Programming Assignment #1

Announcement:	18 January 2022
Submission Deadline:	01 February 2021

Description

The objective of this assignment is to gain an understanding of the methods discussed in the class relating to sampling, interpolation, filtering, and edge detection. You can use C, C++, or Python to implement the solutions. For Python, only OpenCV and NumPy are allowed. Third-party libraries are prohibited, e.g. Pillow, Scikit, skimage, matplotlib, etc.

Part 1: Sampling/Interpolation (3 pts)

A. **Without using OpenCV's resize function,** write a function that takes a colour RGB image and downsamples by factors of 2, 4, 8, and 16. A downsampled image by a factor of n has 1/n of the width and 1/n the height of the original image, i.e. (width/n, height/n). The downsampled images In are calculated by copying every nth column/row, where n is the downsampling factor.

B. Display the images using OpenCV's *imshow* function

C. Using OpenCV's *resize* function, upsample the image I16 by a factor of 10 using the three interpolation techniques: (I) nearest neighbour, (II) bilinear interpolation, (III) bicubic interpolation.

Part 2: Filtering (6 pts)

A. **Without using OpenCV,** write a filter that shifts the image diagonally towards the top right corner.

B.**Without using OpenCV**, write a function that takes as input a neighbourhood size NxN, e.g. 5x5, and a scale σ, and calculates an NxN Gaussian filter. Apply the filter on the given image.

C. **Without using OpenCV,** write a function that takes as input two scales α and β and creates two Gaussian filters. Calculate the difference of Gaussian filtered images and display the result using OpenCV's *imshow* function.

Part 3: Edge detection (6 pts)

A. **Without using OpenCV**, write two functions, each applying the Sobel operators w.r.t to X and Y to the input image. You can assume a fixed kernel size of 3x3. Display the filtered images using OpenCV's *imshow* function.
B. **Without using OpenCV,** write a function for calculating the orientation at each pixel based on the gradient values calculated in part A. Display the orientation map using OpenCV's *imshow* function.
C. **Without using OpenCV,** write a function for calculating the gradient magnitude at each pixel based on the gradient values calculated in part A. Display the gradient magnitude using OpenCV's *imshow* function.
D. Using OpenCV's *canny* function, detect the edges of the image. Display the edge map using OpenCV's *imshow* function.

Part 4: Working with PyTorch (2 pts)
For this part, install and use the libraries: PyTorch (torch, torchvision), matplotlib, and NumPy.
A. Write a script to download the CIFAR10 dataset from the torchvision repository
B. Write a function that randomly picks one image from each class, e.g. airplance, bird, etc, and displays them on screen. A total of 10 classes should be displayed.

Part 5: Extra credit for undergraduates/Compulsory for graduates (5 pts):
A. **Without using OpenCV,** write a function to perform non-maximum suppression. Display the gradient map after thinning.
B. **Without using OpenCV,** write a function to perform all steps of the Canny edge detection as explained in the lectures. This function should only call functions you have implemented, i.e. no OpenCV.

Note: You can use OpenCV's data structures and functions for basic operations in all parts, e.g. Mat, subtract, merge, etc.

Submission (electronic submission through EAS only)
Please create a zip file containing your C/C++ or Python code(**1 source file**) and a readme text file (.txt). In the readme file, document the features and functionality you have implemented and anything else you want the grader to know, i.e. control keys, keyboard/mouse shortcuts, etc, if applicable.

Additional Information

- The source image can be downloaded here.