# LAB # 02: Introduction to Python and OpenCv WRT Image processing

## Lab Objective:

The objective of this lab is to introduce the student with OpenCv especially with respect to image processing.

## Lab Description:

**Open** Source **C**omputer **V**ision Library (**OpenCv**) is an open source software library written in C++ for machine learning and computer vision applications.

**To install** packages in PyCharm Click, Files > Settings > Project: Name > Project Interpreter

Now click on '+' icon to install new packages


Now search for following packages and Install them by clicking 'Install Package'

- opencv-python
- numpy // Used for numerical operations
- matplotlib // Used to plotting graphs

**Why OpenCv?**

OpenCv is comprehensive collection of more than 2500 machine learning and computer vision algorithms that can be used from something as simple detecting faces in images to project augmented reality overlaid with scenery.

Another area in which OpenCv excels is its superior support for multiple interfaces and all the major operating systems as compared to other solutions. OpenCv supports Windows, Linux, OS X and Android and provides interfaces for C, C++, Python, Java and MATLAB.

Some of the applications that can be accomplished easily with OpenCv are: identifying objects, tracking camera movements, stitching images together, finding similar images in a database using an image, face detection and tracking moving objects in a video feed etc.

## Some Useful Commands:

1. To slice a 2D array:

   x = y [row_start: row_ end, col_start: col_end]

2. To create a 2D array of zeros using NumPy:

   my_array = **numpy.zeros ((row, columns), dtype=numpy.uint8)**

3. To create a 2D array of ones using NumPy:

   my_array = **numpy.ones ((row, columns), dtype=numpy.uint8)**

4. To check the size of a 2D array: size = **numpy.shape(my_array)**

5. To join a sequence of arrays along an existing axis:

   F = **np.concatenate ((a, b), axis=0);**

6. To assemble an array from nested lists of blocks.

   img = np.block **([[np.ones ((2, 2)), np.zeros ((2, 3))], [np.zeros ((2, 2)), np.ones ((2, 3))]])**

   Note: all the input array dimensions except for the concatenation axis must match exactly

7. Reading an image using OpenCv: my_image = **cv2.imread("test_image.jpg",0)**

   The second argument determines whether the image is read as a grayscale image or a colored image. **0** is used for reading an image as grayscale and while **1** is used for reading in color. If no argument is passed then the image is read as is.

8. Displaying an image using OpenCv: **cv2.imshow ("Title of the window", my_image**).

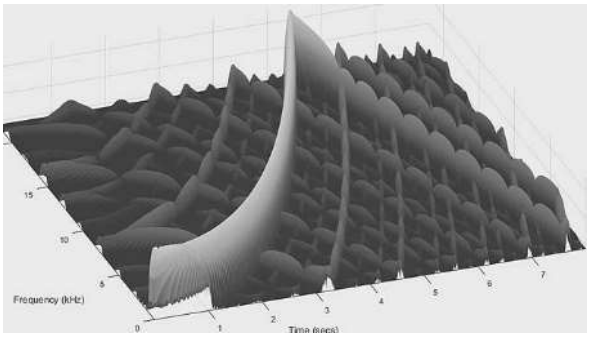
   Two more commands that need to be used while displaying an image are: **cv2.waitKey(x) cv2.destroyAllWindows ().** The waitKey () function waits for a key being pressed for x number of milliseconds. If **0** is passed to waitKey () as an argument, it will wait indefinitely for a key press. **cv2.destroyAllWindows ()** closes are the open image windows.

9. Writing an image to disk: **cv2.imwrite("image_name.jpg", my_image)**

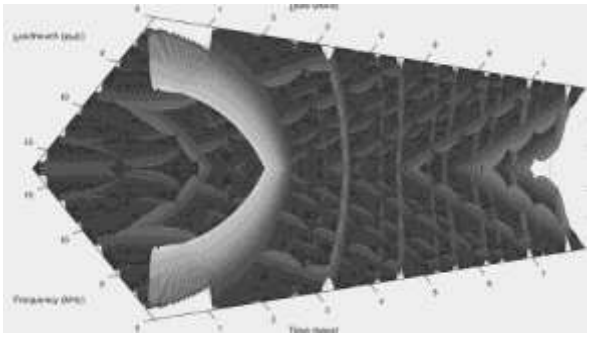
| **LIST** | **TUPLE** | **ARRAY** |
|---|---|---|
| x = [1, 2, 4]<br>print (type (x) ) | y = (1, 2, 4)<br>print (type (y) ) | z = np.array ([1, 2, 4])<br>print (type (z) ) |
| List in memory stores references to objects. Each memory location is a pointer to an object | Same as list however it cannot be changed<br>y [1] = 5 #Error | Array in memory stores a homogenous, densely packed set of numbers of same data type. |
| print ( x*2 ) | print ( y*2 ) | print ( z*2 ) |
| Indexing: l [2][2] | Indexing: t [2][2] | Indexing: a [2, 2] |

## LAB TASKS

**1:** Read any image that you want and save it in gray scale. Now mirror the image that you have read at center i.e. the upper half of the image should be the copy of the lower half but mirrored. Write the image to the disk. See the image below.

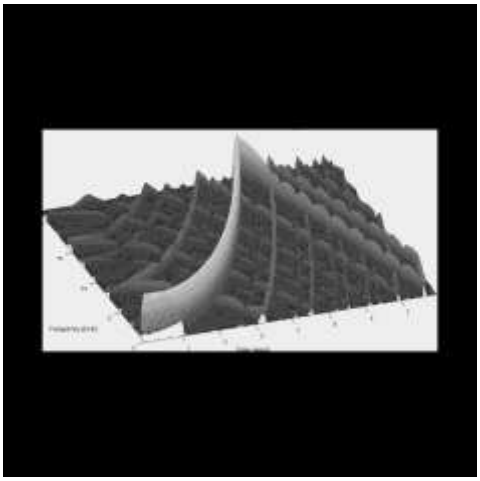


Original

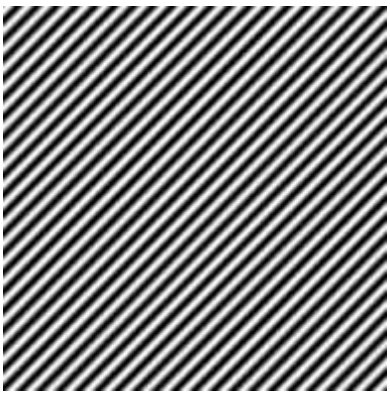


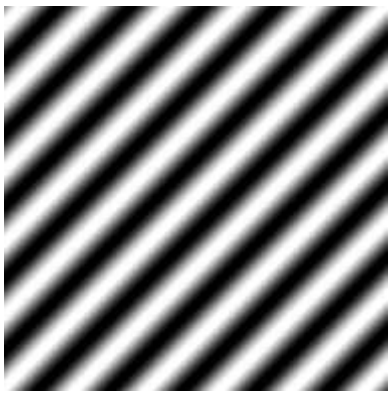
Mirrored

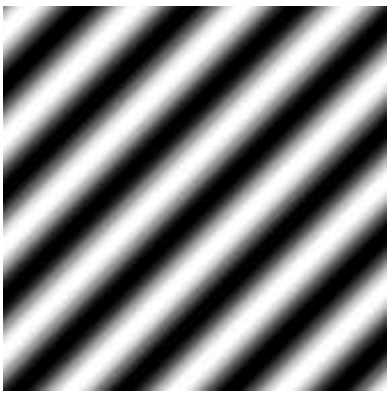
**2:** Now create a generic code that create a border around any landscape image as shown below. The length of right and left borders must be 10% of the original horizontal length of the image. The length of upper and lower border must be such that the image now have same number of rows as columns. Save the image.



**3:** Using the following formula f (i, j) = sin (2 π f (i + j)) where i are j indices of a pixel, draw an image with different frequencies (input from user).

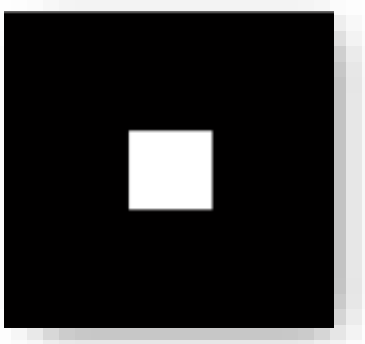
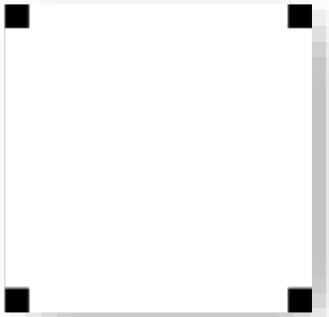
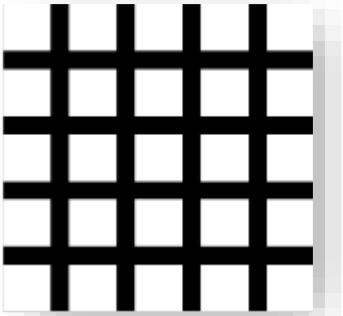






## HOME TASK:

**1:** Write 3 different Python functions that can create the images given below. Code them in such so that the size of the image itself, size of boxes, size of lines and number of horizontal and vertical lines are entered by the user.

**2:** Write a function to create a white image size entered by the user and then create 4 boxes of Black, Blue, Green and Red respectively on each corner of the image as shown below. The size of the colored boxes should be $1/10^{th}$ the size of the image. (**HINT:** the arrays of ones and zeros can be in more than 2 dimensions)

## THINK!!

1. What will be the number of dimension of a grayscale image if opened as colored?
2. Is image a list, tuple or an array?
3. A black and white image can only have what gray levels?
4. Can we flip the image using just one line python code?
5. What is the difference between cv.write and printing an image?