CS2035b Data Analysis and Visualization - Lab 11

General Lab Instructions to Help Labs Run Smoothly

- Read through the lab instructions **before** coming to the lab.
- Do any required pre-lab preparation.
- Bring a **printed** copy of the lab instructions to the lab.
- Note: you **must** both sign the signature sheet in the lab and submit your (mostly) completed lab via Owl by Saturday 10:55pm. The intention of labs is that you do the lab in the period assigned: lab content is timely and usually matched to the lectures or the assignments.

Overview and Preparation

This lab will be done using MatLab 2016b as installed in the Health Sciences 14 and 16 general computing labs. You must attend this lab in HSB14 or HSB16 in order to get assistance from the TA. Attendance will be taken. You can use your UWO login/password to login to these machines. Lab submission is to be done via Owl. Remember, labs are worth 10% of the total grade for this course (there are 11 labs in total, you must do 8 to receive full marks). You must both attend the lab and submit your completed lab to obtain your 1.25 mark!!!

Upon completion of this lab, you should have done the following in the MatLab environment:

- Created a script file, lab1_2017.m, containing the MatLab code to do the exercises below to generate the required edge maps as jpg files. Submit your lab11_2017.m via Owl (no need to submit your jpg files as your lab11i_2017.m file should generate these when run).

Exercise 1: Grayvalue Edge Detection

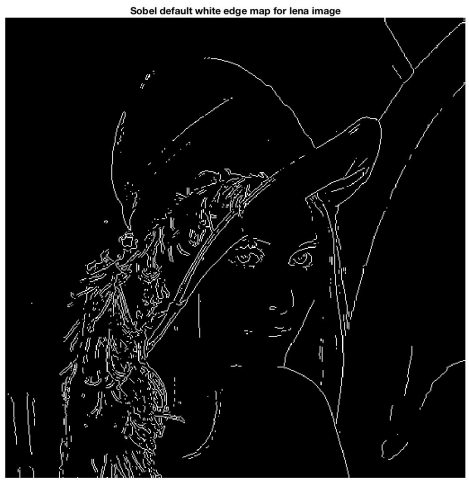
For this exercise we will use the **sobel** and **canny** edge detectors in MatLab (via **edge** in MatLab without any threshold arguments) to compute the edge maps of the **lena** and **mando** images, available on the course webpage as **lena.jpg** and **mando.jpg**. After we read these

images, we convert them to grayvalue images using **rgb2gray**, as edge detection can only be done on grayvalue images. The edge maps, `E`, produces by **edge** are binary images with 0 being black and 1 being white. We can flip the bits (1's becomes 0's and 0's become 1's) by putting the **not** operator in front of the edge map `~E`. A black edge map on a white background looks better that a white edge map on a black background and black backgrounds require a lot of toner cartridge to print! MatLab code to do this for the **sobel** edge detector can be found in **lab11.m** on the course webpage:

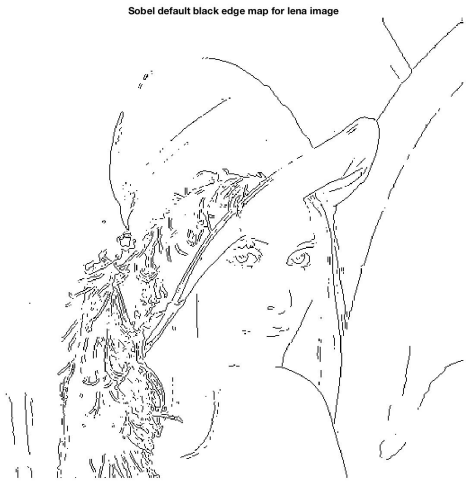
```
RGB1=imread('lena.jpg');
RGB2=imread('mando.jpg');
I1=rgb2gray(RGB1);
I2=rgb2gray(RGB2);
image_size=size(RGB1);
E1=edge(I1,'sobel');
figure
imshow(E1,[]);
title('Sobel default white edge map for the lena image');
print sobel_white_edge_lena.jpg -djpeg
figure
imshow(~E1,[]);
title('Sobel default black edge map for the lena image');
print sobel_black_edge_lena.jpg -djpeg
E1=edge(I1,'canny');
figure
imshow(E1,[]);
title('Canny default white edge map for the lena image');
print canny_white_edge_lena.jpg -djpeg
figure
imshow(~E1,[]);
title('Canny default black edge map for the lena  image');
print canny_black_edge_lena.jpg -djpeg
```

We can see from the edge maps in Figure 1 that the Canny edge maps are far superior to the Sobel edge maps. We also note that it is probably better to have a white background than a black background because you will use a lot less toner when printing the images!!!

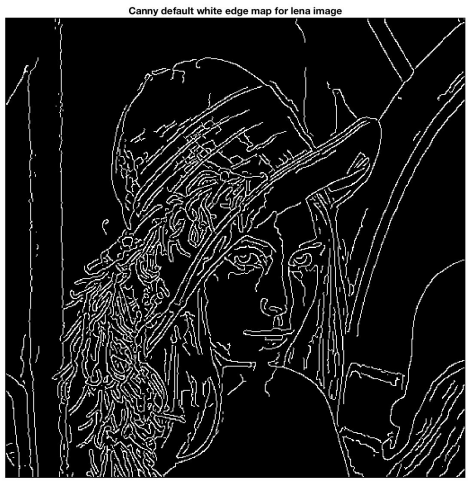
• This exercise requires you to compute the same images for the mando image. The MatLab code in lab11.m already has read in the mando image.



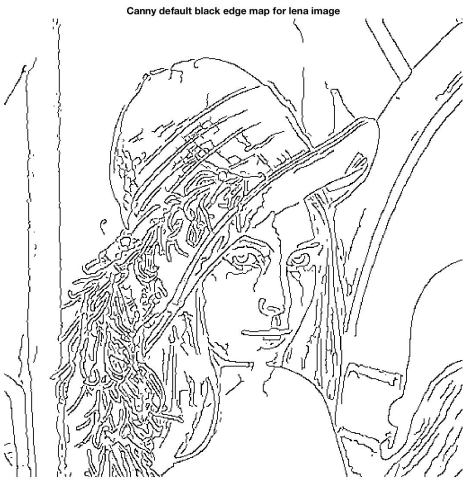
(a)



(b)



(c)



(d)

Figure 1: (a) and (b) white and black Sobel edge maps (on black and white backgrounds) for the lena image and (c) and (d) black and white Canny edge maps for the lena image.

Exercise 2: Colour Edge Detection

Although we can't directly compute edges on colour images we can compute the edges for each colour plane. That is we can compute the edges of the red, green and blue planes as edges on grayvalue images with the same pixel values. But in this case there is no guarantee that each colour pixel will be an edge in the 3 colour planes. We combine the 3 edge maps in the following way:

1. If pixel (i, j) is an edgel in all three colour planes we print white, i.e. (255,255,255), in the colour edge map.

2. If pixel (i, j) is a background pixel in all three colour planes we print black, i.e. (0,0,0), in the colour edge map.

3. If pixel (i, j) is both a edgel in some colour planes and a background pixel in other colour planes we print the colour component 255 for the edgels and 0 for the background pixels. Thus, is there is 1 edgel among the 3 colour planes for a pixel, than that colour (red, green or blue) will be printed. If there are 2 edgels among the 3 colour planes than the colour that results from combining the corresponding edgel colours will be printed. Note that red and green yield yellow, red and blue yield magenta and green and blue yield cyan. The primary colours for light are red, green and blue while the secondary colours for light are yellow, magenta and cyan. For pigments the primary and secondary colours are reversed (look inside a colour laser printer to see the toner cartridge colours). And, of source, the case for 3 edgels or 3 background pixels has been handled above.

We can also equivalently flip the bits in a colour image by changing white to black and black to white, while leaving actually coloured pixels alone. Some MatLab code to do this for the lena image is as follows:

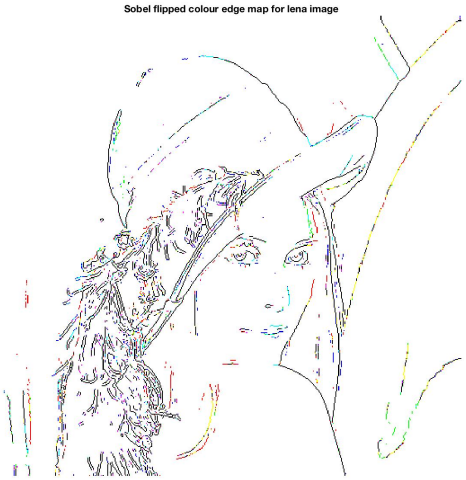
```
R1=squeeze(RGB1(:,:,1));
G1=squeeze(RGB1(:,:,2));
B1=squeeze(RGB1(:,:,3));

% Sobel Colour edges for lena
sobelER1=edge(R1,'sobel');
sobelEG1=edge(G1,'sobel');
sobelEB1=edge(B1,'sobel');
```

```
sobel_colour1=zeros(image_size,'double');
sobel_colour2=zeros(image_size,'double');
sobel_colour1(:,:,1)=sobelER1*255;
sobel_colour1(:,:,2)=sobelEG1*255;
sobel_colour1(:,:,3)=sobelEB1*255;
figure
imshow(sobel_colour1);
title('Sobel colour edge map for lena image');
print sobel_colour_edge_lena.jpg -djpeg

% might be a nicer way to do this
sobel_colour2=sobel_colour1;
for i=1:size(sobel_colour1,1)
for j=1:size(sobel_colour1,2)
if(sobel_colour1(i,j,1)==255 & sobel_colour1(i,j,2)==255 & sobel_colour1(i,j,3)==255)
   sobel_colour2(i,j,1)=0;
   sobel_colour2(i,j,2)=0;
   sobel_colour2(i,j,3)=0;
end % if
if(sobel_colour1(i,j,1)==0 & sobel_colour1(i,j,2)==0 & sobel_colour1(i,j,3)==0)
   sobel_colour2(i,j,1)=255;
   sobel_colour2(i,j,2)=255;
   sobel_colour2(i,j,3)=255;
end % if
end % for j
end % for i

figure
imshow(sobel_colour2);
title('Sobel flipped colour edge map for mando image');
print sobel_flipped_colour_edge_mando.jpg -djpeg
```

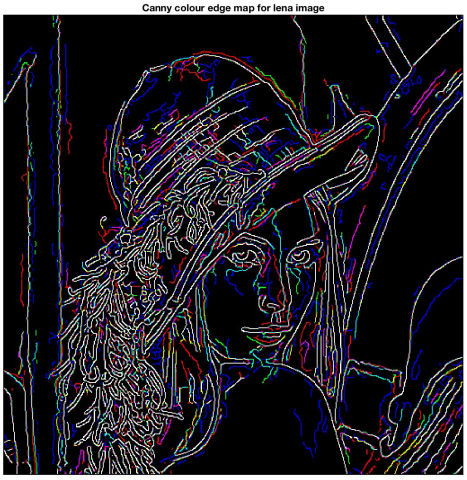
The lena images that results are shown in Figure 2 below:

• This exercise requires you to compute the same images for the mando image. Again the MatLab code in lab11_2017.m already has read in the mando image. Modify lab11_2017.m to produce the same images for the mando image.

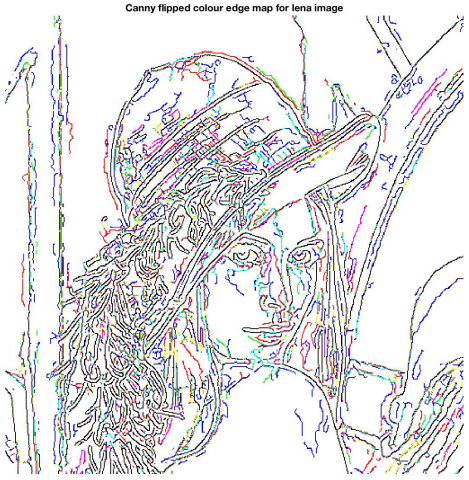
(a)



(b)



(a)



(d)

Figure 2: (a) The colour edge map for lena: a pixel is white when there is an edgel in all 3 colour planes, the pixel is black when there is no edgel in any of the 3 colour planes and either a primary colour (red, green or blue) if there is 1 edgel in the colour planes or a secondary colour (yellow, magenta or cyan) if there are 2 edgels in the colour planes and (b) the flipped colour edge image when the colour are unchanged but black and white are flipped. Both (a) and (b) are for the Sobel edge detector. (c) and (d) are the same as (a) and (b) except that the Canny edge detector is used instead of the Sobel edge detector.