CS 485/685 Project 1

Image Filtering

Due 2/20/2024 11:59pm

Logistics

Implementation

You must implement everything stated in this project description that is marked with an **implement** tag. You cannot import any packages unless specifically noted by the instructor. In this project, you may import the numpy, math, cv2, PIL, matplotlib, and skimage packages. Unless otherwise specifically noted, you must implement all project elements from scratch (i.e. don't simply call the histogram equalization function from cv2). You are welcome to ask about particular packages as they come up. The idea is to implement everything from scratch.

Deliverables

Each student should submit a single ZIP file, containing their project code (a single `project1.py` file) and your writeup (README.txt). You should create a folder called Project1 (exactly like this with the capital P) and put all your code and your writeup in this folder. Then zip this folder up. That way when I extract your folder, I can run my tests from outside the directory without having to change anything. Your zip file should be named lastname_firstname_project1.zip. Your code should run without errors in a Linux environment. If your code does not run for a particular problem, you will lose 50% on that problem. You should submit only one py file, named `project1.py`. If your file does not match the filename provided exactly, you will lose 50%. Do not worry about what Webcampus does to your file. Each time you submit your project, it will add a -1, -2, -3 onto it. When I download it and unzip it, there is no issue!

Grading

I have provided you with a test script (test_script.py) you can use to test out your functions. I will be using a similar test script, so if your code works with this script it should work with mine! Each student must work independently. You are to submit your own original work. You can work with whatever images you'd like, just use grayscale.

1 Load and Display Images

You will **implement** two functions:

```
load_img(file_name)
```

and

```
display_img(image)
```

You are welcome to use whatever packages you would like to do this (e.g. cv2, PIL, skimage, matplotlib). Try out the different packages to see what kind of differences they have! Choose your favorite and explain why you used this package in the README.txt.

The first function should return an image, and the second function should not return anything.

Assume for the entire project that you are working with grayscale images.

2 1D and 2D Filtering

You will **implement** a function for generating gaussian filters:

```
generate_gaussian(sigma, filter_w, filter_h)
```

If either filter_w or filter_h are 1, we are generating a 1D filter.

This function should return a 1D or 2D filter. Detail your implementation in your README.txt.

You will also **implement** a filtering function:

```
apply_filter(image, filter, pad_pixels, pad_value)
```

This function should take the image, the filter (1D or 2D), as well as the number of pixels to pad on each side and the value with which to pad. If the pad_value=0, then you should pad with 0s, if the pad value equals anything else, you should pad with the edges.
This function should return a filtered image.

Finally, you will **implement** a function to apply a median filter to an image:

```
median_filtering(image, filter_w, filter_h)
```

This function should return a filtered image.

4 Histogram Equalization

Implement a function to perform histogram equalization on a grayscale image.

```
hist_eq(image)
```

This function should return a filtered image.

5 Image Transformation

Implement a function to rotate a grayscale image.

```
rotate(image, theta)
```

Theta represents the angle of rotation. Assume you are rotating around the center of the image and that theta is in radians.
This function should return a rotated image. Detail your implementation in your README.txt.

6 Edge Detection

Implement a function to perform edge detection on a grayscale image.

```
edge_detection(image)
```

You may implement this however you like. You must utilize the four main steps of edge detection as covered in the lecture. But, which filters and approaches you take for each step are up to you! Detail your approach in the README.txt.
This function should return an edge image.