ECE 6310

Introduction to Computer Vision

Fall 2022

Lab 1

Harshal B. Varpe

In this lab the students were asked to implement three version of 7x7 mean filter. Each implementation was to be timed multiple times and average time values were to be considered to decide which implementation of the filter was faster. The three different version were: A standard 7x7 mean filter, a Separable 7x7 filter, and Separable – Sliding window 7x7 filter. The filters were written in C language with help of baseline code of 3x3 mean filter. The image format is ".ppm". The output images files were supposed to have no differences.

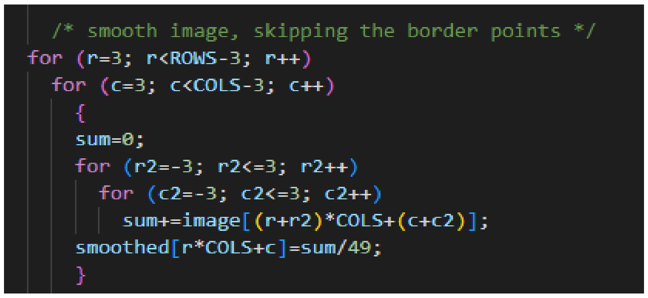
Fig 1 - Input image (bridge.ppm)

The following table shows the how much time each run of each filter took in nanoseconds. Each version of filter was run 10 times.

	Standard 7x7 Filter	Separable 7x7 Filter	Separable – Sliding Window 7x7 Filter
1	21924000	7416800	2987100
2	20967600	6857800	3054300
3	21334800	6887500	2752600
4	21629200	7116800	2899100
5	21450900	7034500	2679800
6	21780800	7046600	3036400
7	21786300	7007400	2769300
8	21383900	6893600	2776300
9	21280300	7290400	3234300
10	21688500	6896000	3136300
Avg	21522630 (21.52263 ms)	7044740 (7.04474 ms)	2932550 (2.93255 ms)

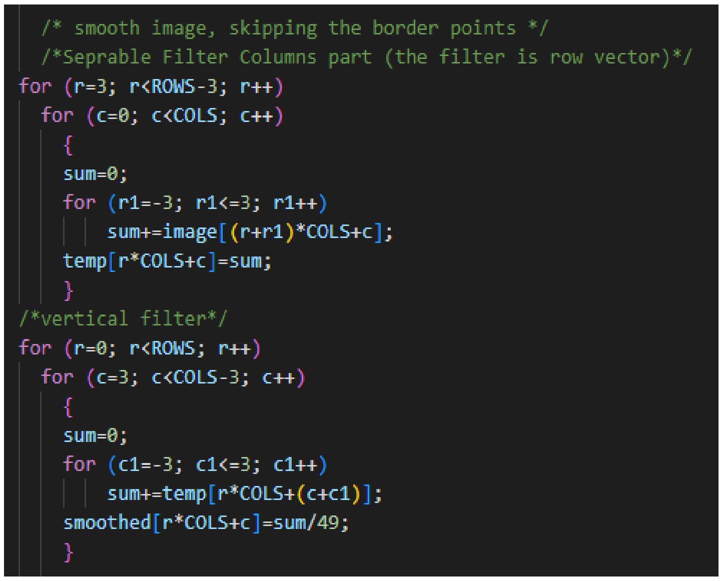
From the table above, we can see that results were as expected. The standard filter requires way more calculations. The four nested for loops required increase the time complexity of the algorithm. Separable filter was approximately 3 times more efficient than the standard filter. This is because we use only three nested for loops, thus decreasing the time complexity a bit. We can also see that the hybrid of separable filter and sliding window filter was the fastest. It was approximately 7 times more efficient than the standard filter and approximately 2.4 times more efficient than the separable filter.

One of the challenges while coding up the separable filter and separable sliding filter, was figuring out the data type for the intermediate image. I had to switch the datatype of the intermediate image to float as compared to the initial and final image. I thought that because of the intermediate calculations, char data type may not be able to store the required data. It may also affect the accuracy of final results since unsigned char may round decimal numbers. Fortunately, there are plenty of stack overflow posts regarding this type of data conversion and accuracy.



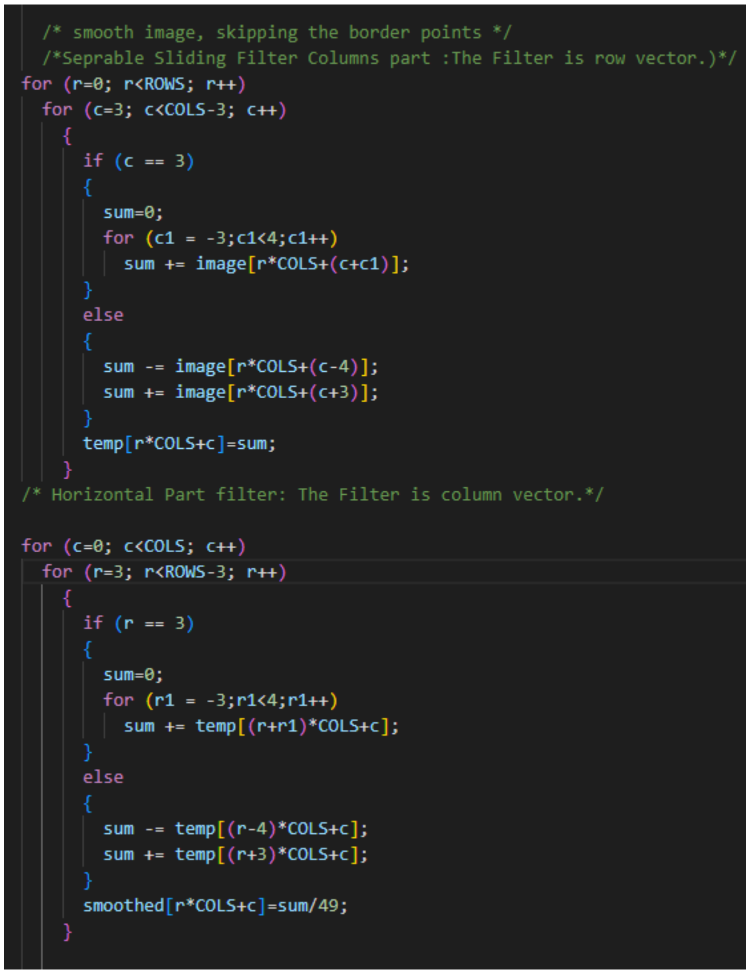
```
  /* smooth image, skipping the border points */
for (r=3; r<ROWS-3; r++)
  for (c=3; c<COLS-3; c++)
    {
    sum=0;
    for (r2=-3; r2<=3; r2++)
      for (c2=-3; c2<=3; c2++)
        sum+=image[(r+r2)*COLS+(c+c2)];
    smoothed[r*COLS+c]=sum/49;
    }
```

Fig 2 – Standard 7x7 mean Filter



```
 /* smooth image, skipping the border points */
 /*Seprable Filter Columns part (the filter is row vector)*/
for (r=3; r<ROWS-3; r++)
  for (c=0; c<COLS; c++)
    {
    sum=0;
    for (r1=-3; r1<=3; r1++)
        sum+=image[(r+r1)*COLS+c];
    temp[r*COLS+c]=sum;
    }
/*vertical filter*/
for (r=0; r<ROWS; r++)
  for (c=3; c<COLS-3; c++)
    {
    sum=0;
    for (c1=-3; c1<=3; c1++)
        sum+=temp[r*COLS+(c+c1)];
    smoothed[r*COLS+c]=sum/49;
    }
```

Fig 3 – Separable 7x7 Filter

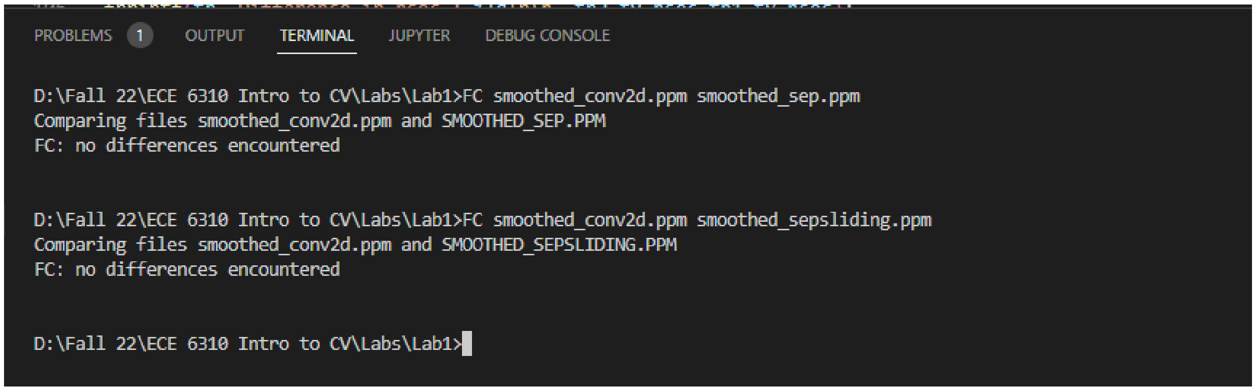


```
  /* smooth image, skipping the border points */
  /*Seprable Sliding Filter Columns part :The Filter is row vector.)*/
for (r=0; r<ROWS; r++)
  for (c=3; c<COLS-3; c++)
    {
      if (c == 3)
      {
        sum=0;
        for (c1 = -3;c1<4;c1++)
          sum += image[r*COLS+(c+c1)];
      }
      else
      {
        sum -= image[r*COLS+(c-4)];
        sum += image[r*COLS+(c+3)];
      }
      temp[r*COLS+c]=sum;
    }
/* Horizontal Part filter: The Filter is column vector.*/

for (c=0; c<COLS; c++)
  for (r=3; r<ROWS-3; r++)
    {
      if (r == 3)
      {
        sum=0;
        for (r1 = -3;r1<4;r1++)
          sum += temp[(r+r1)*COLS+c];
      }
      else
      {
        sum -= temp[(r-4)*COLS+c];
        sum += temp[(r+3)*COLS+c];
      }
      smoothed[r*COLS+c]=sum/49;
    }
```

Fig 4 – Separable Sliding Window Filter

The result images produced had no differences. The differences were checked using FC and diff commands.

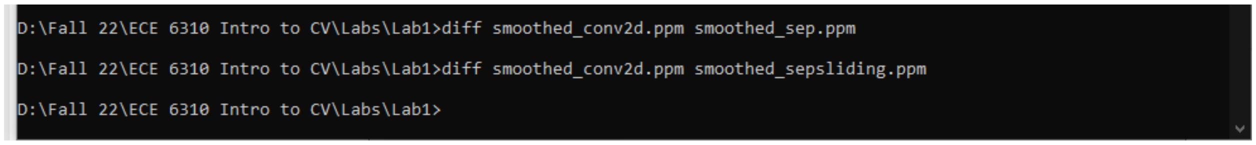


```
D:\Fall 22\ECE 6310 Intro to CV\Labs\Lab1>FC smoothed_conv2d.ppm smoothed_sep.ppm
Comparing files smoothed_conv2d.ppm and SMOOTHED_SEP.PPM
FC: no differences encountered


D:\Fall 22\ECE 6310 Intro to CV\Labs\Lab1>FC smoothed_conv2d.ppm smoothed_sepsliding.ppm
Comparing files smoothed_conv2d.ppm and SMOOTHED_SEPSLIDING.PPM
FC: no differences encountered


D:\Fall 22\ECE 6310 Intro to CV\Labs\Lab1>
```

Fig 5 – FC command to check differences



```
D:\Fall 22\ECE 6310 Intro to CV\Labs\Lab1>diff smoothed_conv2d.ppm smoothed_sep.ppm

D:\Fall 22\ECE 6310 Intro to CV\Labs\Lab1>diff smoothed_conv2d.ppm smoothed_sepsliding.ppm

D:\Fall 22\ECE 6310 Intro to CV\Labs\Lab1>
```

Fig 6 - Diff command to check differences between files.

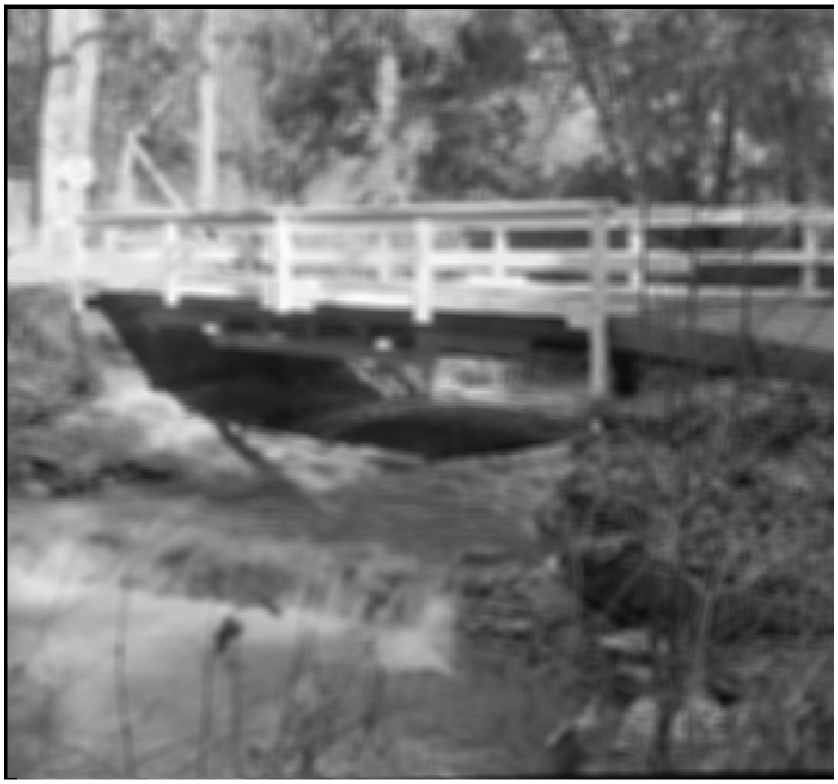


Fig 7 – Output Image