Shashi Shivaraju
XID:C88650674

REPORT
ECE 6310 Lab #1 – Convolution, separable filters, sliding windows

Objective: To implement three versions of a 7x7 mean filter using basic 2D convolution ,separable filters(1x7 and 7x1)an1d separable filters along with a sliding window respectively and to perform time profiling of each version.

Implementation: 7x7 Mean Filter

a) 2D convolution version:
For a KxK filter, 2D convolution requires K^2 operations per pixel.

[Code snippet]:

```
/*2D convolution of 7x7 mean filter with
 * the input image excluding the borders*/
      for(R=N;R<row-N;R++)
      {
           for(C=N;C<col-N;C++)
           {
                sum = 0;
                for(r1=-N;r1<=N;r1++)
                {
                     for(c1=-N;c1<=N;c1++)
                     {
                         sum = sum +image[(R+r1)*col+(C+c1)];
                     }
                }
                smooth[col*R+C]=sum/(N0*N0);
           }
      }
```

Here,
int N0 = 7 = order of the 7x7 mean filter.
int N = N0/2
int row = height of the image in pixels
int col = width of the image in pixels
unsigned char* image is a pointer to memory storing input.
unsigned char* smooth is a pointer to memory storing output.

b)Separable filters version:

For a KxK filter, separable filter convolution requires 2K operations per pixel which is lesser than 2D convolution method.

[Code snippet]:

```
    /* Convolution of 7x7 mean filter with the input image
     * excluding the borders using separable filters */

     /*1D convolution with row separable filter*/
      for(C=0;C<col;C++)
     {
          for(R=0;R<row;R++)
          {
                sum = 0;
                for(r1=0;r1<=2*N;r1++)
                {
                    /*check added to prevent invalid read*/
                     if((R+r1)*col + C < row*col)
                     sum = sum +image[(R+r1)*col + C];
                }
                inter[R*col+C]=sum;
          }
     }
     /*1D convolution with column separable filter*/
     for(R=0;R<row-(2*N);R++)
     {
           for(C=N;C<col-N;C++)
           {
                sum = 0;
                for(c1=-N;c1<=N;c1++)
                {
                    sum = sum +inter[(R*col)+(C+c1)];
                }
                smooth[col*(R+N)+C]=(unsigned char)(sum/(N0*N0));
           }
      }
```

Here,

int N0 = 7 = order of the 7x7 mean filter.

int N = N0/2

int row = height of the image in pixels

int col = width of the image in pixels

unsigned char* image is a pointer to memory storing input.

int* inter is a pointer to memory storing intermediate result.

unsigned char* smooth is a pointer to memory storing output.

c) Separable filters along with a sliding window version:

Using sliding window along with separable filters convolution further speeds up as it uses the summation from the preceding pixels.

[Code Snippet]:

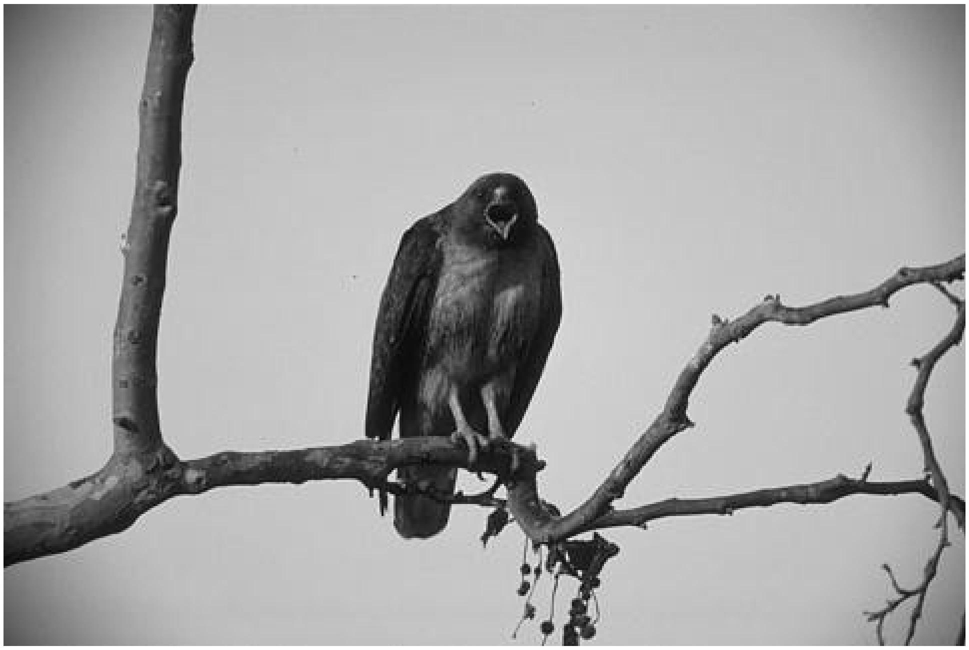
```
/*1D convolution with row separable filter and sliding window*/
     for(C=0;C<col;C++)
      {
            for(R=0;R<row;R++)
            {
                   if(0 == R)
                   {
                         sum = 0;
                         for(r1=0;r1<=2*N;r1++)
                         {
                                 sum = sum +image[(R+r1)*col + C];
                         }
                   }
                   else
                   {
                    /*check added to prevent invalid read*/
                     if(((R-1)*col+ C) < row*col && ((R+2*N)*col+C) < row*col)
                     sum = sum-image[(R-1)*col+ C+image[(R+2*N)*col+C];
                   }
                   inter[R*col+C]=sum;
            }
      }
     /*1D convolution with column separable filter and sliding window*/
      for(R=0;R<row-(2*N);R++)
      {
             for(C=0;C<col-(2*N);C++)
             {
                   if(0 == C)
                   {
                         sum = 0;
                         for(c1=0;c1<=2*N;c1++)
                         {
                               sum = sum +inter[(R*col)+c1];
                         }
                   }
                   else
                   {
                      sum = sum - inter[(R*col)+(C-1)]+inter[(R*col)+(C+2*N)];
                   }
                   smooth[col*(R+N)+(C+N)]=(unsigned char)(sum/(N0*N0));
             }
       }
```

Here,
int N0 = 7 = order of the 7x7 mean filter.
int N = N0/2
int row = height of the image in pixels
int col = width of the image in pixels
unsigned char* image is a pointer to memory storing input.
int* inter is a pointer to memory storing intermediate result.
unsigned char* smooth is a pointer to memory storing output.

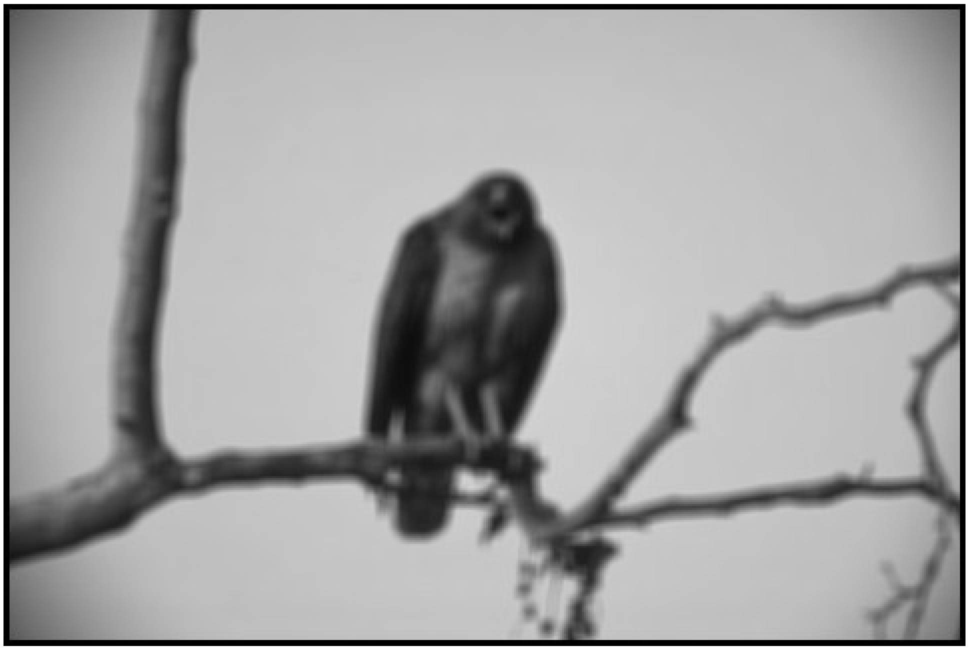
Shashi Shivaraju
XID:C88650674

Test Image and corresponding output images after convolution with the above mentioned three versions of 7x7 mean filters:

[INPUT 1]:
481 x 321 8bit Gray scale image of PPM format named hawk.ppm

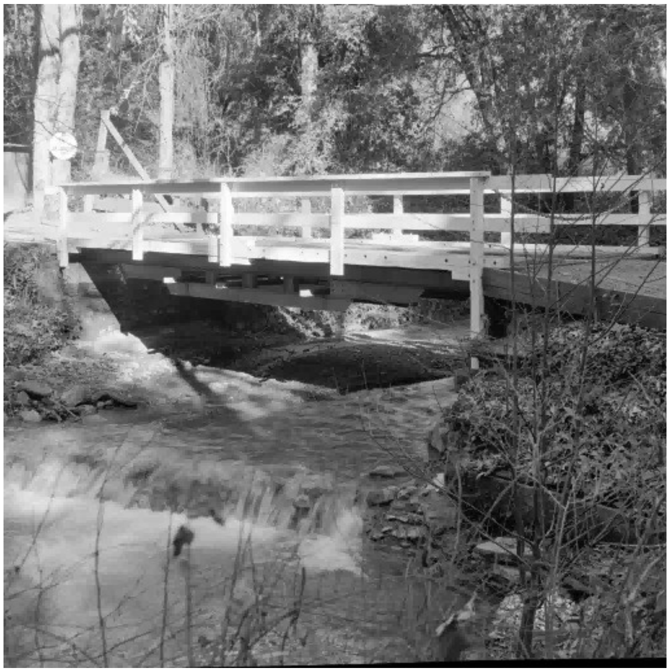


[OUTPUT 1.1]: **2D Convolution Version**
481 x 321 8bit Gray scale image of PPM format named smooth_2d_conv.ppm
[OUTPUT 1.2]: **Separable filters version**
481 x 321 8bit Gray scale image of PPM format named smooth_sep_filter_conv.ppm
[OUTPUT 1.3]: **Separable filters along with sliding window version**
481 x 321 8bit Gray scale image of PPM format named sep_filter_sliding_win_conv.ppm

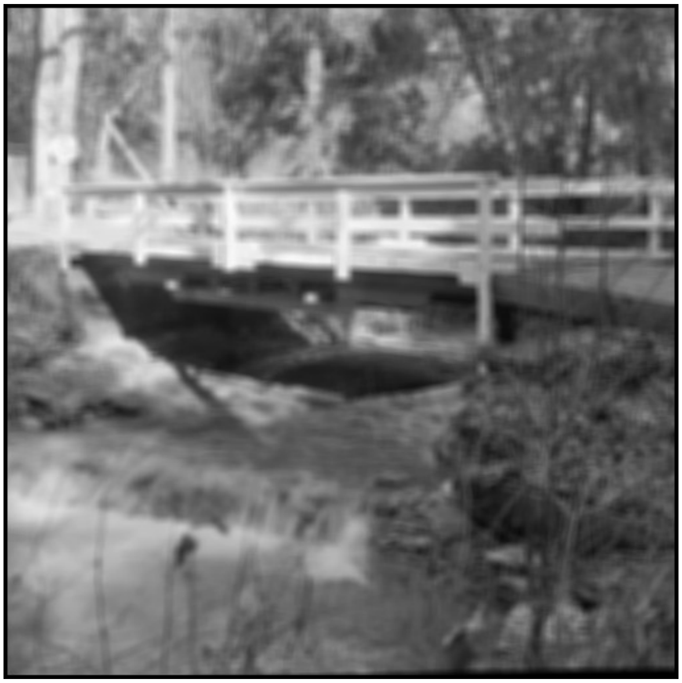


Shashi Shivaraju
XID:C88650674

[INPUT 2 *]:
512 x 512 8bit Gray scale image of PPM format named bridge.ppm



[OUTPUT 2.1*]: 2D Convolution Version
512 x 512 8bit Gray scale image of PPM format named smooth_2d_conv.ppm
[OUTPUT 2.2*]: Separable filters version
512 x 512 8bit Gray scale image of PPM format named smooth_sep_filter_conv.ppm
[OUTPUT 2.3*]: Separable filters along with sliding window version
512 x 512 8bit Gray scale image of PPM format named sep_filter_sliding_win_conv.ppm



[*NOTE:Images not as per actual scale].

Time Profiling:

Time profiling of each version of 7x7 Mean filter was performed using “clock_gettime()” API on Linux platform.

Image	**[INPUT 1]: 481 x 321 8bit Gray scale image of PPM format named hawk.ppm**			**[INPUT 2]: 512 x 512 8bit Gray scale image of PPM format named bridge.ppm**		
Filter Versions	**2D Convolution**	**Separable Filters**	**Separable Filters + Sliding Window**	**2D Convolution**	**Separable Filters**	**Separable Filters + Sliding Window**
	Time in milliseconds					
Execution 1	23.084642	8.163289	2.869328	40.119082	13.590054	5.516861
Execution 2	27.340901	7.631167	2.496004	41.154090	14.630571	5.402167
Execution 3	29.061463	8.020481	3.136193	41.122631	13.248951	5.525777
Execution 4	26.750735	8.058073	2.510615	40.943528	13.104147	5.552004
Execution 5	26.811390	7.389134	2.297801	39.842573	13.782945	5.609693
Execution 6	22.111399	7.931809	2.394722	42.161180	13.087325	5.478592
Execution 7	26.198748	8.029354	2.544971	39.969292	14.273819	5.570454
Execution 8	27.213302	7.731050	2.376942	38.885591	13.538449	5.529653
Execution 9	27.206629	7.838099	2.429441	41.070906	13.176331	5.831053
Execution 10	25.571885	7.767438	2.359502	41.055308	13.647795	5.711113
Average	**26.1351094**	**7.8559894**	**2.5415519**	**40.119082**	**13.608038**	**5.572737**

Clearly form the above table, the Separable Filters along with Sliding Window version takes the least amount of computational time among Separable Filters version and conventional 2D convolution versions of 7x7 mean filter.

Validation for Bit Exactness:

The output images obtained from the three implementations were compared using tools like **BeyondCompare (Hex Compare Method)** and also using **VBinDiff 3.0_beta4** on Linux platform.
All the three output images were identical and hence bit exact.