# Homework1_hitenkothari

August 30, 2023

#**ECE 4554/ ECE 5554 / Computer Vision** This file contains the coding problems (Problems 5 and 6) for Homework 1. Your job is to implement/modify the sections within this notebook that are marked with "TO DO".

##**TO DO**: Enter your Virginia Tech Username (PID) here: hitenkothari ##**Honor Code reminder**

Please review the Honor Code statement in the syllabus. This is not a "team project". *Also, if you make use of code from ANY source except the instructor, you must provide comment lines in your code blocks to state where you obtained that code.* Failure to cite sources of code that you borrowed will be considered a violation of the Honor Code.

##**Code libraries** You are allowed to use NumPy and Matplotlib functions to perform matrix operations and graphics/plotting operations. You are also allowed to use any OpenCV functions that are provided in this start-up notebook, but do not use any other OpenCV functions without permission from the instructor.

##**Submission guidelines** for the coding problems (Google Colab)

1. Please verify that you have entered your Virginia Tech Username in all of the appropriate places.
2. After your solutions are complete, click Runtime->"Restart and run all"; then verify that all of your solutions are visible in this notebook.
3. Click File->Save near the top of the page to save the latest version of your notebook at Google Drive.
4. Verify that the last 2 cells have executed, creating a PDF version of this notebook at Google Drive. (If you face difficulty with this step, please refer to https://pypi.org/project/notebook-as-pdf/)
5. Look at the PDF file and confirm that all of your solutions are displayed correctly there.
6. Download your notebook file and the PDF version to your laptop.
7. On your laptop, create a ZIP version of this notebook file. (Please don't include any separate data files.) Use file name Homework1_Code_USERNAME.zip, with your own Username.
8. For your PDF version, use file name Homework1_Notebook_USERNAME.pdf, with your own Username.
9. **Submit these 2 files and your PDF file for Problems 1-4 SEPARATELY to Canvas.** Do not zip them all together.

#**Environment Set-up**

[1]:
```
# Mount your Google Drive to this notebook
# The purpose is to allow your code to access your files
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[2]:
```
# Change the directory to your own working directory
# You code will be able to read and write files in your working directory
# TO DO: If needed, edit the 'chdir' line to provide your Google-Drive␣
 ↪directory name
import os
os.chdir('/content/drive/MyDrive/Colab Notebooks')
```

[3]:
```
# Import library modules and print version information
import sys
import cv2 # OpenCV library
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image # PIL is the Python Imaging Library

# The following is a substitute for cv2.imshow,
#  which you would use on your local machine but Colab does not support it
from google.colab.patches import cv2_imshow

print('Python version:', sys.version)
print('OpenCV version:', cv2.__version__)
print('NumPy version: ', np.__version__)
```

```
Python version: 3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0]
OpenCV version: 4.8.0
NumPy version:  1.23.5
```

### #Problem 5: Working with Python and NumPy (10 points)

a) For each of the following Python commands, write a brief sentence or phrase that describes the result. (You don't need to write the numerical result.) Try to guess the outputs before running the commands interactively. If needed, consult "Familiarization with Colab" and other documentation.

[4]:
```
a = np.array([[1,2,3], [4,5,6]])
b = a[1,:]
c = a[:,2]
d = np.matmul(a.T, a)
e = 0.5 * np.ones_like(a) + 0.2

v = np.random.randn(3,4)
w = v[v>=0]
x = np.arange(0,10)
```

```
y = x.reshape(2,5)
z = np.random.permutation(10)
```

TO DO: Write your descriptions here, in this text box. 1. This will create a 2 by 3 numpy array stored in variable a. 2. b consists of 2nd row of a (python rows and columns start from 0). 3. c consists of 3rd column of a. 4. d is matrix multiplication of a and its transpose (3 by 2 and 2 by 3) resulting in a 3 by 3 numpy array. 5. e is a ones numpt array with same shape as 'a' and further linear mathematical operations carried on it. 6. v is a 3 by 4 numpy array with random initial numbers from normal distribution. 7. w is numpy array with having values wherever v is greater than 0. 8. x is a sorted numpy array ranging from 0 to 10, excluding 10. 9. y is a 2 by 5 numpy array made from reshaping x. 10. z is a random permutation array made from numbers within range 0 to 10 (excluding 10).

[5]:
```
# Verify your answers
print(f'a: {a}\n\n b: {b}\n\n c: {c}\n\n d: {d}\n\n e: {e}\n\n v: {v}\n\n w:␣
  ↪{w}\n\n x: {x}\n\n y: {y}\n\n z: {z}')
```

```
a: [[1 2 3]
 [4 5 6]]

 b: [4 5 6]

 c: [3 6]

 d: [[17 22 27]
 [22 29 36]
 [27 36 45]]

 e: [[0.7 0.7 0.7]
 [0.7 0.7 0.7]]

 v: [[ 1.705845   -0.97619838  0.89543904 -0.27477483]
 [ 1.1409108  -0.09852515  0.71000253  0.00589338]
 [-0.51639835 -0.42438956  0.51716624  1.10066157]]

 w: [1.705845   0.89543904 1.1409108  0.71000253 0.00589338 0.51716624
 1.10066157]

 x: [0 1 2 3 4 5 6 7 8 9]

 y: [[0 1 2 3 4]
 [5 6 7 8 9]]

 z: [9 4 5 0 3 2 1 7 6 8]
```

b) Let y be the vector: y = np.array([1, 2, 3, 4, 5, 6]). Use the reshape command to form a new matrix z that looks like this:

[[1,2], [3,4], [5,6]]

[6]:
```
y = np.array([1, 2, 3, 4, 5, 6])
###################################
# TO DO: write the code to create z
z = y.reshape(3,2)

###################################
print(f'z: {z}')
```

```
z: [[1 2]
 [3 4]
 [5 6]]
```

c) Use np.min and np.where to set x to the maximum value that occurs in z (from the previous part of this problem), and set r to the row location and c to the column location where that maximum value occurs. Remember that Python uses 0-indexing.

[7]:
```
###################################
# TO DO: write the code to assign values to x, r, and c
x = np.max(z)
r,c = np.where(z==x)
row,col= r+1,c+1 # increased by one to store in general math terminology
###################################
print(f'x: {x}\nr: {r}\nc: {c}\nrow: {row}\ncol: {col}')
```

```
x: 6
r: [2]
c: [1]
row: [3]
col: [2]
```

d) Let v be the vector: v = np.array([1, 8, 8, 2, 1, 3, 1, 9]). To a new variable x, assign the number of 1's that are present in the vector v. (Try to perform this operation using one line of Python code.)

[8]:
```
v = np.array([1, 8, 8, 2, 1, 3, 1, 9])
###################################
# TO DO: write the code to create x
x = (v==1).sum()

###################################
print(f'x: {x}')
```

```
x: 3
```

e) Use np.random.randint and create an array named u that contains the result of rolling a six-sided die over N=10 trials.

[9]:
```
###################################
# TO DO: write the code to create u
u = np.random.randint(1,7,10)
```

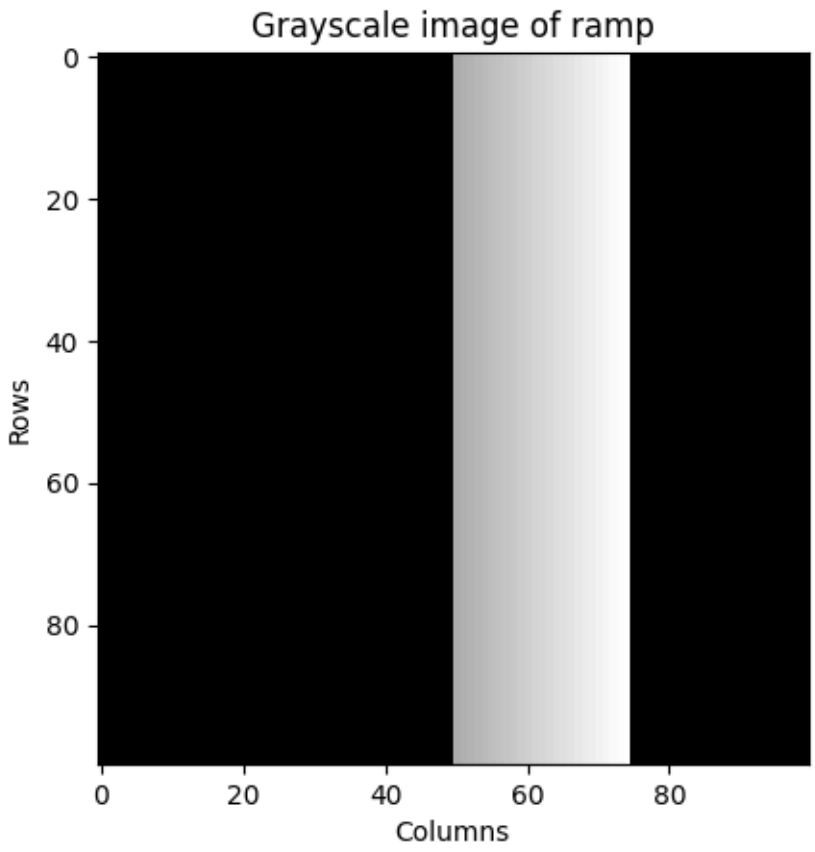
```
###################################
print(f'u: {u}')
```

```
u: [1 6 3 6 4 6 3 6 5 6]
```

f) Create a 100 x 100 matrix of integers named ramp. Initialize each element of ramp to 0, and then modify the matrix as follows: for columns 50 to 74 (inclusive), let c represent the current column index and assign the value c to all rows in that column.

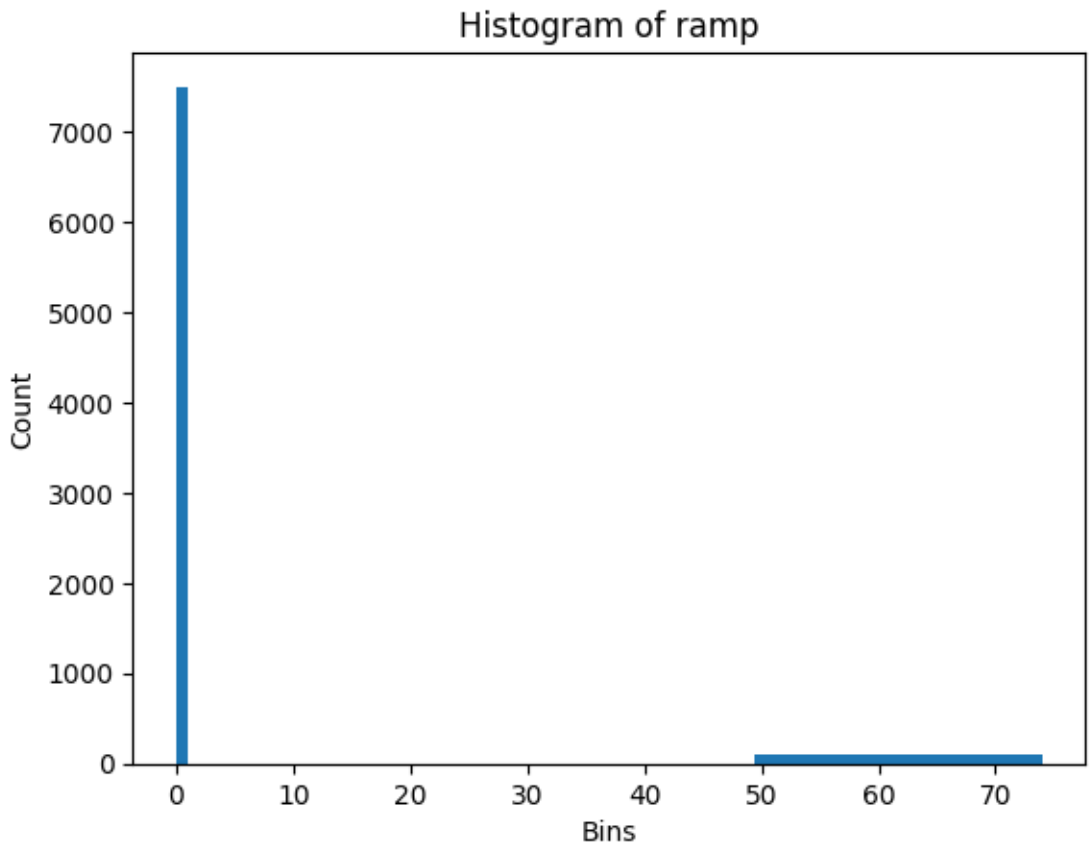
[10]:
```
###################################
# TO DO: write the code
# The following is a temporary assignment to ramp;
#  try it, just for illustration of imshow(), and then replace this line with␣
 ↪your code
ramp = np.zeros((100,100))

for c in range(50,75):
  ramp[:,c]=c

###################################
# The next line should display your matrix ramp as a monochromatic image;
# With a different color map (cmap), the display would be color-coded
plt.imshow(ramp.astype(np.uint8), interpolation='none', cmap=plt.cm.gray)
plt.title('Grayscale image of ramp')
plt.xlabel('Columns')
plt.ylabel('Rows')
print(f'shape of ramp: {ramp.shape}')
```

```
shape of ramp: (100, 100)
```




g) Plot a histogram of the values that are present in matrix ramp. Use 75 bins in your histogram.

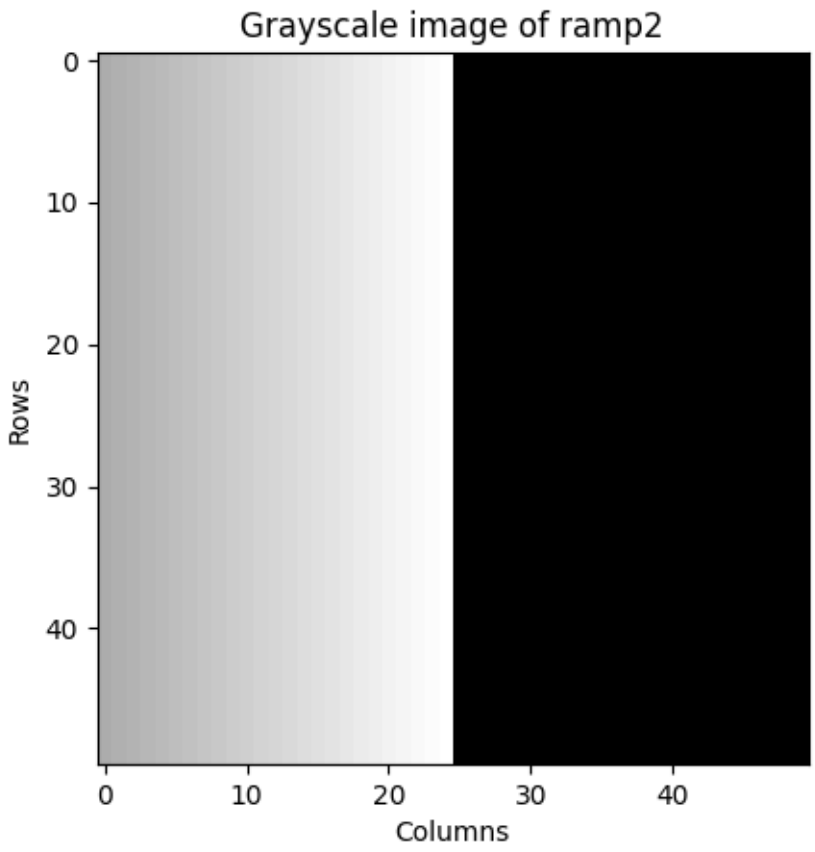
[11]:
```
###################################
# TO DO: write the code
ramp_hist = ramp.flatten() #converting matrix to 1-d array for histogram
plt.hist(ramp_hist,bins=75)
plt.title('Histogram of ramp')
plt.xlabel('Bins')
plt.ylabel('Count')
plt.show()
```




h) Create a new matrix named ramp2 that consists of the lower right quadrant of ramp. Display the result as a monochromatic image.

[12]:
```
###################################
# TO DO: write the code
ramp2 = ramp[ramp.shape[0]//2:,ramp.shape[1]//2:]
plt.imshow(ramp2.astype(np.uint8), interpolation='none', cmap=plt.cm.gray)
plt.title('Grayscale image of ramp2')
plt.xlabel('Columns')
plt.ylabel('Rows')
###################################
```

[12]:
```
Text(0, 0.5, 'Rows')
```

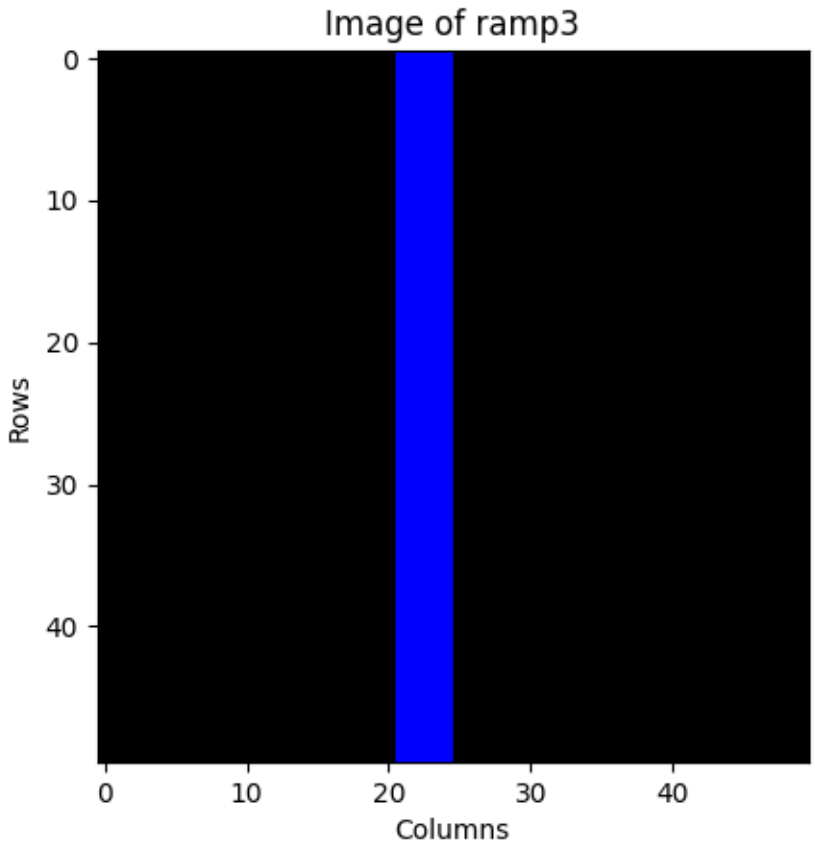



i) Create a new matrix ramp3 that represents a *color* image that has the same number of rows and columns as ramp2, but with 3 channels to represent Red, Green, and Blue values. Set the values in ramp3 to be blue (i.e., R = 0, G = 0, B = 255) wherever the intensity in ramp2 is greater than the threshold t = 70; the values in ramp2 should be black everywhere else. Be careful with type-casting. Display ramp3 so that only black and blue appear.

[13]:
```
###################################
# TO DO: write the code
ramp3 = np.zeros([50,50,3]) #creating a zero matrix having same rows and cols␣
 ↪as ramp2 and three planes for rgb values

for i in range(ramp2.shape[0]):
  for j in range(ramp2.shape[1]):
    if ramp2[i,j]>70:
      ramp3[i,j] = [0,0,255]
    else:
      ramp3[i,j] = [0,0,0]

plt.imshow(ramp3.astype(np.uint8), interpolation='none')
plt.title('Image of ramp3')
```

```
plt.xlabel('Columns')
plt.ylabel('Rows')
###################################
```

[13]:
```
Text(0, 0.5, 'Rows')
```




### #Problem 6: More image operations (10 points)

For this problem, you will write Python/NumPy/OpenCV code that will input an image and then perform some image-based operations.

First, verify that you can read an image from a file, convert it to a new image in grayscale format, and then display both images. You must first upload the file stripes.png to your working directory at Google-Drive. (By the way, this image is a cover album by the musical artists Tegan and Sara.)

Unlike the previous problem, this example uses cv2_imshow to display images.

[14]:
```
# read an image file, and then display color and grayscale versions of the image
img_color = cv2.imread("/content/drive/MyDrive/Colab Notebooks/stripes.png",␣
  ↪cv2.IMREAD_COLOR)
cv2_imshow(img_color)
```

```
print ('\n')
img_grayscale = cv2.cvtColor(img_color, cv2.COLOR_BGR2GRAY)
cv2_imshow(img_grayscale)
```

a) Write a Python function named **subsamp** that accepts a grayscale image as an argument, and returns a smaller (downsampled) version of the input. Your function should determine the dimensions (width and height) of the input image. The dimensions of the output image should be width/2 and height/2, discarding any fractional part from the division by 2.

Your function could compute grayscale pixel values by averaging 4 neighboring pixels. Special rules may be needed near the image border, but for all other pixels the following approach should be used. Using math (not Python) notation, again let (r, c) refer to a particular (row, column) of the output image. The output pixel value at location (r, c) should be computed as the average of the 4 pixels from the input image at locations (2r, 2c), (2r+1, 2c), (2r, 2c+1), and (2r+1, 2c+1).

It can be very helpful to use print/display statements during debugging. However, for the final code that you submit, your subsample1 function should not print or display anything.

[15]:
```
###################################
def subsamp(img_in):
  '''Create a new image that has half the resolution as the input image.

  Input:
    img_in: a grayscale image, of the type provided by cv2.imread

  Return value:
    a new grayscale image that is half the size in both dimensions,
```

```
    and of the same type as img_in

  TO DO: implement the function.
  '''

  height,width = img_in.shape
  output_height,output_width = height//2,width//2

  img_out = np.zeros([output_height,output_width])

  for i in range(output_height):
    for j in range(output_width):
      img_out[i,j] =␣
↪(int(img_in[2*i][2*j])+int(img_in[2*i+1][2*j])+int(img_in[2*i][2*j+1])+int(img_in[2*i+1][2*
↪/4

  return img_out
```

[16]:
```
####################################
# Test your function subsamp by displaying the input and output images
# It should be apparent that the output is half the size
#  (both vertically and horizonally) as the input image
print('Input image:')
cv2_imshow(img_grayscale)
print('Output image:')
img_small1 = subsamp(img_grayscale)
cv2_imshow(img_small1)
```

```
Input image:
```

Output image:

Test your subsamp function using an image that you provide. This new test image must have a different size than stipes.png. In the following code block, write all code that you need to load your new image and use it to test your subsamp function.

[22]:
```
##################################
```

```
# TO DO: write code here to test your subsamp function using an input image␣
 ↪file that you provide
img_color_custom = cv2.imread("/content/drive/MyDrive/Images/eye1.jpg", cv2.
 ↪IMREAD_COLOR) #change the path as per required user file
# cv2_imshow(img_color)

print ('\n')
img_grayscale_custom = cv2.cvtColor(img_color_custom, cv2.COLOR_BGR2GRAY)
# cv2_imshow(img_grayscale)

print('Input image:')
cv2_imshow(img_grayscale_custom)
print('Input image shape: ', img_grayscale_custom.shape)
print('Output image:')
img_small1_custom = subsamp(img_grayscale_custom)
cv2_imshow(img_small1_custom)
print('Output image shape: ', img_small1_custom.shape)
```

```
Input image:
```

```
Input image shape:  (183, 275)
Output image:
```

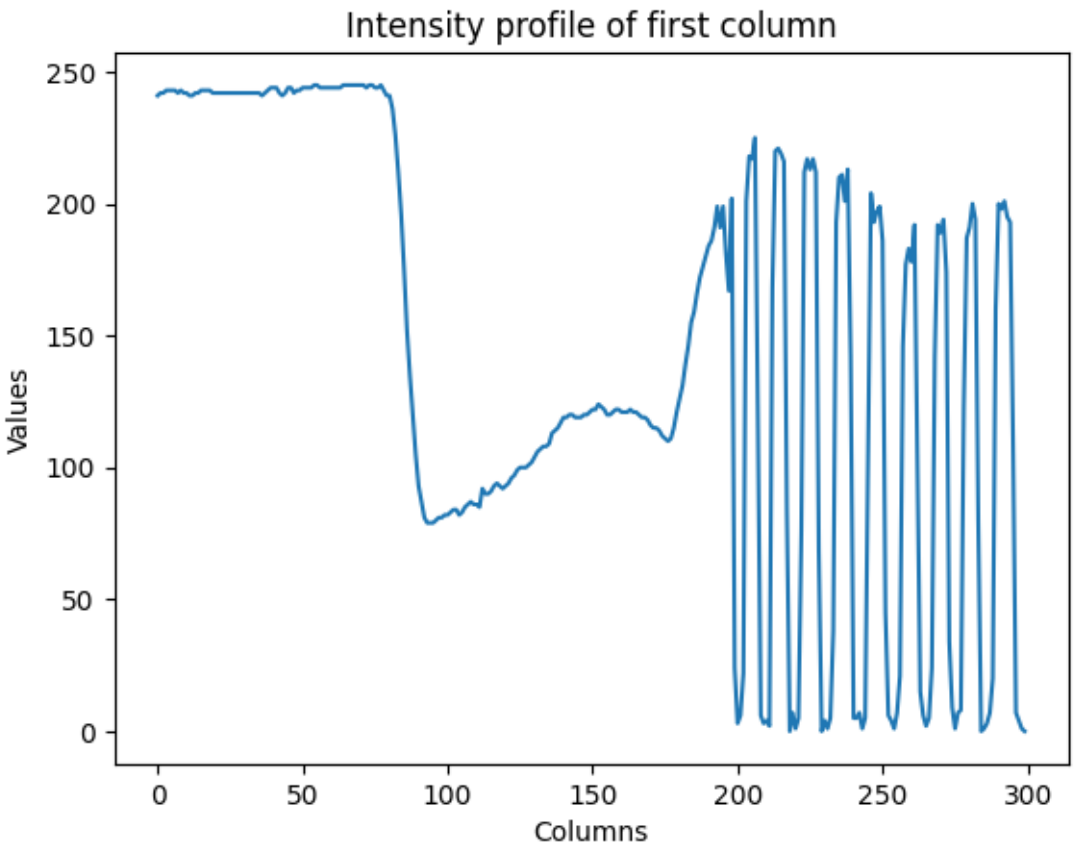
```
Output image shape:  (91, 137)
```

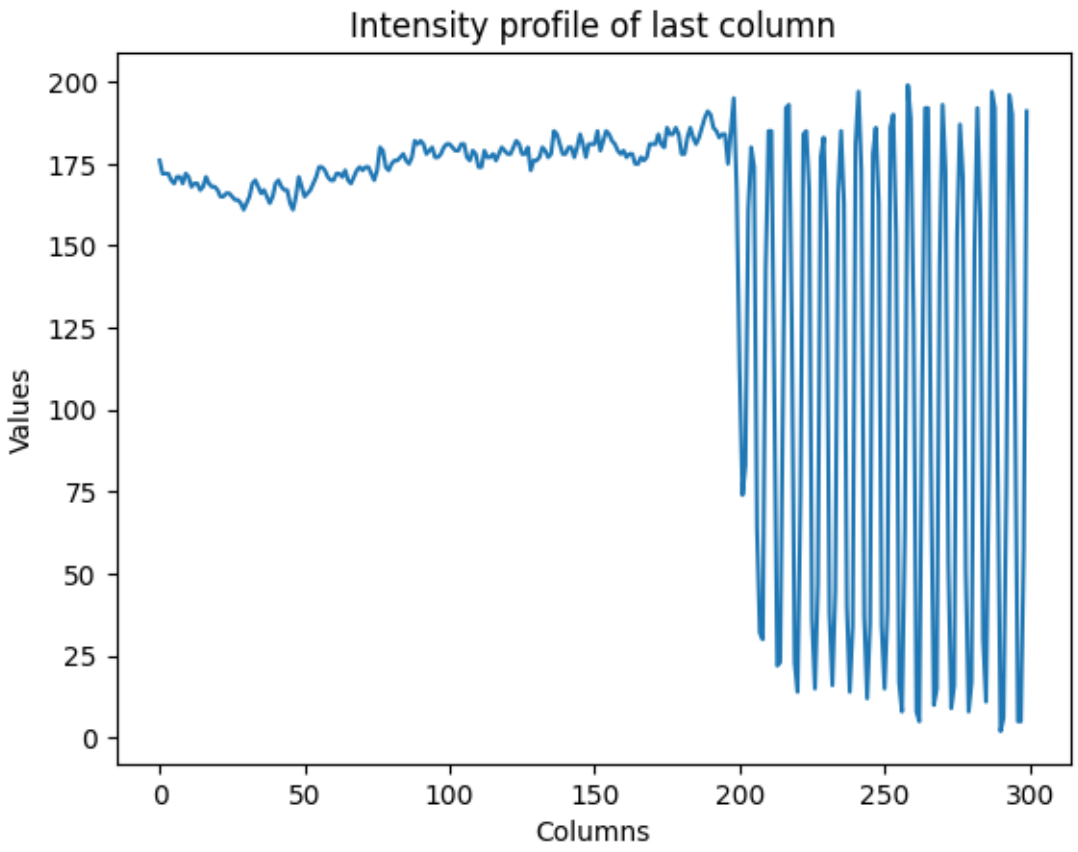
b) Examine those stripes! Consider again the full-size, *monochromatic* version of stripes.png. It is possible to follow a particular column (or row) from that image and generate a plot that is known as an **intensity profile**. Example intensity profiles are shown here: https://imagej.nih.gov/nih-image/more-docs/Tutorial/Profile.html.

In the following code block, write Python code that will plot 2 intensity profiles: one for the first column of stripes.png, and one for the last column of stripes.png.

It is suggested (but not required) that you use functions from `matplotlib.pyplot` module, which you read about in the Familiarization with Colab exercise.

[18]:
```
###################################
# TO DO: write code here to generate the required intensity profiles
plt.plot(img_grayscale[:,1]) #intensity plot of first column
plt.title('Intensity profile of first column')
plt.xlabel('Columns')
plt.ylabel('Values')
plt.show()


plt.plot(img_grayscale[:,img_grayscale.shape[1]-1]) #intensity plot of last␣
 ↪column
plt.title('Intensity profile of last column')
plt.xlabel('Columns')
plt.ylabel('Values')
plt.show()
```


Intensity profile of first column
Values
Columns
250
200
150
100
50
0
0
50
100
150
200
250
300




Creating a PDF version of your current notebook:

[19]:
```
#The following two installation steps are needed to generate a PDF version of␣
 ↪the notebook
#(These lines are needed within Google Colab, but are not needed within a local␣
 ↪version of Jupyter notebook)
!apt-get -qq install texlive texlive-xetex texlive-latex-extra pandoc
!pip install --quiet pypandoc
```

[23]:
```
# TO DO: Provide the correct file name in the line below
!jupyter nbconvert --to PDF "/content/drive/MyDrive/Colab Notebooks/
 ↪Homework1_hitenkothari.ipynb"
```

```
[NbConvertApp] Converting notebook /content/drive/MyDrive/Colab
Notebooks/Homework1_hitenkothari.ipynb to PDF
[NbConvertApp] Support files will be in Homework1_hitenkothari_files/
[NbConvertApp] Making directory ./Homework1_hitenkothari_files
[NbConvertApp] Making directory ./Homework1_hitenkothari_files
[NbConvertApp] Making directory ./Homework1_hitenkothari_files
[NbConvertApp] Making directory ./Homework1_hitenkothari_files
```

```
[NbConvertApp] Making directory ./Homework1_hitenkothari_files
[NbConvertApp] Making directory ./Homework1_hitenkothari_files
[NbConvertApp] Making directory ./Homework1_hitenkothari_files
[NbConvertApp] Making directory ./Homework1_hitenkothari_files
[NbConvertApp] Making directory ./Homework1_hitenkothari_files
[NbConvertApp] Making directory ./Homework1_hitenkothari_files
[NbConvertApp] Making directory ./Homework1_hitenkothari_files
[NbConvertApp] Making directory ./Homework1_hitenkothari_files
[NbConvertApp] Writing 64146 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 593200 bytes to /content/drive/MyDrive/Colab
Notebooks/Homework1_hitenkothari.pdf
```