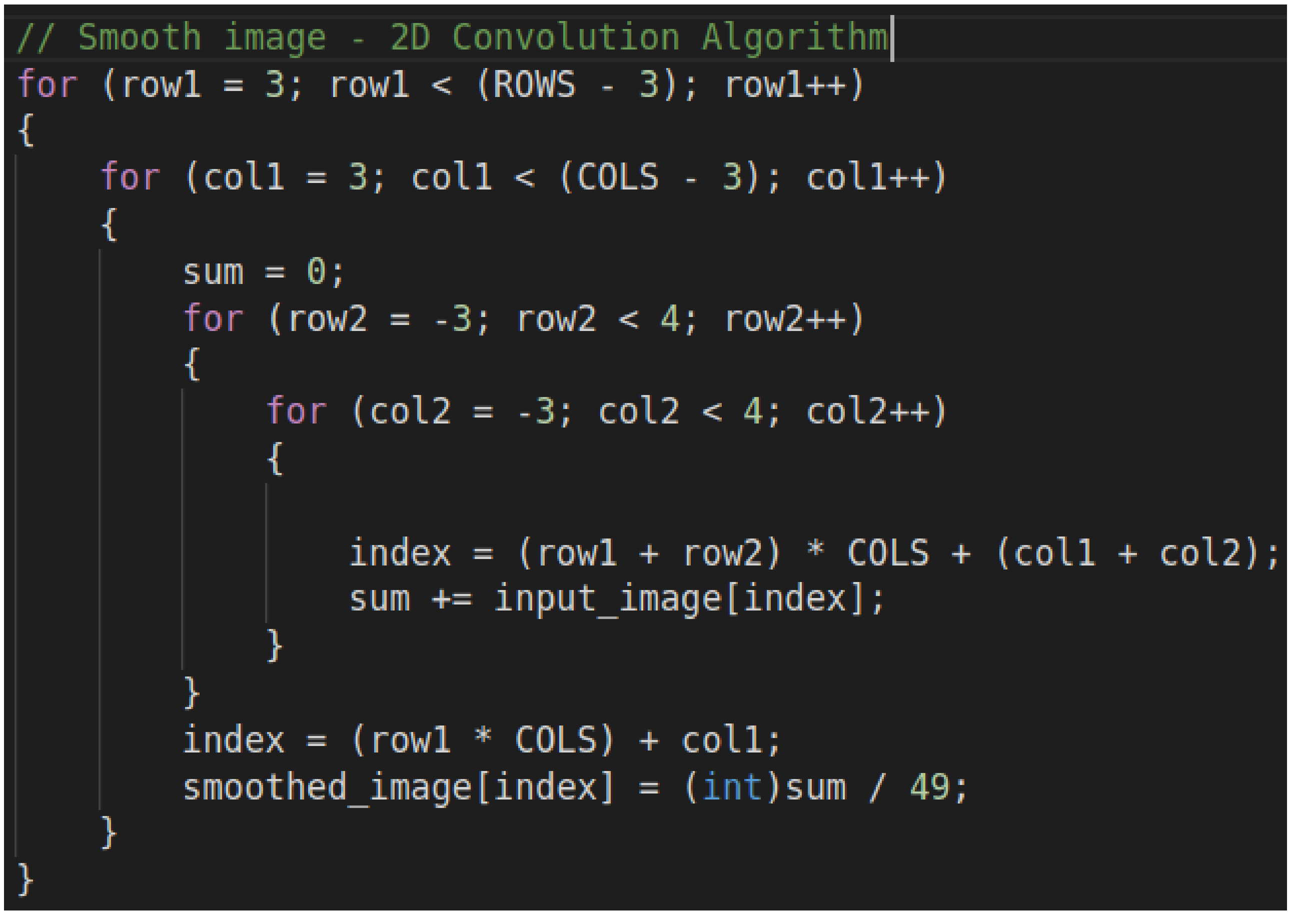
Name: Rodrigo Ignacio Rojas Garcia
Course: ECE 4310
Lab #: 1

Convolution, Separable Filters, and Sliding Windows

In this project the student was to implement three versions of a 7x7 mean filter. The first version was a basic 2D Convolution filter. The second version used separable filters, one 1x7, and the other 7x1. The third version consisted on using the sliding window algorithm and implementing it with the separable filters as well.

2D Convolution:

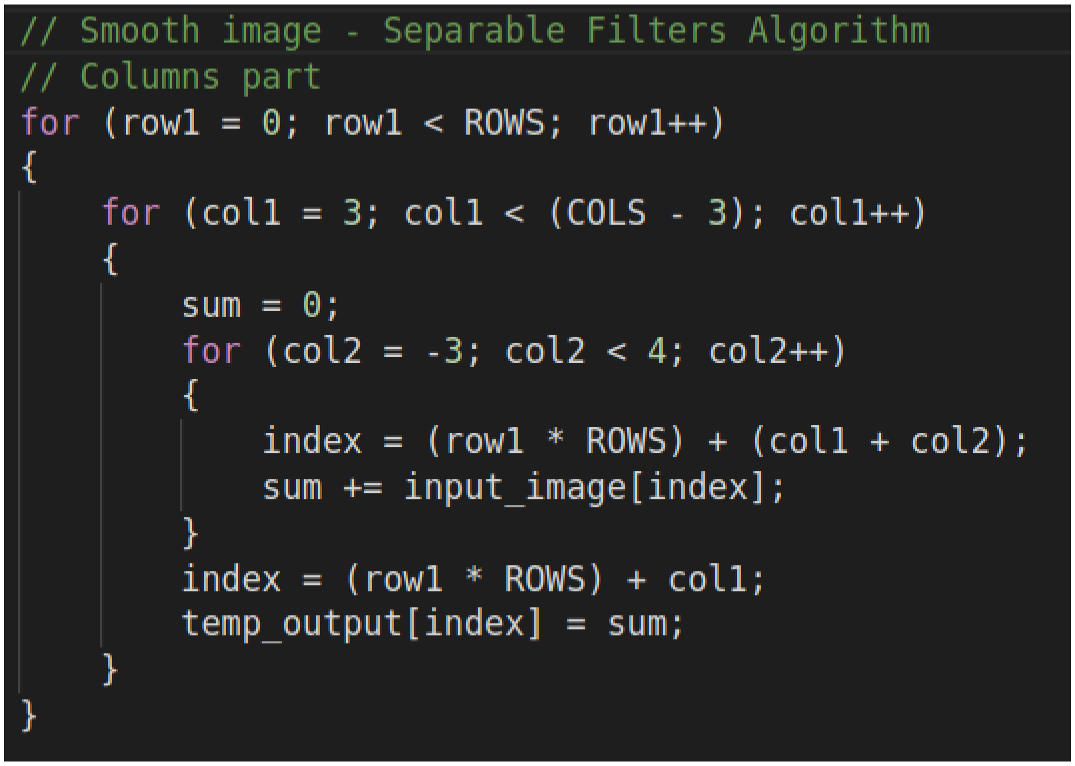
Code:



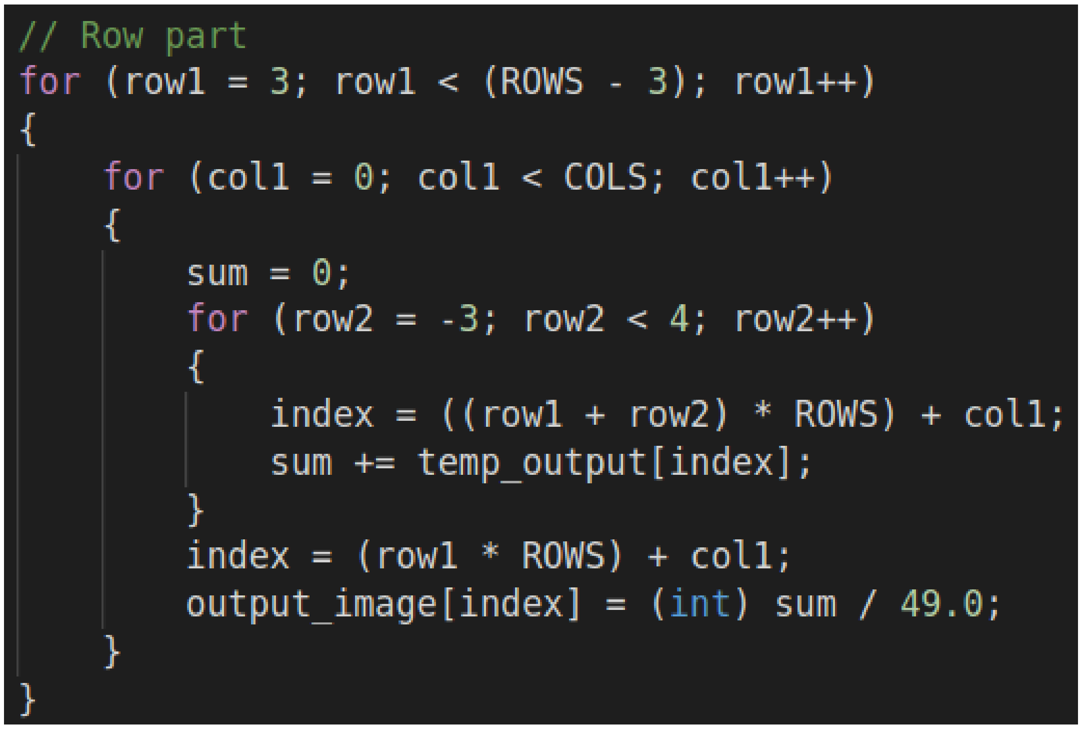
```
// Smooth image - 2D Convolution Algorithm
for (row1 = 3; row1 < (ROWS - 3); row1++)
{
    for (col1 = 3; col1 < (COLS - 3); col1++)
    {
        sum = 0;
        for (row2 = -3; row2 < 4; row2++)
        {
            for (col2 = -3; col2 < 4; col2++)
            {

                index = (row1 + row2) * COLS + (col1 + col2);
                sum += input_image[index];
            }
        }
        index = (row1 * COLS) + col1;
        smoothed_image[index] = (int)sum / 49;
    }
}
```

<u>Separable Filters:</u>

Code:



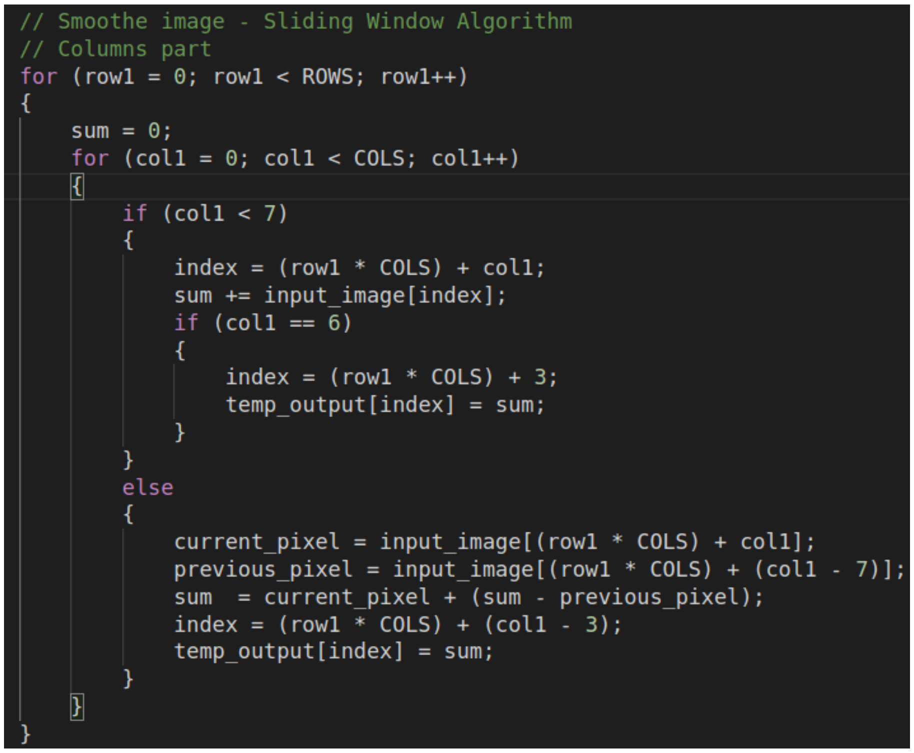
```
// Smooth image - Separable Filters Algorithm
// Columns part
for (row1 = 0; row1 < ROWS; row1++)
{
    for (col1 = 3; col1 < (COLS - 3); col1++)
    {
        sum = 0;
        for (col2 = -3; col2 < 4; col2++)
        {
            index = (row1 * ROWS) + (col1 + col2);
            sum += input_image[index];
        }
        index = (row1 * ROWS) + col1;
        temp_output[index] = sum;
    }
}
```



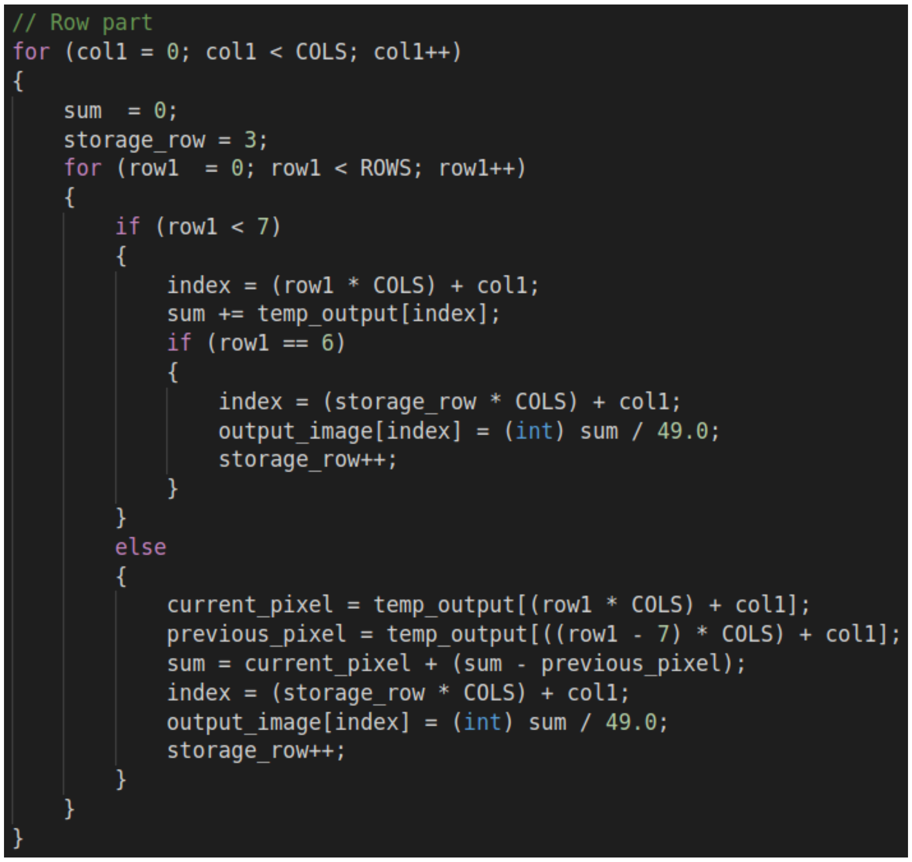
```
// Row part
for (row1 = 3; row1 < (ROWS - 3); row1++)
{
    for (col1 = 0; col1 < COLS; col1++)
    {
        sum = 0;
        for (row2 = -3; row2 < 4; row2++)
        {
            index = ((row1 + row2) * ROWS) + col1;
            sum += temp_output[index];
        }
        index = (row1 * ROWS) + col1;
        output_image[index] = (int) sum / 49.0;
    }
}
```

Sliding Window with Separable Filters

Code:



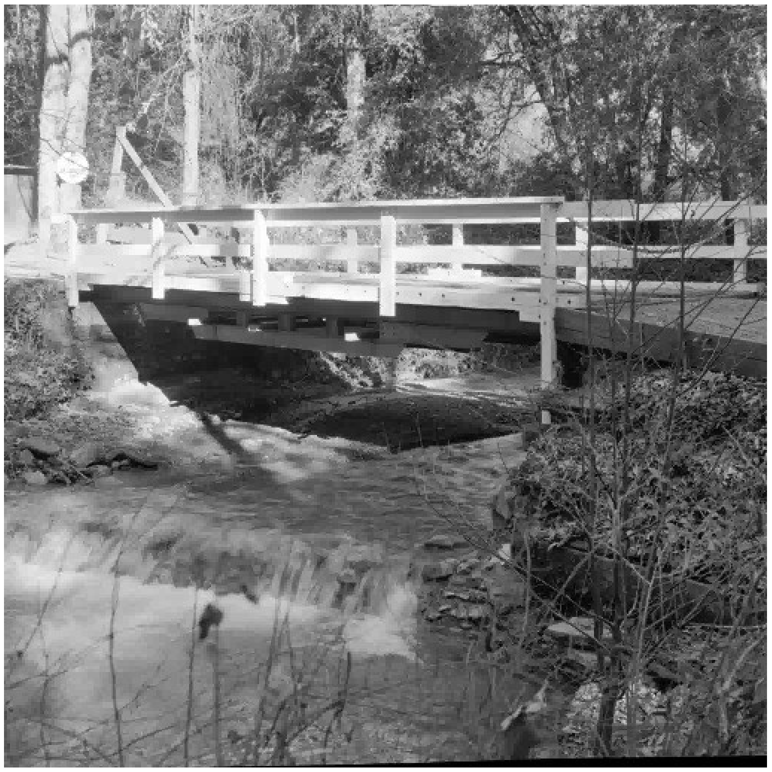
```
// Smoothe image - Sliding Window Algorithm
// Columns part
for (row1 = 0; row1 < ROWS; row1++)
{
    sum = 0;
    for (col1 = 0; col1 < COLS; col1++)
    {
        if (col1 < 7)
        {
            index = (row1 * COLS) + col1;
            sum += input_image[index];
            if (col1 == 6)
            {
                index = (row1 * COLS) + 3;
                temp_output[index] = sum;
            }
        }
        else
        {
            current_pixel = input_image[(row1 * COLS) + col1];
            previous_pixel = input_image[(row1 * COLS) + (col1 - 7)];
            sum  = current_pixel + (sum - previous_pixel);
            index = (row1 * COLS) + (col1 - 3);
            temp_output[index] = sum;
        }
    }
}
```



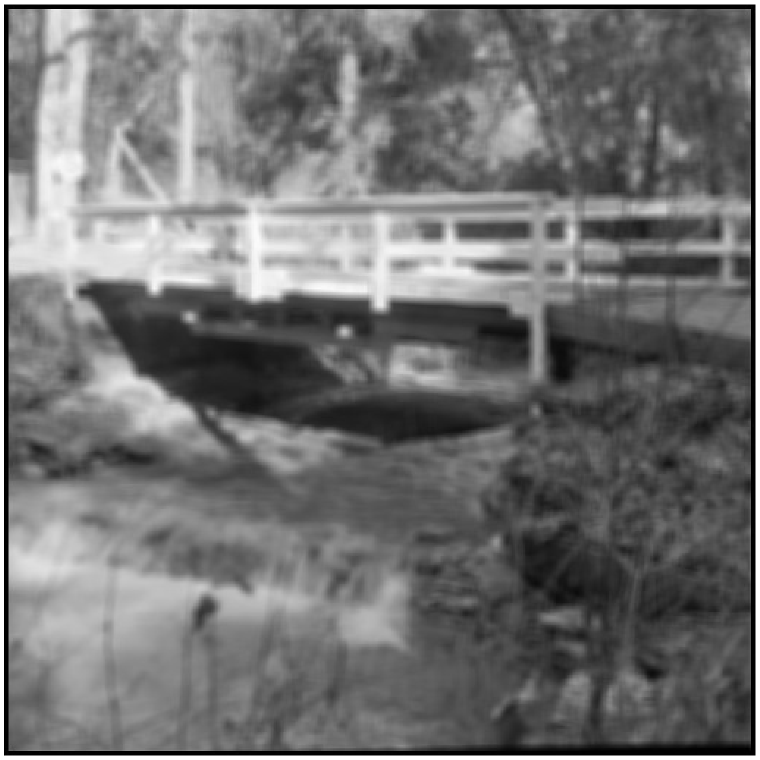
```
// Row part
for (col1 = 0; col1 < COLS; col1++)
{
    sum  = 0;
    storage_row = 3;
    for (row1  = 0; row1 < ROWS; row1++)
    {
        if (row1 < 7)
        {
            index = (row1 * COLS) + col1;
            sum += temp_output[index];
            if (row1 == 6)
            {
                index = (storage_row * COLS) + col1;
                output_image[index] = (int) sum / 49.0;
                storage_row++;
            }
        }
        else
        {
            current_pixel = temp_output[(row1 * COLS) + col1];
            previous_pixel = temp_output[((row1 - 7) * COLS) + col1];
            sum = current_pixel + (sum - previous_pixel);
            index = (storage_row * COLS) + col1;
            output_image[index] = (int) sum / 49.0;
            storage_row++;
        }
    }
}
```

<u>Results:</u>

Original Picture:



Smoothed Picture:

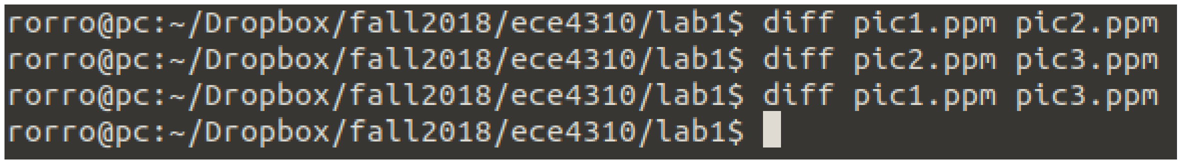


Time:

Time (ms)		
2D Convolution	Separable Filters	Sliding Window
40.71	16.11	13.64
71.16	26.94	17.59
101.79	37.40	21.45
132.76	47.78	25.52
163.24	58.28	29.62
193.88	69.19	33.51
224.51	79.57	37.42
255.22	89.97	41.27
285.79	1300.63	45.14
316.1	111.42	49.02

Average Time (ms)		
2D Convolution	Separable Filters	Sliding Window
31.60	11.14	4.90

Difference:



```
rorro@pc:~/Dropbox/fall2018/ece4310/lab1$ diff pic1.ppm pic2.ppm
rorro@pc:~/Dropbox/fall2018/ece4310/lab1$ diff pic2.ppm pic3.ppm
rorro@pc:~/Dropbox/fall2018/ece4310/lab1$ diff pic1.ppm pic3.ppm
rorro@pc:~/Dropbox/fall2018/ece4310/lab1$ 
```

Conclusion:

It can be concluded that all the different algorithms produce the same picture, but the Sliding Window algorithm is the fastest of all three algorithms.