Programming in Computer Graphics – Fall 2023

Homework 1 - Image Processing

Submission Instructions below

Deadline: On canvas

Getting your hands wet with 2D graphics, and related mathematical operations, is a necessary precursor for 3D graphics programming. In this assignment, we will practice with creating various raster graphics objects, generate images, and then manipulate our results. Use the **class assignments** skeleton code to assist you in the tasks below:

1. **Median Filter**
   Implement **medianFilter(filterSize)**, a function which is the median filter we discussed in class, that accepts an input integer number. The input number should be odd, e.g. **num**=3, 9, 21, and similar. The filter would then apply an average filter of size **num X num** to an image. Note that large filter numbers, e.g. 9 and up may run pretty slow.

2. **Gaussian Filter**
   Create a function called **gaussSample(filterSize)** that accepts an input integer number. The input number should be either 3 or 5. If a user provides a non-supported input, you get to decide how the program will behave. The program will apply the relevant gaussian filter below to our input image (illustrated below).

1/16 X

1	2	1
2	4	2
1	2	1

1/159 X

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

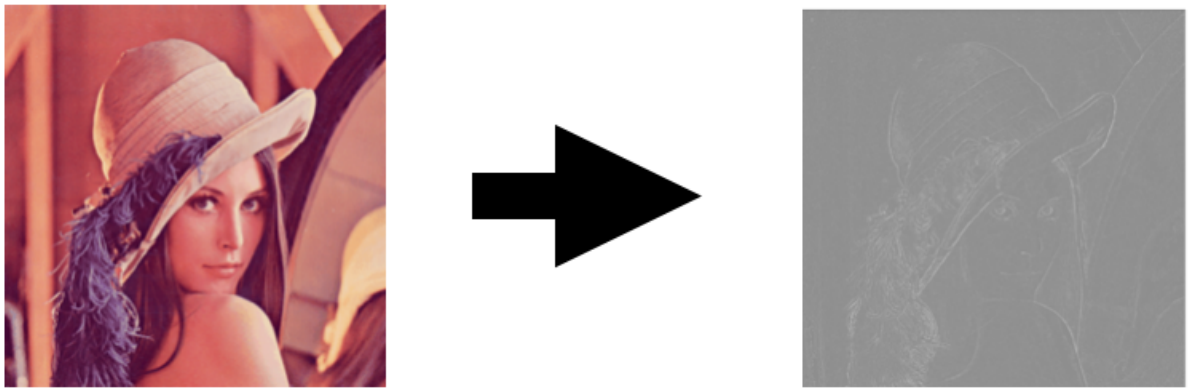
Remember that each destination pixel's color is calculated as a weighted average of its source's neighbors.

3. **Laplacian**

   Implement a function called **laplace()**. In case of a colored image, your code should first convert the image to grayscale:
   greyPixelValue = 0.2126*red + 0.7152*green + 0.0722*blue;
   After the conversion, apply the following filter to your image:

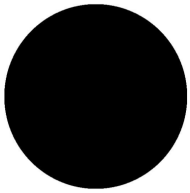
0	1	0
1	-4	1
0	1	0

   Note that after applying this laplacian filter, cell values might be lower than 0 or higher than 255, which is beyond RGB values, which range between 0 and 255. You will therefore need to normalize the value back to the **0 to 255 range**. For example, if the minimum cell value is **min_** , and maximum cell value is **max_**, and the current cell value is cur, then:
   normalized_cell_value = 255 * (cur - min_) / (max_ - min_)



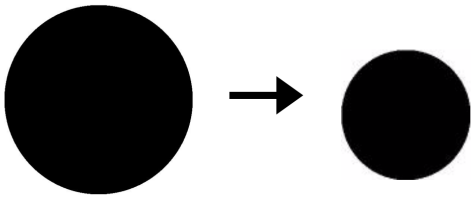
4. **Create an Image of a Circle**

   Write a program called **createCircle(r)** that, given an input integer r, creates an image of a circle of radius r. The image should be 2r pixels wide, and 2r pixels high.
   The circle's interior (not including the perimeter) should be black, and the exterior white. That means, the pixel (x,y) inside the circle should be black if it is less than distance r from its radius.



5. **Scaling images**

   Write a function called **scaleDown()** that takes an input image,, and scales it down in half.



Hence, if the image was of width 10, it will now be of width 5.

Submission and Implementation Instructions

- Submit links to your p5js solutions as a text input in this format:

> My name, my email
> 1. AverageFilter: Link to solution, e.g. editor.p5js.org/tomerwei/sketches/oDmFOdgEb
> 2. Gaussian sampling: editor.p5js.org/tomerwei/sketches/xyz1
> 3. Laplacian: Link to solution editor.p5js.org/tomerwei/sketches/xyz2
> 4. ... and so on

- External image processing packages are not allowed, If unsure regarding specific packages or instructions, contact the instructor.
- You may not use non-approved external javascript libraries in your solution. You are expected to have implemented the code for each of the tasks below by yourself.
- Use the preload function if the assignment expects an input image
- You are not allowed to use 1-line p5js functions that solve the problem for you.

General Instructions

- Submit your solution as a text input on Canvas
- Your program should not crash in normal operation as defined by spec above.
- The program should return within a several seconds for small images and up to a few minutes for large images (say up to 45 seconds is fine).
- If when running your code there is an error, your grade will have an error too.
- You are expected to do your homework by yourself. You can share ideas and discuss general principles with others in the class, but all the code that you submit must be your own work; do not share your homework code with others, and do not look for previous solutions by using a search engine or visiting sites like GitHub. Please consult a TA or the instructor if you have any questions about this policy.
- Regarding image boundary conditions, we treat the pixels after the image boundary as if they are a continuation of the image. For example, the yellow highlighted cells are the image itself, and the ones in white are the “fake” boundary continuation.

15	33	33
15	33	33
90	230	230