for local maxima of the gradient of I. The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be "fooled" by noise, and more likely to detect true weak edges.” Matlab.

Example:

	 Sobel
 Roberts	 Prewitt
 Laplacian	 Canny

Example:

```
Ie=edge(Igs,'canny');
imshow(Ie)
imshow(1-Ie)
```

7- Morphological Operations:

Morphological operations are defined for binary images, and they are a simpler version from the filter operations in gray-scale images.

The filter here is called “structuring element”, and the filtering operations is defined in terms of set theory (intersection or union).

There are two basic operations, namely : erosion and dilation. Where all the other morphological operations are combinations of sequences of these two basics.

Regardless of the mathematics behind the operations, we may define these operations as follows:

- Erosion is the operation of removing one layer from edges from a 2D binary image.
- Dilation is the operation of adding one layer to the edges for a 2D binary image.

The process starts with defining the filter (the structuring element), the most common structuring element is the “cross”.

SE:

0	1	0
1	1	1
0	1	0

Example:

```
se=[0 1 0;1 1 1;0 1 0];
imshow(1-Ie)
Idil=imdilate(Ie,se); %dilation
imshow(1-Idil)
```

Example:

```
I=imread('B.png');
se=[0 1 0;1 1 1;0 1 0];
imshow(I)
Iero=imerode(I,se); %erosion
imshow(Iero)
```

Opening and closing:

Opening is the process of erosion then dilation. (Separates the components with narrow edges in between).

Closing is the process of dilation then erosion. (Combine near components)

These two operations are very useful in TEXT PROCESSING.
You may generate your own examples.

For more details on Morphological operations see:
http://homepages.inf.ed.ac.uk/rbf/HIPR2/open.htm

Homework:

Using your own clear face image:

1- Import the image to Matlab workspace using the command imread.
2- Convert the image to gray-scale and to binary and save both images in your computer
   (hint: use the command imwite to save from the workspace to the work directory
   Example:
   ```
   I=imread('RGB.jpg');
   Igs=rgb2gray(I);
   imwrite(Igs,'grayscale.png')
   ```
)
3- Detect the edges of your image using Sobel and Canny methods and save each image to your computer, comment which one is better visually.
4- Add Gaussian noise to your image and filter it with a median filter once and with an average filter once, save all images and comment on the results.

5- Add impulsive noise to your image and filter it with a median filter once and with an average filter once, save all images and comment on the results.
6- Find the histogram of your image, and perform histogram equalization, save the histogram equalized image to your computer and comment on the result.
7- From the binary image of yours, perform dilation, erosion, opening and closing. Save all images to your computer.

Save all your codes, all you comments in a word file, and all your images with a single rar or zip file, and send it to the email (lab.eeng428@gmail.com)

You have exactly 6 days to submit, late submissions will not be considered. Cheating in any aspect will be reported to the dean office.

Bonus:

1- (25 points extra added to your total), in part 4 and 5, use PSNR comparisons between the original image and noisy image, the original image and the denoised image to show that your denoising is efficient.
2- (25 points extra added to your total), in part 4, perform histogram equalization for the noisy image before performing average filtering. Explain briefly if the noise removal is possible with averaging filter or not for the histogram equalized noisy image!.
3- (50 points added to your total), there exist some methods to track a moving object in a video. One of these methods is known by Kalman filter. Find a code on the Internet that implements this method, and apply it on a video of YOUR OWN RECORD. The video should contain the date of the recording (otherwise no grade)
4- (50 points added to your total), there is ready function to apply Kalman filter on a video in the Matlab robotic tool box (on the lab website). Use this function on a video of YOUR OWN RECORD. The video should contain the date of the recording (otherwise no grade)

The bonus homework should be attached in a separate rar of zip file, and should contain all the information in a smart and organized manner.