

Lab 5- Image Processing

The exercises from this lab will introduce image processing with the Raspberry Pi and Python. There are Python scripts to capture webcam images, display and alter them. Advanced exercises will include topics of computer vision such as edge detection and blob detection from images and videos. For understanding and executing the Lab exercises you need to familiarize yourself with the Python programming language, Python modules, lists, functions, math operations and loops.

The setup comprises of a webcam with a Raspberry pi and some pre-installed Python image processing libraries.

In this lab python version 3 will be used.

Equipment needed:

- Raspberry Pi
- SD card with Raspbian OS
- Keyboard, mouse, HDMI cable

Software dependencies:

- git
- Python

Contents

Cloning code from GitHub	2
Tasks.....	2
Task 1: Capture an image using a webcam and the Pi	2
Task 2: Change RGB values of pixels in an image	2
Task 3: Edge detection.....	3
Task 4: Blob detection	3
Task 5: Changing individual pixels	3
Challenges.....	4
Challenge 1	4
Challenge 2	4
Challenge 3	4

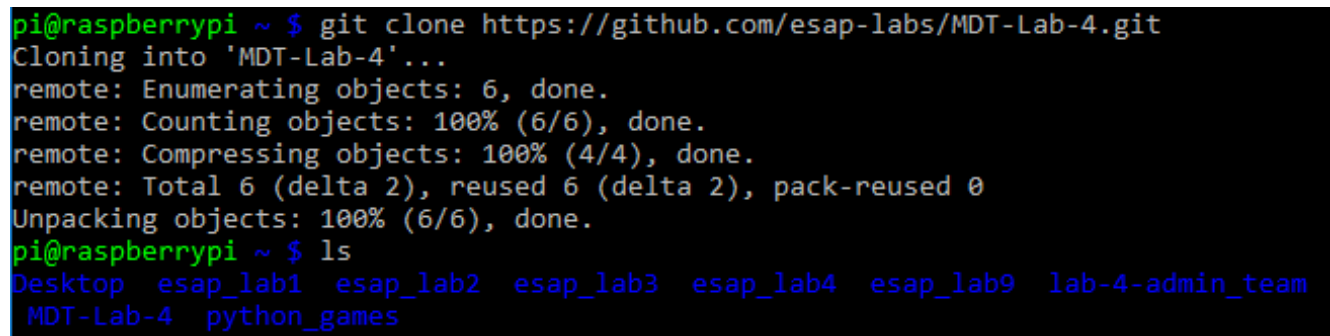
Cloning code from GitHub

Use Git to retrieve all the Python files needed for this lab.

1. 'Clone' the code from GitHub using the following command:

```
git clone https://github.com/esap-labs/MDT-Lab-4.git
```

2. Open directory called 'MDT-Lab-4'
3. All Python files will be in this directory.



```
pi@raspberrypi ~ $ git clone https://github.com/esap-labs/MDT-Lab-4.git
Cloning into 'MDT-Lab-4'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 2), reused 6 (delta 2), pack-reused 0
Unpacking objects: 100% (6/6), done.
pi@raspberrypi ~ $ ls
Desktop  esap_lab1  esap_lab2  esap_lab3  esap_lab4  esap_lab9  lab-4-admin_team
MDT-Lab-4  python_games
```

Tasks

Task 1: Capture an image using a webcam and the Pi

1. Run the script using `python3 lab5.1.py`
2. Open `lab5.1.py` using `nano` and change the window title to "MDT Lab 4", save the file and rerun it.
3. Note that the program waits for 10000 milliseconds and then closes. Change the default value to 5 second instead.

Task 2: Change RGB values of pixels in an image

Access a given pixel in the image by treating the image as an array, and by providing the x,y position of the pixel. After accessing the pixel values, set the RGB values of a pixel, (between 0 and 255). For example- take every predominantly red pixel in the image and change it to blue.

1. Run the script using `python3 lab5.2.py`
2. Notice that this script has a bug! It is not changing predominantly red pixels to blue. Instead it's changing bluish pixels to blue. The problem is that `openCV` expects utilizes the `BGR` format but our code is written to expect `RGB` format.
3. Fix this by creating another array that utilizes the `rgb` format before your for loop.

```
rgb=cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
```

Then modify the code: `r,g,b=image[x,y]` to:

```
r,g,b=rgb[x,y]
```

4. The if statement ensures that a pixel is changed to blue only if red is greater than green and red is greater than blue. Modify that line to add an additional constraint that the value of the red channel must be greater than 128
5. Change the name of the first window to “lab 5.2” and the name of the second window to “lab 5.2 red filter”

Task 3: Edge detection

Implement an edge detection algorithm based on the Sobel operator. This algorithm works by calculating the gradient of the intensity of the image at each point, finding the direction of the change from light to dark and the magnitude of the change. This magnitude corresponds to how sharp the edge is.

1. Run file lab5.3.py
2. Note that width and height are hard coded in the for loops. This is bad coding practice. OpenCV will give you the width, height and number of channels using:

```
height,width,channels=image.shape
```

Enter this code before the python loops and use the width and height variable instead of the hard-coded values.

3. It would be nice if we had a function we could use any time we wanted to apply a Sobel filter to an image. Implement a python function, `def sobel(img)` that takes a cv2 image as input and returns a sobel filtered image as output. Modify your code to use this function and display the returned image.
4. Change the name of the first window to “task 1.3” and the name of the second window to “Sobel filter”

Task 4: Blob detection

Detect a blob of colour in an image captured by camera. It will try and find something nearby which is bright green, red or blue. This is done by filtering out the other colours. Do this by iterating over each pixel, and by testing its red, green and blue intensities against each other. Then we will overlay a marker on the blob region which will be drawn in its centre.

1. Run lab5.4.py

Task 5: Changing individual pixels

Starting with your code from Task 2, set the value of pixel at x position 120 and y position 64 to an orange colour (whose RGB value is 255, 128, 0).

Save your program as lab5.5.py and run it.

Challenges

The following challenges all involve the code written in Task 5.

Challenge 1

Notice that you have to squint to see the pixel in task 5. Instead of just a pixel, make an 8x8 square with an upper left corner at 120,64 and the same orange colour.

Create a python function `make_square` that will draw a square of any size on the image. The function should take as input an image (as an array) the upper left corner (as a tuple), the size of the square (as an integer) and the colour as a vector of length 3.

```
def make_square(img, upper_left, size, colour):
```

Use the function to make 3 squares on the image.

Save your code as challenge1.py and run it.

Challenge 2

Using the same technique used in challenge 1.2, create a solid circle, the colour of your choice, with radius 4 centred at 100,24 drawn on top of the image taken from the camera.

Recall the general equation of a circle is;

$$(x-h)^2 + (y-k)^2 = r^2$$

where h,k are the the coordinates of the centre and the radius is r.

Another form of the equation is:

$$x = h + r \cos(t), \quad y = k + r \sin(t)$$

Either can be used for this exercise.

Hint: import the python math library or numpy and use the appropriate functions.

Save your code as challenge2.py and run it.

Challenge 3

Starting with the code from challenge 2, create a python function `make_circle` that will draw a circle of any radius anywhere on the on the image. The function should take as a parameter the image (as an array), the centre of circle (as a tuple), the radius (as a scalar), and the colour as a vector of length 3. Use the function to draw a circle on the image shown.

```
def make_circle(img, center, radius, color):
```

Use the sobel function you wrote in Task 3 on the image with the circle.

Save your code as challenge3.py and run it.

Challenge 4

Implement a python script to detect corners in the below provide image

Image file can be downloaded from:

https://drive.google.com/file/d/1xhu4IVX_WW5IJPwrEp60msT3KsdaWJVY/view



Hint: (Algorithm: Harris Corner Detector)

Also save the output as 'detected corners.jpg' _b_y_ using cv2.imwrite command.

Expected Output:

