**Assignment – 1: Filtering - Due: 19 March 2023**

**In this assignment, you will be working alone. It is not allowed to use “convolution filtering” operators through opencv (convolution must be implemented.)**

**Upload your python file (A1.ipynb), and a text file that includes any references used for the assignment. (reference can be a web URL, discussion with the non-group students, etc.)**

**The assignment objective is to understand the convolution operator and kernel size.**

Write a function that filters (convols) the image with a given kernel.

(20p) Compute horizontal gradient (with sobel)→ save as image\_sx

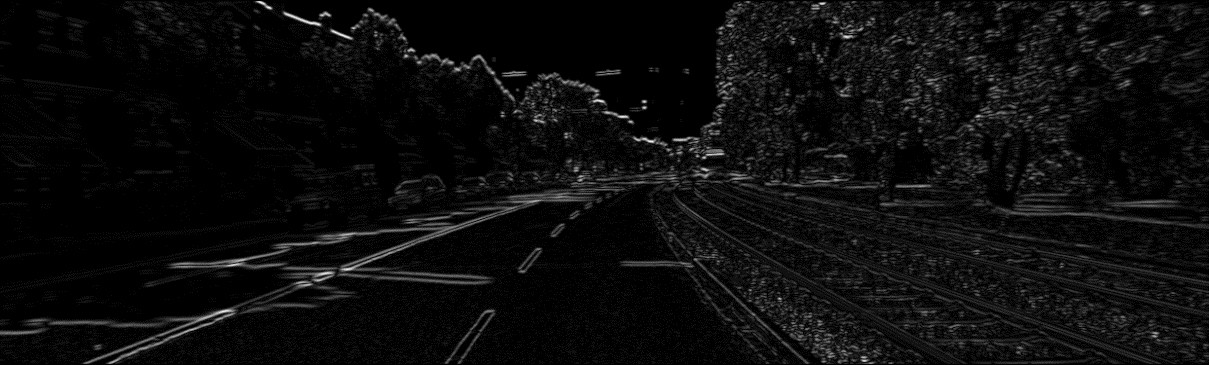
(20p) Compute vertical gradient (with sobel) → save as image\_sy

(20p) Compute the magnitude of gradients → save as image\_grad

(20p) Compute edge orientations  → save as image\_orient

Sample images and sx, sy, grad outputs:





(20p) The Sobel operator is a 3x3 kernel. However, vertical and horizontal gradients can be implemented with different sizes such as 5x5, 7x7, etc. Write a function that generalizes Sobel kernel to any size.

Finally, the main function must take an image and kernel size and writes the desired output. Please follow the [A1.ipynb](https://www.google.com/url?q=https://colab.research.google.com/drive/1CPA09ukD3pLHhHN-RUgT2tLNep9JCKAc&sa=D&source=editors&ust=1679259136458112&usg=AOvVaw2Ut-qtlWSGQvuBcuFKh-Ub) file and create your own copy **(on colab File ->  Save a copy in Drive)** and submit it along with your readme.txt file.

A1.ipynb file content:

import cv2

import numpy as np

import math

from google.colab.patches import cv2\_imshow

# DO NOT USE ANY BUILT-IN CONV. FUNCTIONS

def hsv\_to\_rgb(h, s, v):

    i = int(h\*6.) # XXX assume int() truncates!

    f = (h\*6.)-i; p,q,t = v\*(1.-s), v\*(1.-s\*f), v\*(1.-s\*(1.-f)); i%=6

    if i == 0: return (v\*255, t\*255, p\*255)

    if i == 1: return (q\*255, v\*255, p\*255)

    if i == 2: return (p\*255, v\*255, t\*255)

    if i == 3: return (p\*255, q\*255, v\*255)

    if i == 4: return (t\*255, p\*255, v\*255)

    if i == 5: return (v\*255, p\*255, q\*255)

def gaussian(sigma, radius):

    """

    Computes a Gaussian convolution kernel.

    """

def gaussian\_1D(sigma, radius):

    """

    Computes a 1-D Gaussian convolution kernel.

    """

def generalized\_sobel(ksize):

  # write sobel filter

def my\_conv(img, kernel):

  # write your convolution function

def main():

  !wget -O image.png https://drive.google.com/uc?id=1Mk4u2ulet7XtkaiVB3Im34QxSchMSskt&export=download

  ksize = 1

  sx = generalized\_sobel(ksize)

  sy = sx.transpose()

  # read image

  img = cv2.imread('image.png')

  h,w,c = np.shape(img)

  if c == 3:

    img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

  gx = my\_conv(img, sx);

  gy = my\_conv(img, sy);

  gradient\_strength = np.sqrt(gx\*gx + gy\*gy)

  orientation = np.arctan(gy/(gx+0.00001))

  gx = (gx + 127)

  gy = (gy + 127)

  orientation\_color = np.zeros((h,w,3))

  for i in range(0,h):

    for j in range(0,w):

      orientation\_color[i,j,::-1] = hsv\_to\_rgb((orientation[i,j]+np.pi/2)/(np.pi), 1, gradient\_strength[i,j]/np.max(gradient\_strength))

  cv2\_imshow(gx)

  cv2\_imshow(gy)

  cv2\_imshow(gradient\_strength)

  cv2\_imshow(orientation\_color)

  # write images to a file

main()