Lab 1

Hanz Cuevas Velásquez, Bob Fisher
Advanced Vision
School of Informatics, University of Edinburgh

Week 2, 2018

This week's lab sheet is primarily aimed at students who have less backgraund in Matlab. This lab will focus on giving an introduction about how to read and write images and videos, how to use the webcam to acquire data and some basic operations.

Before anything else, we need to go inside the folder where we downloaded the lab material using the address bar of Matlab or using the terminal command `cd` in the Matlab command window.

1 Reading and writing images

We will start by reading an image in Matlab by using the command `imread('file')` where `file` is the path to our desired image. In our case, the image we want to read is in the Images folder, so, we input the following command in the command window:

```
Img = imread('Images/uni.png');
```

To observe how the `imread` command stored our image, we run the function `whos`.

```
whos Img
```

This function will return five values:

- Name
- Size
- Bytes
- Class
- Attributes

In this lab we will only focus on two of these values. The first one is the Size of the image. We can observe that the size of our image is 1024 x 1017 x 3 which indicates that the image has 1024 rows, 1017 columns, and 3 dimensions, chromaticities or channels: R, G, and B. Each of the values in the image represents a pixel and each pixel varies from 0 – 255. To access to each individual dimension we run the following code:

```
R = Img(:,:,1); % Red
G = Img(:,:,2); % Green
B = Img(:,:,3); % Blue
```

Now, we want to see the colour image and the 3 dimensions. To show multiple images in one figure or window, we have to use the command `subplot(#rows, #columns, current possition)`, `imshow(Image)` where `#rows`, and `#columns` are the number of images we want per row and column. In other words, Matlab divides the current figure into a `#rows`-by-`#columns` grid. The `current possition` indicates the position of the image in the grid. For this exercise we need four grids, so we will use a subplot of 2 rows and 2 columns. The command `imshow(Image)` displays the `Image` in a figure. The student should write the code to show the following figure, 1:

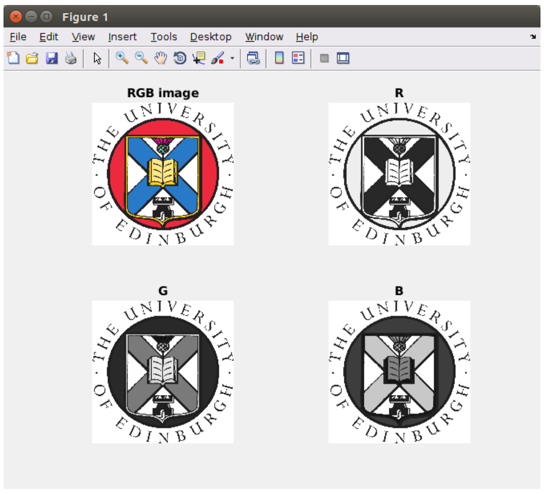

Figure 1: RGB, R, G, and B images shown in a figure.

If we go to the "Data Cursor" icon and click in any white part of the RGB image, we will see that it outputs the values of that pixel in the R, G, and B dimensions. E.g. `[R, G, B]:[255 255 255]`. On the other hand, if we click in any red part of the RGB image, we can observe that these values changes and now it will show the following: `[R, G, B]:[238 41 61]`. These values can also be seen in the R, G, and B images we have in the figure, however, in this case, this value will be labeled as index.

In the second part of this lab, we will investigate the second value shown after we use the command `whos`. For that, we run again `whos Img` and observe that its Class is uint8, which means that the image is represented by an unsigned 8 bit integer format. Sometimes, operations on images are easier when using floating point. For example, if we use the uint8 class, and we want to subtract one image from another, the result will be 0 everywhere the value of the second image is higher than the value of the first image. Also, most of the functions in the Image Processing Toolbox need to receive as input an image with a range of values beteen 0 and 1.

To transform an image to floating point we will use the function `im2double(Img)`. This function will transform an image `Img` into double precision rescaling the data from $0 - 1$. We will run the `im2double` command on the RGB image and also on the R, G, and B dimensions we created before:

```
Img = im2double(Img);
R = im2double(R);
G = im2double(G);
B = im2double(B);
whos Img
```

If we run the command `whos Img`, we will observe that the class has changed to double.

Another useful tool in computer vision is to transform an image to different colour spaces. Here, we will transform our RGB image to three of the most used colour spaces: *gray*, *HSV*, and *Lab*. To transform to these colour spaces, we need to run the following command, `rgb2colour_space(Img)` where `colour_space` should be replaced by the name of the colour space we want E.g. `rgb2hsv(Img)`. Please, write the code to transform our RGB image to *gray*, *HSV*, and *Lab*. You should show a figure similar to figure 2.

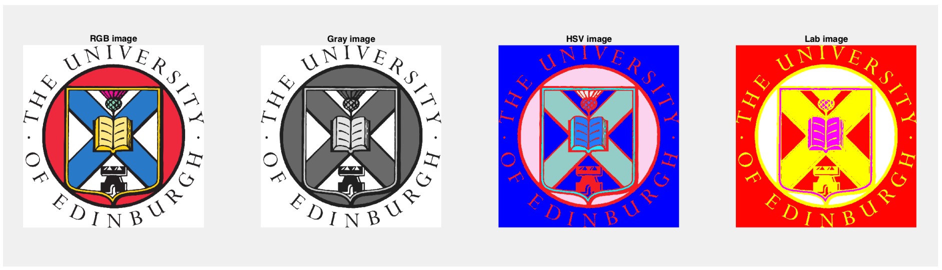

Figure 2: Different colour spaces.

Note: To show the titles on the top of the images, you have to use the command `title()`, E.g. `title('RGB image')`.

The final topic we will discuss, in this section, is how to save an image. An image can be saved in two ways, the first one using `imwrite(Img, 'file')` or after showing the image with `imshow`, clicking on the save icon. In this lab we will save the gray image in the `Images` folder using the first method and running the following code.

```
imwrite(Gray, 'Images/Gray_logo.png')
```

If we check our `Images` folder, we will find our gray image.

2 Reading and writing videos

As seen above, Matlab commands do all the difficult jobs for us, so we only have to input a little bit of code and focus on the task or project itself. The next tool we will use is how to read and write videos. First of all, we will run the command.

```
v = VideoReader('Images/plant.avi');
```

If you are using linux and Matlab shows you the following error: "*Error using VideoReader/init Could not read file due to an unexpected error. Reason: Unable to initialize the video obtain properties*", please refer to Appendix A to solve it.

We can show the content (frames) of the video by running the following command.

```
v = VideoReader('Images/plant.avi');
vidFrames=read(v);
nFrames=v.NumberOfFrames; % Get the number of frames of the video
for i=1:nFrames
   % showing the images in color (row,column,colour_space,frame)
   imshow(vidFrames(:,:,:,i));
end
```

What we are doing here is creating a multimedia reader object using `VideoReader`. Then, we extract the frames that the video has using `read()`, and also extract the number of frames the video has using `v.NumberOfFrames`.

To save a video we will use a similar structure as before. In this case we will use command `VideoWriter` and store our video using gray scale rather than colour. The video format we are going to use is AVI, however, the student can use the different variety of formats that Matlab offers[1]. Write the following code in the command window:

```
vw= VideoWriter('Images/example');
open(vw);
for i=1:nFrames
```

[1]https://uk.mathworks.com/help/matlab/ref/videowriter.html

```
    frame=rgb2gray(vidFrames(:,:,:,i)); % frames are grayscale
    writeVideo(vw, frame); % writeVideo(where we will save, what we will save);
end
close(vw);
```

You can see the new video in your folder `Images`.

In this exercise we used the frames extracted from the original `plant.avi` video, transform it to gray scale and saved them using the function `writeVideo(video, frame)` where `video` is the video writer object we created with `VideoWriter` and `frame` is the image we are storing into the video. In `VideoWriter('file', type)` we have to specify the path where we want our video to be saved and the type of file we want.

3 Webcam usage

In this last section of the lab, we will focus on learning how to use use the webcam in Matlab. To capture video from our webcam we will need the "MATLAB Support Package for USB webcams" toolbox. For that, we write the command `webcam` in the command window, it will give us the following: "MATLAB Support Package for Webcams has not been installed. Open Support Package Installer to install the Webcam Support Package.". We click on the link provided by the error and install the toolbox; you will need to create a Matlab account using your student mail to install it[2].

To find the webcams that are connected to our computer we will run the command `webcamlist`. This will give us a cell array with the names of the webcams we have. To access to one of the webcams, we run the command `webcam('webcam_name')` where you will replace `'webcam_name'` for the name of the first webcam in the `webcamlist`. Finally, we can view what our webcam is capturing by using the command `preview()`:

```
webcamlist;
cam = webcam('webcam_name');
preview(cam);

% clear the webcam object
clear('cam');
```

It is important to free the webcam using the command `clear('cam')`. The function `preview`, apart from showing us what the webcam is capturing, will show us some information about the webcam in the bottom part of the window: The time stamp, the resolution, and the frame rate. As a final exercise, we will store 10 images captured by the webcam in our Images folder. In other words, we will store the first 10 frames. To perform this task we will write the following code in the command window:

```
cam = webcam('webcam_name');
for i=1:10
    img = snapshot(cam); % Gets one frame of the video
    file_name = ['Images/Image' num2str(i) '.png'] % saves the image as Image(i).png
    imwrite(img, file_name)
    imshow(img)
end
clear('cam')
```

If we go to the Images folder, we will find our 10 images, `Image1.png` to `Image10.png`. Now, the student should use the code from Section 2 to record a video from the webcam and save it using the commands learned in this sheet.

[2]https://uk.mathworks.com/mwaccount/register
More info in https://www.ed.ac.uk/information-services/computing/desktop-personal/software/main-software-deals/matlab/getting-matlab

A Problem with VideoReader

If the error in Section 2 is seen, you can fix it, on a self-managed unix machine, in the following way[3]. First, open your unix terminal and execute the following code[4]:

```
sudo apt-get install gstreamer0.10-*
sudo add-apt-repository ppa:mc3man/gstffmpeg-keep
sudo apt-get update
sudo apt-get install gstreamer0.10-ffmpeg
```

Restart Matlab and check if it has fixed the problem. If not, try the following[5]:
We need to ensure that the system's libstdc++.so.6 version is higher than the one shipped with Matlab.

- In a terminal, navigate to matlabroot/sys/os/glnxa64 and type the following:
  `ls -l`
  Replace `matlabroot` by the path of the folder where you installed Matlab. The version of the library shipping with Matlab should be libstdc++.so.6.0.17 or higher.

- Go to /usr/lib/x86_64-linux-gnu and type the following:
  `ls -l libstdc++*`
  If the resulting version is higher than 6.0.17, then continue with the following steps.

- Navigate to matlabroot/sys/os/glnxa6 and execute:
  ```
  unlink libstdc++.so.6
  ln -s /usr/lib/x86_64-linux-gnu/libstdc++.so.6 libstdc++.so.6
  ```

- Restart Matlab.

[3]Tested on Ubuntu 16.04

[4]Solution from: https://uk.mathworks.com/matlabcentral/answers/294258-hello-i-have-a-problem-with-videoreader-on-matlab-r2016a-with-ubtuntu-lts16-04

[5]Solution from: https://www.mathworks.com/support/bugreports/1246784