

Assignment #1

CSE 6367-001/102

Student Name:

Spring 2026

Student ID:

Due date: 02/05/26 11:59 pm Central Time

Submission Guidelines:

Submit your source code through Canvas in a single .ipynb file. The name of the .ipynb file should be <lastname>.<firstname>.<StudentID>.ipynb. (Example: Jane_Doe_1001234567.ipynb) The images are available in the ./Images directory. Your TA will use the same directory name to grade your submission. You don't need to attach the images folder with your submission.

1. **(1 pts)** Import the `colors.png` image and apply the following operations:
 - (a) Display the Red, Green and Blue color channels separately in a 1×3 plot using `matplotlib`.
 - (b) Compute a binary mask (Binary Dyadic operation) on the original image based on pixel intensity values where red pixel values are in the range $[100, 255]$ and blue and green pixel values are in the range $[0, 50]$ and show the masked image.
 - (c) Convert the original image to grayscale using the formula

$$I_{\text{Grayscale}} = (I_{\text{Red}} + I_{\text{Green}} + I_{\text{Blue}})/3,$$

and display the grayscale image.

Note: You can use `scikit-image`, `opencv`, or `imageio` to read the image into a `numpy` array.

2. **(3 pts)** Import the `apple.jpg` and the `stop-sign.jpg` images. The images will be in the RGB color space. Create a copy of the images and **manually** convert them to the HSV color space. Then use binary operations to extract all the pixels containing the green apple and the stop sign respectively. Repeat this processing for each image and each color space. Display the original image, the masked image in HSV color space, and the masked image in the RGB color space in a 2×3 plot in `matplotlib`.
3. **(3 pts)** Read the `bandnoise.png` image and display it along with its 2D Discrete Fourier Transform (via the FFT) in a 1×2 `matplotlib` figure. Note that the image has a specific type of noise. Design and apply a filter using the Image's Fourier Transform to remove the image's noise. Finally, display all three images (Noisy image, Fourier Transform image, and clean image) in a single `matplotlib` figure.

Note: You can use any library to compute the FFT. It is helpful to use the properties of the Fourier transform.

4. **(2 pts)** Read the `dog.jpg` image and apply the following point processing to it:

- (a) Create a darker image by subtracting 128 from each pixel intensity value.
- (b) Create a low contrast image by dividing each pixel intensity value by 2.
- (c) Invert the original image where the pixel values are updated as $255 - \text{current pixel intensity value}$.
- (d) Create a brighter image by adding 128 to each pixel intensity value.
- (e) Create a high contrast image by multiplying each pixel intensity value by 2.

Display all of these images along with the original image in a 2×3 grid.

5. **(3 pts)** Read the `cameraman.jpeg` image and apply the following operations:

- (a) Perform average blur with kernel size 9×9 and 25×25 . Display the original image with the filtered/processed images in a 1×3 grid.
- (b) Perform Gaussian blur with kernel size 9×9 , sigma (standard deviation of the Gaussian) of 2.0 and another Gaussian blur with kernel size 25×25 , sigma 15. Display the original image with the filtered/processed images in a 1×3 grid.
- (c) Perform median blur with kernel size 5×5 and 15×15 . Display the original image with the filtered/processed images in a 1×3 grid.
- (d) Resize the Gaussian blurred image with kernel size 25×25 and sigma 15 to 40×40 pixels, and resize the original image to 40×40 pixels. Display the original image with the filtered/processed images in a 1×3 grid.

Note: Apply zero-padding to make the filtered image size same as original image.