Homework 1
M1522.001000_001 Computer Vision (2021 Fall)
Due: Sep 28 Tuesday 03:30PM.

1 Overview

In this assignment, you will implement basic filters and image pyramids using Python. There are also four writing questions at the end.

All the files you need are in `hw1.zip` from the course ETL. Download and unzip the file to create a `hw1` directory. You will do your work here. The directory contains:

- `hw1_[1-2].py`, a partial Python programs that you need to complete.
- `utils.py`, some utility code for handling images. You don't need to modify this file.
- `images`, a directory that contains test images.
- `requirements.txt`, List of Python packages needed for this HW.

Make sure you have Python 3.8 installed. You can check your version with command `python3 -V`. To run the program, you will also need som Python packages including Numpy, Pillow, and Matplotlib. To install them with the package manager for Python, run: `pip3 install -r requirements.txt`.

You can test your program by running each of hw1 files (e.g. `python3 hw1_1.py`). Without your implementation, the program will just load the test images and exit.

Since there are many students in this course, the grading of your homework will be highly automated with (hopefully) little TA intervention. To avoid the risk of your submission being skipped or losing marks, you should make sure:

- Your program runs without an error and is reasonably fast.
- Your program does not use any third-party image processing library (including `ImageFilter` in Pillow, and functions in Opencv). You can import any Python built-in packages you want.
- The names and signatures of the required functions remain unchanged.
- You follow the submission rule below.

After finishing your assignment, zip your files again and submit to ETL.

Finally, note that we will use a code similarity checker to detect plagiarism. You are expected to work on the assignment individually. For your sake, please do not copy and paste other student's code!

2 Programming

2.1 HW1_1.py [20 pts]

In this part, you should implement four functions related to filtering an image. We refer to `input_image` as a three-dimensional Numpy array representing an RGB image, and `Kernel` as a two-dimensional Numpy array which will be applied to the `input_image`. Same *Kernel* must be applied to each of the RGB channels.

- `reflect_padding(input_image,size)`
  Return padded image whose borders are filled with the values mirrored from the *input_image* so that a kernel with the size of *size* can be used for convolutional operations. Note that you must use this padding function in the below functions for this HW.
- `convolve(input_image,Kernel)`
  Return convolved image with the given *Kernel*.
- `median_filter(input_image,size)`
  Return median filtered image. Apply median filter for every pixel in *input_image* and return the resulting image. Note that the filter *size* must be odd.
- `gaussian_filter(input_image,size,sigmax,sigmay)`
  Return gaussian filtered image. Apply gaussian filter for every pixel in *input_image* and return the resulting image. Use *size* as the filter size, *sigmax* and *sigmay* as the standard deviation in X direction and Y direction respectively.

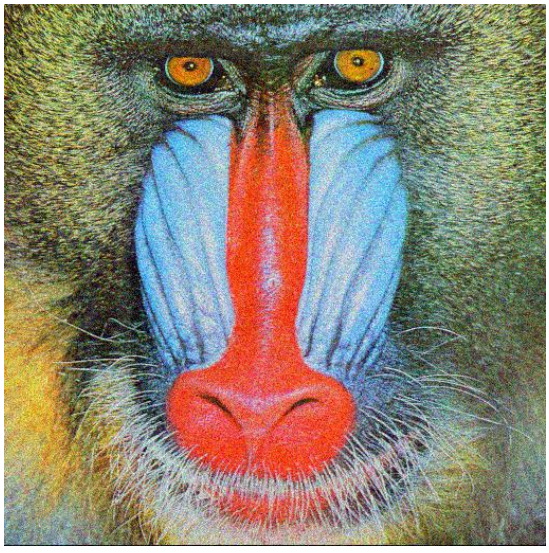


Figure 1: Gaussian noise corrupted test image

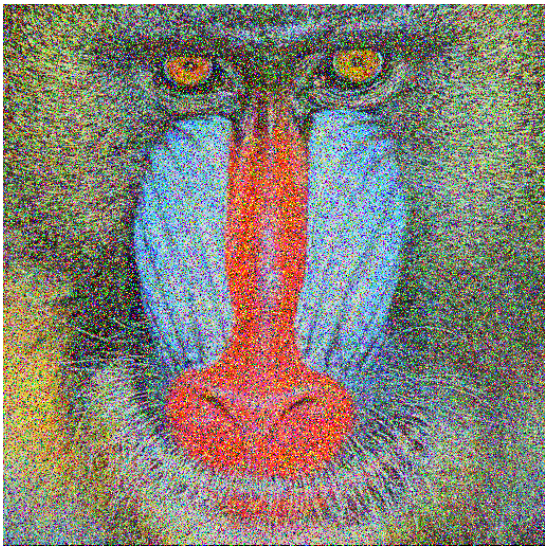


Figure 2: Salt and pepper noise corrupted test image

As you learned in class, Guassian filters are useful for removing Gaussian noise from an image. You can test your Gaussian filter function with the test image `images/gaussian_noise.jpg`. Similarly, you can test your Median filter function with the test image `images/salt_and_pepper_noise.jpg`.

2.2 HW1_2.py [20 pts]

In this part, you will write three functions for blending two images. Implementing functions below, you can freely use functions defined in the `utils.py` which are useful and necessary for completing this HW1_2.py file. Be careful when you do matrix operations on image arrays since the data type of them is unsigned int. Without carefully writing code, you will get undesired output images. Note also that all OpenCV-related functions except the four functions in `utils.py` are prohibited.

- `gaussian_pyramid(input_image,levels)`
  Return a gaussian pyramid with given number of levels. Elements in the list must be arranged in descending order in image resolution.
- `laplacian_pyramid(gaussian_pyramid)`
  Return a laplacian pyramid from *gaussian_pyramid*. Elements in the list must be arranged in descending order in image resolution.
- `blend_images(image1,image2,mask,level)`
  Return blended image. With the functions you built above, you can smoothly blend two images. This function will put an object in the *image2* to the *image1*.

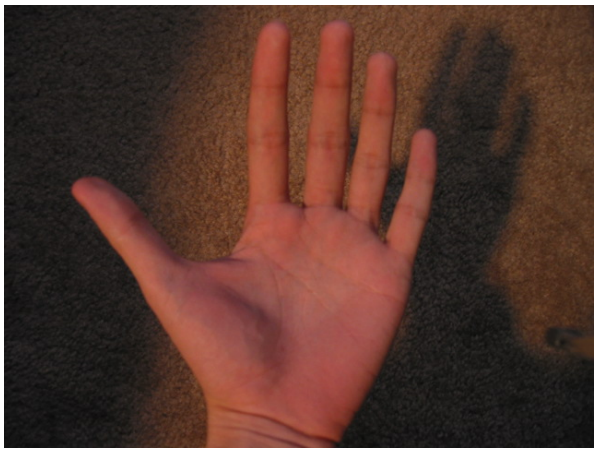


Figure 3: hand (*image1* in `blend_images`)

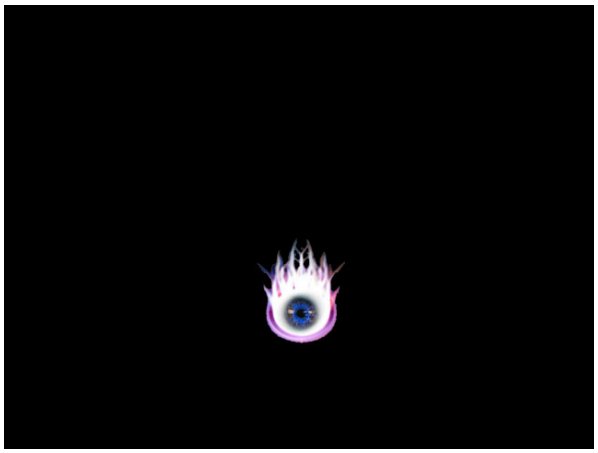


Figure 4: flame (*image2* in `blend_images`)

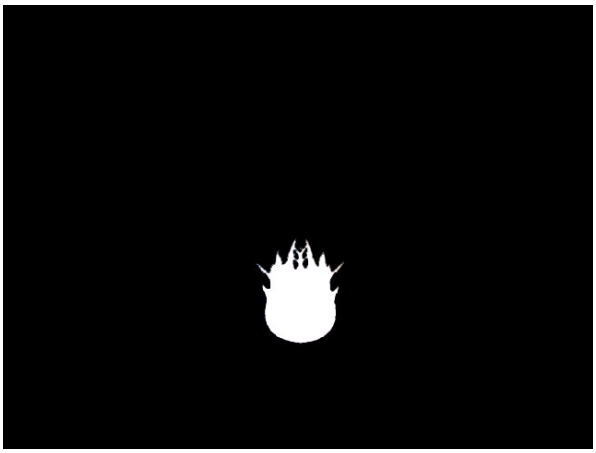


Figure 5: mask (*mask* in `blend_images`)

Test your image blending function with `images/hand.jpg`, `images/flame.jpg` and `images/mask.jpg`. See if the result looks more natural than simple concatenation.

3 Theory

Write a document with your solutions (`solution.pdf`) and include it in the zip file for submission.

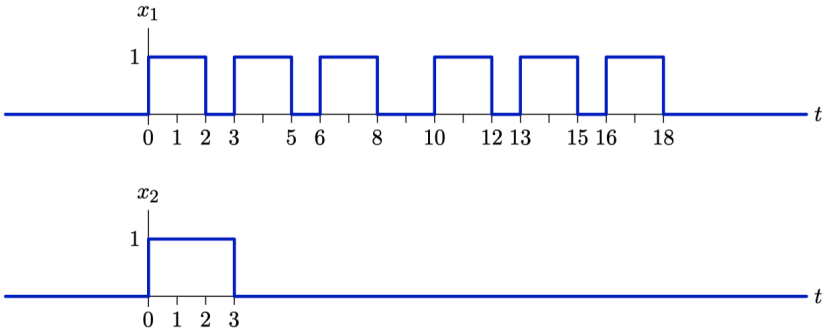
3.1 Composing Filters [10 pts]

Consider the following three filters $\mathcal{G}$, $\mathcal{E}$ and $\mathcal{M}$. $\mathcal{G}$ is a Gaussian smoothing kernel, $\mathcal{E}$ is one of the linear kernels used by the Sobel edge detector and $\mathcal{M}$ is a median filter. Is applying $\mathcal{G}$ to an image followed by $\mathcal{E}$ equivalent to applying $\mathcal{E}$ to an image followed by $\mathcal{G}$? How about if $\mathcal{M}$ is used in place of $\mathcal{G}$? In both cases, explain your answer.

Hint: Think about the properties of convolution.

3.2 Convolution [20 pts]

Signals $x_1(t)$ and $x_2(t)$ are shown in the plots below, and are zero outside the indicated intervals. Draw the result of convolving $x_1(t)$ with $x_2(t)$. Make sure that the important break-points are clear.



3.3 Decomposing a Steerable Filter [10 pts]

In the continuous domain, a two dimensional Gaussian kernel $\mathcal{G}$ with standard deviation σ is given by $G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$. Show that convolution with $\mathcal{G}$ is equivalent to convolving with $\mathcal{G}_x$ followed by $\mathcal{G}_y$, where $\mathcal{G}_x$ and $\mathcal{G}_y$ are 1-dimensional Gaussian kernels in the x and y coordinate respectively, with standard deviation σ. From a computational efficiency perspective, explain which is better, convolving with $\mathcal{G}$ in a single step, or the two step $\mathcal{G}_x$-and-$\mathcal{G}_y$ approach.

3.4 Fourier Transform [20 pts]

a) Find the Fourier transform of the signal $\delta(t-5)$ and draw the magnitude and phase as a function of frequency, including both positive and negative frequencies.

b) Show that if $x_3(t) = ax_1(t) + bx_2(t)$, then $X_3(w) = aX_1(w) + bX_2(w)$.