Visual Computing Fall 2019

Image Processing Assignment

Due:23 Oct, 7pm (Before class)

Instructions

Input images

Download images from the course website to create your own image gallery. You will be using these images for generating the results of your assignment. For each part of the assignment you should show your results using **all of** these images.

Matlab

For this assignment, you should use Matlab. UCI provided free Matlab for students. If you don't have Matlab you can download and install it using your own UCInetID. In the following link, you can find the instruction of installing student version of Matlab for UCI:

http://laptops.eng.uci.edu/software-installation/matlab

Deliverables

One person from each group should upload a **PDF report** on **Gradescope** and a zipfile of your code on Canvas. Add any explanations and the result for each section of the assignment in your PDF report to be graded.

Groups

Each assignment could be done in groups of 2 or individually. The groups for different assignments could be different if you wanted to change your group. At the last page of the PDF report write a small paragraph explaining each person's role and contribution on the assignment.

Submitting to Gradescope

All group members should have an account on Gradescope. The Entry Code for the Gradescope of this class is **9VE6NG**. One person from each group should upload the assignment on gradescope and select which page contains which answer in the question outline. After you submit the assignment, on the top right of the page you see group members name. The submitter's name is there by default and you should edit it and add the other group member. The added group members should receive an email notification that they have joined the group for that assignment.

Part 0. Getting Started

Given a 100 × 100 double matrix **A** representing a grayscale image (hint – use "imread"), write a few lines of MATLAB code to do each of the following. Try to avoid using loops. Also, you may find a RGB image or using uint8 pixel format and hence need to transform it before processing. (Useful commands: rgb2gray, im2double, imresize). For this part, you could use an arbitrary image.

a. Sort all the intensities in A, put the result in a single 10000-dimensional vector x, and plot the values in x.

b. Display a figure showing a histogram of A's intensities with 32 bins.

c. Create and display a new binary image with the same size as A, which is white wherever the intensity in A is greater than a threshold t, and black everywhere else.

d. Generate a new image (matrix), which has the same size as A, but with A's mean intensity value subtracted from each pixel. Set any negative values to 0.

e. Let y be the vector: y = [1: 8]. Use the reshape command to form a new matrix s whose first column is [1, 2, 3, 4]', and whose second column is [5, 6, 7, 8]'.

f. Create a vector [1, 3, 5 …, 99]. Extract the corresponding pixel from the image in its two dimensions, i.e., subsample the original image to its half size.

g. Use fspecial to create a Gaussian Filter and then apply the imfilter function to the image with the created Gaussian Filter, by doing so you should see a blurred image. Change three combinations of parameters of the Gaussian Filter and compare the results.

h. Apply the conv2 instead of imfilter function to the same process (for one Gaussian Filter), do you see any changes? Why?

Hint: For the loading image formats, there might be uint8, unit16 and double type for each pixel. Generally, before any image processing, we would like to change the integer type to double type in order to avoid some unnoticeable side effects (Like fixed maximum value).

Part 1: Gaussian Pyramid

Write a program to generate Gaussian pyramid using convolution. Generate each level of the pyramid by applying a 2×2 box filter to the image in the immediately preceding level.

1/4	1/4
1/4	1/4

Pick your test pictures with size of 2^N in each dimension. Then use the above filter in each level and produce a smaller image with half the size of the image in previous level in each dimension. At the end, you should get a 1 by 1 image with the color that is the average of all pixels in original image.
Because it is hard to illustrate the smaller images in the next step you need to create images with the same size at each level.

For this purpose, use the same filter as Step 1 but resample each level using bilinear interpolation to produce an image with the same size as the original image (One possible way is to use the interp2(image, scale, method)). So, you can see images with different levels of blurring in each level and at the end you should get an image with the same size as original image for each level of Gaussian Pyramid. In other words, you will create a series of intermediate image with dimensions 2^N , 2^{N-1} , ... ,1 and its bilinear interpolated image with the same size as original image.

Create the Gaussian pyramid for all images in the gallery and put the images in the PDF file. Please consider that all the images in different levels should have same size.

Part 2: Laplacian Pyramid

Write a program to generate the Laplacian pyramid by subtracting the consecutive levels of the Gaussian pyramid. Create the Laplacian pyramid for all the images in the gallery and put the results in the PDF format.

Part 3: Multi-Scale Edge Detection

Step 1: Generate the second order derivative images at different scales (or resolution) using a Laplacian operator given below:

-1/8	-1/8	-1/8
-1/8	1	-1/8
-1/8	-1/8	-1/8

Apply the Laplacian operator to every level of the Gaussian pyramid generated in the part 1.

Step 2: Segment the second order derivative image by assigning value to 1 to all pixels of magnitude greater than 0 and value 0 to all pixels of magnitude less than or equal to zero.

Step 3: Detect the zero crossing in the segmented image. This is done by tagging any pixel which has at least one neighbor who is of different value than the pixel itself.

Step 4: Examine the pixels surrounding the zero crossing pixels in the second order derivative image.
Calculate the local variance and mark it as an edge pixel if this value is greater than a certain threshold.

This completes the edge detection.

Show the result for all the images in the gallery in your PDF document.

Part 4: Multi-Resolution Spline.

User your code for creating Gaussian and Lapalcian pyramid to blend two image using following steps:

Step 1: Create the Laplacian pyramid for both images.

Step 2: Create a binary mask to use for blending images. In the binary mask, some of the image areas are given a pixel value of all bits set to 0s and the surrounding areas a value of all bits set to 1s. A simple binary mask example is:

0	0	1	1
0	0	1	1
0	0	1	1
0	0	1	1

Step 3: Create the Gaussian pyramid for the above mask.

Step 4: Now create a new Laplacian pyramid by linear interpolation of two Laplacian pyramids from step 1. For combining each pixel of images in each level of Laplacian pyramids use the value of the generated Gaussian pyramid from step 3 as the coefficient of linear interpolation:

$$LS_l(i,j) = GR_l(i,j)LA_l(i,j) + (1 - GR_l(i,j))LB_l(i,j)$$

LS is the blended image pyramid, GR is the Gaussian pyramid while LA, LB refer to the Laplacian pyramid for the two images. The subscript "*l*" refers to the pyramid level.

Step 5: Now add the images of all levels of the pyramid from step 4.

Choose 3 pairs of images from the gallery. Show the original images, your mask and the final result in your PDF file.