International Journal of Applied
and Natural Sciences (IJANS)
ISSN(P): 2319-4014; ISSN(E): 2319-4022
Vol. 5, Issue 2, Feb - Mar 2016; 35-56

International Academy of Science,
Engineering and Technology
Connecting Researchers; Nurturing Innovations

DIGITAL LOW-PASS FILTERS WITH MILDER LOW-PASS EFFECT ON DIGITAL IMAGES

ISSA.A.AL-SHAKHRAH

Professor of Medical Physics, Department of Physics, University of Jordan, Amman, Jordan

ABSTRACT

Image filtering consists of modifying the original image by logically "reimaging" it with a mathematical imaging device in which spatial response can be controlled by the user**.** Image filtering is performed by a mathematical operation called convolution, which is simply the successive replacement of each point in the original image by a new value produced by a weighted combination of the original point and its surrounding neighbour points. Filtering generally requires definition of a filtering kernel or small matrix; often a few filtering kernels are predefined in imaging computer systems. The filtering kernel is generally square with a matrix size of 3 x 3 pixels, 5 x 5 pixels or 7 x 7 pixels. Nine spatial low-pass filters (masks) are developed, then implemented and tested in our laboratory by using programs that were written in Borland c^{++} and visual FORTRAN. The results of the application of the developed low-pass digital filters (masks) on digital images (smoothing, attenuates image noise, edge enhancement and increasing the contrast (sharpening) of fine details regions), comparing between the effect of different dimensions filters (3x3 and 5x5), and milder low pass effect are presented and demonstrated. As the size of the filter (mask) gets larger, and/or the weight of the centre pixel of the kernel get higher, the effect of the mask on the input image becomes more and more.

KEYWORDS: Low-Pass Digital Filters, Milder Low-Pass Effect and Unsharp Masking Enhancement

INTRODUCTION

Image processing, with the intent of improving display information, was one of the first applications of the computer in nuclear medicine (1). Image filtering consists of modifying the original image by logically "reimaging" it with a mathematical imaging device in which spatial response can be controlled by the user. The difference between a logical and a real imaging device (e.g., scintillation camera) is that the response function of the logical device can have negative values, whereas that of the real device can only is positive. The practical result of this difference is that the logical imaging device used for image filtering can be specified to improve spatial resolution and make edge and count density transitions more obvious. Alternatively, the response can be specified so that filtering smoothes the image and reduces the noise so that small differences in count densities can be more readily perceived by the viewer (2).

More information on general image processing may be obtained from different references (3-5), and more detailed descriptions of various nuclear medicine analysis techniques may be found in Gottschalk and Erikson (6,7).

The term filtering is used to indicate operations that smooth or sharpen images; that is, increase or decrease image blurring. In planar nuclear medicine applications, however, only smoothing is frequently employed. While filtering can be described mathematically in complex terms, the actual operation is usually simple (8).

Filtering generally requires definition of a filtering kernel or small matrix; often a few filtering kernels are

predefined in imaging computer systems. The filtering kernel is generally square with a matrix size of 3 x 3 pixels, 5 x 5 pixels or 7 x 7 pixels. The numbers contained in the matrix are called weights. In a sense, the kernel is placed so that its middle pixel is on top of one pixel in the image. The kernel will then cover a 3 x 3 (or 5 x 5, etc.) pixel region of the image (8).

An image is composed of basic frequency components, ranging from low frequencies to high frequencies. Where rapid brightness transitions are prevalent, there are high spatial frequencies. Slowly changing brightness transitions represent low spatial frequencies. The highest frequencies in an image are found wherever sharp edges or points are present-like a transition from white to black within a one- or two -pixel distance.

An image can be filtered to accentuate or remove a band of spatial frequencies, such as the high frequencies or low frequencies. These digital image processing operations are known as spatial filtering operations. Other spatial filtering operations make it possible to highlight only the sharp transitions in the image, such as the edges of objects. These operations are a subset of spatial filtering operations known as edge enhancement operations.

Spatial filters are implemented through a process called spatial convolution. Spatial convolution is the method used to calculate what is going on with the pixel brightness around the pixel being processed. It is a mathematical method used in signal processing and analysis, and although the operation is mathematically complex, we can study its action in an intuitive, pictorial manner. The spatial convolution process is also referred to as a finite impulse response (FIR) filter.

The spatial convolution process uses a weighted average of the input pixel and its immediate neighbours to calculate the output pixel brightness value. The group of pixels used in the weighted average calculation is called the kernel. Kernel dimensions are generally that of a square with an odd number of mask values in each dimension. The kernel can have the dimensions of 1xl, which is the trivial case of simply a point process, 3 x3, 5 x 5, and so on. The larger the size of the kernel of pixels used in the calculation, the greater the degrees of freedom of the spatial filter. This means that the flexibility and precision of the spatial filter are increased when more neighbouring pixels are taken into account in the calculation, in practice, 3 x 3 and 5 x 5 kernels are used in most spatial filtering operations.

A weighted average calculation is called a linear process because it involves the summation of "elements" multiplied by "constant" values. The "elements" are the pixel brightness's in the kernel and the "constant" values are the weights, or convolution coefficients. In the simple case where the weights are each equal to1 /number of elements in the kernel, we have a conventional averaging process. We are left with the average brightness of the kernel's pixels. If we alter the weights, certain pixels in the kernel will have more or less influence on the overall average. In fact, the selection of these weights directly determines the spatial filtering action; such as high-pass, low-pass, or edge enhancement filtering.

The mechanics of spatial convolution are straightforward. In carrying out a 3 x 3 kernel convolution, nine convolution coefficients are defined and labelled as seen below:

a b c

d e f

g h i

This array of coefficients is called the convolution mask. Every pixel in the input image is evaluated with its eight neighbours, using this mask to produce an output pixel value. We can visualize that the mask is placed over all input pixel.

The pixel and its eight neighbours are multiplied by their respective convolution coefficients and the multiplicands are summed. The result is placed in the output image at the same centre pixel location. This process occurs pixel by pixel for each pixel in the input image. The equation for the spatial convolution process is:

$$O(x, y) = aI(x-1, y-1) + bI(x, y-1) + cI(x+1, y-1) + dI(x-1, y) + eI(x, y) + fI(x+1, y) + gI(x-1, y+1) + hI(x, y+1) + iI(x+l.y+l)$$

Where it is implied that every input pixel is processed through the equation, creating a corresponding output pixel value (9)

Convolution mask values can generally take on any numeric value. It is important, however, that when the convolution process is executed, the final resulting value be between 0 and 65535 (for a 16-bit output image). This is typically handled by clipping resulting values that are less than 0 to 0.

The aim of the present study was to construct digital spatial low-pass filters (masks), applying these filters on digital images and comparing between the efficiency of these masks on digital images (smoothing, attenuates image noise, edge enhancement and increasing the contrast (sharpening) of fine details regions), comparing between the effect of different dimensions filters 3x3 and 5x5 and milder low pass effect of these filters.

METHODS

Image filtering is performed by a mathematical operation called convolution, which is simply the successive replacement of each point in the original image by a new value produced by a weighted combination of the original point and its surrounding neighbour points. Using larger filter functions, such as 5 x 5 or 7 x 7, will cause larger portions of the original image to have an effect on the value of the new image points (10).

The counts (density) in the image pixels underneath each element of the 3 x 3 kernel are multiplied by the kernel weights and the results are all summed together. The image pixel beneath the centre of the kernel is replaced by this summation. This operation is repeated for every image pixel. Figure 1 shows the process graphically for 3x3 filter function. In this process, each image pixel is replaced by a weighted average of itself and some of its neighbours in its immediate neighbourhood.

The mask (low-pass filter) coefficients add to 1. All coefficients are positive, and must therefore be fractional. In practice, low-pass mask coefficients are generally normalized to integer values. Then a brightness divider is applied to the resulting output value to bring the result within a 0 to 255 brightness range. The common names for the low-pass filtering are mean filtering, smoothing, averaging, and box filtering.

The following spatial low-pass filters (masks) (one to nine) are developed or constructed, then implemented and tested in our laboratory by using programs that were written in Borland c^{++} and visual FORTRAN as shown in Appendices 1 and 2:

Filters 1-9

Filter 1

6	25	37	25	6
25	100	150	100	25
37	150	225	150	37
25	100	150	100	25

6	25	37	25	6

X (1/1597)

Filter 2

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

X (1/25)

Filter 3

1	1	1
1	1	1
1	1	1

X (1/9)

Filter 4

2	3	2
3	5	3
2	3	2

X (1/25)

Filter 5

1	1	1
1	2	1
1	1	1

X (1/10)

Filter 6

1	2	1
2	4	2
1	2	1

X (1/16)

Filter 7

3	6	3
6	12	6
3	6	3

X (1/48)

Filter 8

1	1	1
1	16	1
1	1	1

X (1/24)

Filter 9

1	1	1
1	20	1
1	1	1

X (1/28)

RESULTS AND DISCUSSIONS

The results (effects) of (a) the application of the developed low-pass digital filters (masks from mask1 to mask9) on digital images, either edge detection, smoothing, attenuating image noise, or unsharp mask enhancement, (b) comparing between the effect of different dimensions filters (3x3 and 5x5) and (c) milder low pass effect are indicated and demonstrated on figures 2 to 14.

Common spatial filtering operations include high-pass, low-pass, and edge enhancement filters. A high-pass filter accentuates the high-frequency details of an image and attenuates the low-frequency details. A low-pass filter has the inverse effect. Edge enhancement filters detect and enhance image edge details.

A spatial low -pass filter has the effect of passing, or leaving untouched, the low spatial frequency components of an image. High-frequency components are attenuated and are virtually absent in the output image. A common low-pass convolution mask is composed of all nine coefficients having the value of 1/9:

1	1	1
1	1	1
1	1	1

X (1/9)

This mask carries out a straight pixel brightness averaging process, as described earlier. It is often referred to as a box filter. Two aspects are immediately evident that the coefficients sum to 1 (9 x 1/9 = 1) and that they are all positive numbers. These two facts hold true for all low-pass filter masks.

The output brightness in a region of constant brightness pixels in the output is the same as the input brightness. This correlates with the fact that there is no spatial activity in the region (as evidenced by a lack of gray-level changes), which means that there is a spatial frequency of zero. Zero is the lowest possible frequency and, of course, would be expected to be passed, unchanged, by the low-pass filter, the constant brightness is passed untouched

The low-pass filter process over an image area of sharp black-to-white transitions. The bright range becomes greatly attenuated**.** As the low-pass mask moves over the pixel groups in the high-frequency region, similar middle gray values result. This produces an output image that is composed of slightly varying middle gray brightness. The high-frequency black-to-white transitions of the input image have been attenuated to transitions of minimal gray-level values. This is exactly the high frequency attenuation that we expect in a low-pass filter.

The visual effect of a low- pass filter is image burring. This is because the sharp brightness transitions that we perceive as edges become attenuated to small brightness transitions. The small brightness transitions appear to have less detail and look fuzzy or blurred as shown in figure 2.

We can see that the output image is related to the spatial frequency content of the input image- slow-changing areas are left unchanged, or are changed slightly, and fast changing areas are averaged to yield only the slow-changing aspects (Figure 3 a and b).

If we were to pass the low-pass mask over an area with a single-pixel-width line having constant background pixel brightness, we would expect the line to be blurred. This is because the line represents high spatial frequency content. In fact, a sharply defined line is actually composed of frequency components spanning a wide spectrum of low to high frequencies. As we move the mask over the line, pixels are replaced by the average of the bright line pixels and the

constant background pixels. The resulting values are somewhere between the two. The line blurs into the background. Only the low--frequency components of the line remain at the end of the entire image convolution.

The low-pass filter's blurring effect can provide for better analysis of an image's low-frequency details by removing visually disruptive high-frequency edges and patterns. By subtracting a low-pass filtered image from its original, a sharpened image can by created. This operation is known as unsharp masking enhancement, figure 4**.**

The unsharp masking enhancement operation sharpens an image by subtracting a brightness-scaled, low-pass-filtered image from its original, produces a subtle and visually pleasing result. The resulting image appears with sharpened high-frequency details; low-frequency details are left untouched (7). It is used primarily to enhance the subjective spatial appearance of an image's high-frequency details. Images suffering from poor spatial definition are best suited for the unsharp masking enhancement. Poor spatial definition can be enhanced by applying the mask more than one time on the original image (Figure 5 a, b and c).

The unsharp masking operation is implemented by first performing a low-pass filtering operation on the original image. The low – passed image is then brightness-scaled down to a desired level. The brightness-scaled image is subtracted from the original image. The resulting image contains sharpened edge detail as shown in figures 6 and 7 (11).

Low-pass filtering smoothes an image by attenuating high-spatial frequency details the convolution mask weighting coefficients are selected to vary the cut off point where higher frequencies become attenuated. Further, the resulting low-pass filtered image can be brightness-scaled down and summed with the original image to create milder low-pass filter effects (Figures 10 and 11)**.**

Low-passed image using mask 1 (all 1s), but with a centre coefficient of 16 and a gain multiplier of 1/24 the effect is to add a brightness-scaled down, low-passed image to the original image. The resulting image shows a milder low-pass effect.

The characteristics of low-frequency objects of interest can be enhanced by removing high-frequency image information using low-pass filtering. In particular, edges and sharp lines and points are smoothed, while low-frequency attributes are left untouched. For example, an image of an object containing only low spatial frequencies with a distracting high-frequency grid pattern can be enhanced in this way. A case like this is a biological specimen imaged under a microscope with measurement reticles. The low-pass effect is to blur the sharp lines of the reticles while leaving the low-frequency object relatively unaffected.

Mean filtering is usually thought of as a convolution filter. Like other convolutions it is based around a kernel, which represents the shape and size of the neighborhood to be sampled when calculating the mean. Often a 3×3 square kernel is used, as shown in Figure 1, although larger kernels (e.g. 5×5 squares) can be used for more severe smoothing. (Note that a small kernel can be applied more than once in order to produce a similar but not identical effect as a single pass with a large kernel (12, 13).

Mean filtering is most commonly used as a simple method for reducing noise in an image. Figure 12 shows the effect of applying a 3×3 mean filter. Note that the noise is less apparent, but the image has been `softened'. If we increase the size of the mean filter to 5×5, we obtain an image with less noise and less high frequency detail, as shown in figure 13.

This result is not a significant improvement in noise reduction and, furthermore, the image is now very blurred.

Large filter (filter 2) and mild low-pass effect may be better for edge enhancement as shown in figure 14.

There are two main problems with mean filtering, which are:

- A single pixel with a very unrepresentative value can significantly affect the mean value of all the pixels in its neighborhood.
- When the filter neighborhood straddles an edge, the filter will interpolate new values for pixels on the edge and so will blur that edge. This may be a problem if sharp edges are required in the output (14).

In general the mean filter acts as a low-pass frequency filter, therefore, reduces the spatial intensity derivatives present in the image (15).

Computer processing is a prerequisite for display and quantification of all modern medical images. The outcome of clinical reading and diagnosis directly depends on the technique and type of processing performed. The appearance of images can be drastically changed, which may either enhance or degrade the presentation of important information. Complicated mathematical techniques can be applied to image data to provide either accurate or inaccurate quantitative measures of physiological function. To ensure that image quality is enhanced and quantified values are correct, the user must have a competent understanding of the processing techniques (4).

As the size of the filter (mask) gets larger, and/or the weight of the centre pixel of the kernel gets higher, the smoothing effect becomes more and more (Figures 5 and 6 for 3x3 and 5x5 dimensions masks).

Adding the original image to the low-passed restored the overall gray level variations in the image, with the low-pass increasing the contrast at the locations of gray-level discontinuities. The net result is an image in which small details were enhanced and the back- ground tonality was perfectly preserved.

CONCLUSIONS AND RECOMMENDATIONS

Image differentiation enhances edges and other discontinuities (such as noise) and deemphasizes areas with slowly varying gray-level values (7).

General understanding of a processing technique includes the knowledge of what an operation really does, how it works and why it is important, and fully understanding the basis of processing procedures helps in solving problems and overcoming processing failures... The grasp of this information allows the user to adjust strategies to particular situations.

There are two main problems with mean filtering, which are:

- A single pixel with a very unrepresentative value can significantly affect the mean value of all the pixels in its neighborhood.
- When the filter neighborhood straddles an edge, the filter will interpolate new values for pixels on the edge and so will blur that edge. This may be a problem if sharp edges are required in the output.
- In general the mean filter acts as a low-pass frequency filter and, therefore, reduces the spatial intensity derivatives present in the image.

FIGURES CAPTIONS

Figure 1 (a) Raw data before filtering low values of the pixels

150	20	150	20	150	20	150
20	**150**	**20**	**150**	**20**	**150**	20
150	**20**	**150**	**20**	**150**	**20**	150
20	**150**	**20**	**150**	**20**	**150**	20
150	**20**	**150**	**20**	**150**	**20**	150
20	**150**	**20**	**150**	**20**	**150**	20
150	20	150	20	150	20	150

Input Image I(x,y)

(b) Low – pass filter

1	1	1
1	1	1
1	1	1

X (1/9)

(c) Results after filtering

54	65	63	63	63	63	80
76	**92**	**78**	**92**	**78**	**92**	78
92	**78**	**92**	**78**	**92**	**78**	92
78	**92**	**78**	**92**	**78**	**92**	78
92	**78**	**92**	**78**	**92**	**78**	92
78	**92**	**78**	**92**	**78**	**92**	76
73	57	57	57	57	57	54

Output Image O(x,y)

Figure 1: Low-Passed Spatial Filtering Operation. (a) The Original Image Matrix (b) Results after Filtering

Figure 2: (a). Original Noisy Baby Photographic Image (b). Low-Passed Filtered Image Using Mask 4

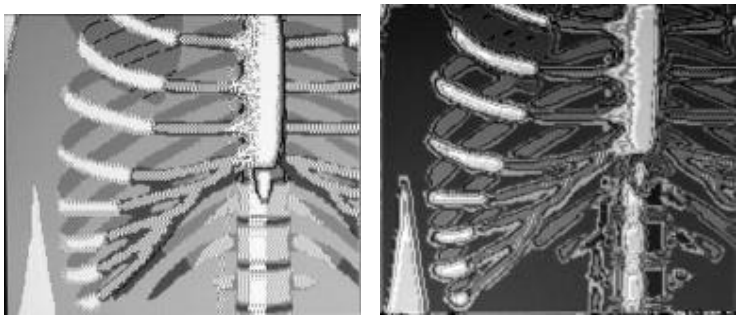

Figure 3: (a). Original Ribs Phantom Photographic Image (b). Low-Passed Filtered Image Applying Mask 2

Figure 4: (a). Original Noisy Baby Photographic Image (b) Low Passed Filtered Image Applying Mask1 (c). The Resulting Image (Subtracting b from a)

Figure 5: (a). Original Noisy Baby Photographic Image (b). Low-Passed Filtered Image Applying Mask 3 (3x3 Dimensions) Two Times. (c). The Resulting Image (Subtracting b from a)

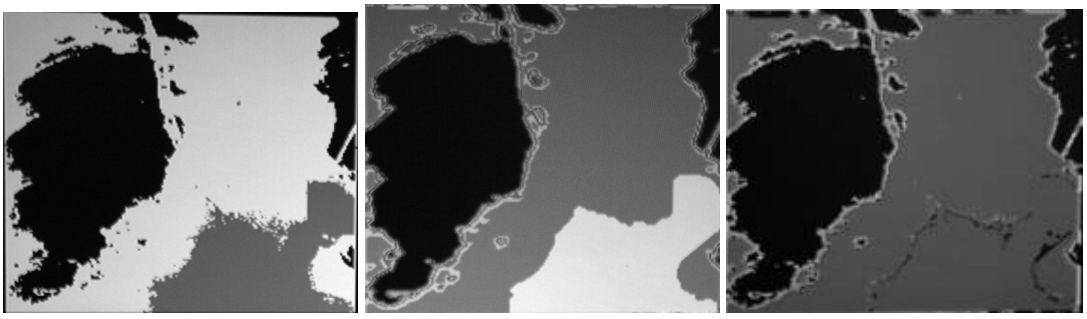

Figure 6: (a). Original Noisy Radiographic Chest x-Ray Image (b). Low-Passed Filtered Image Applying Mask 2 (5x5 Dimensions). (c). The Resulting Image (Subtracting b from a)

Figure 7: (a). Original Ribs Phantom Photographic Image (b).Low-Passed Filtered Image Using Mask 1. (c). The Resulting Image (Subtracting b from a)

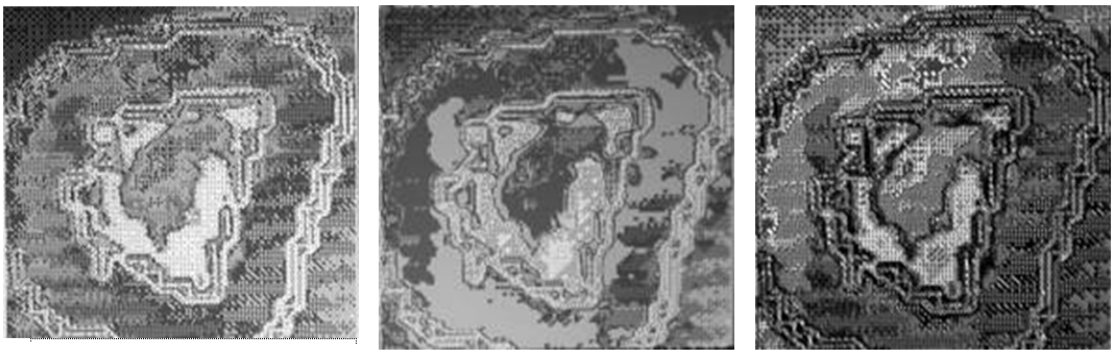

Figure 8: (a). Original Very Noisy Radio Isotopic Kidney Image (b). Low-Passed Filtered Image Applying Mask 2 (3x3 Dimensions). (c). The Resulting Image (Subtracting b from a)

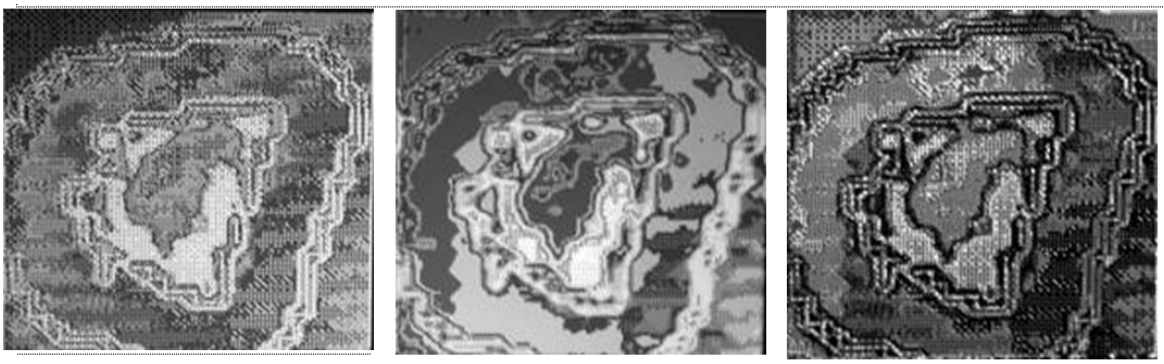

Figure 9: (a). Original Very Noisy Radio Isotopic Kidney Image (b). Low-Passed Filtered Image Applying Mask 2 (5x5 Dimensions). (c). The Resulting Image (Subtracting b from a)

Figure 10: (a). Original Noisy Baby Radiographic Image (b) Low-Passed Filter Image Applying Masks 8. (c) The Resulting Image (Adding a to b), The Resulting Image shows A Milder Low-Pass Effect

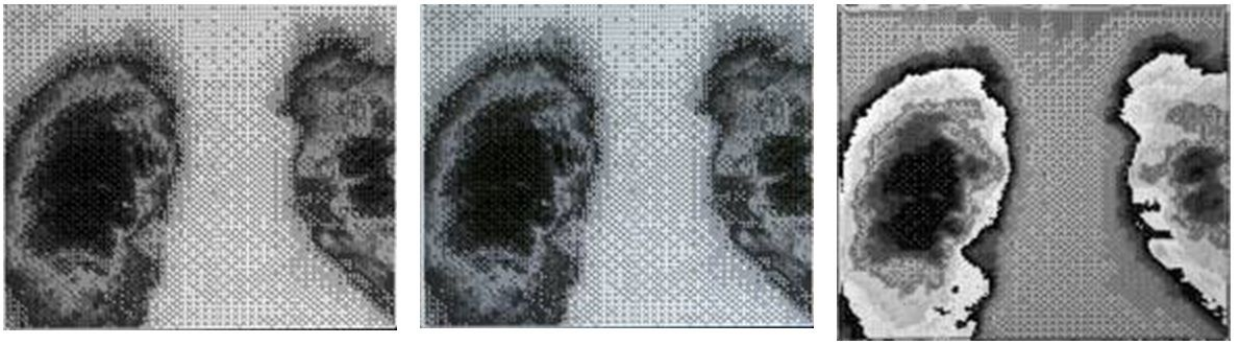

Figure 11: (a). Original Noisy Radiographic Chest x-Ray Image (b) Low-Passed Filter Image Applying Masks 8. (c) The Resulting Image (Adding a to b), The Resulting Image shows a Milder Low-Pass Effect

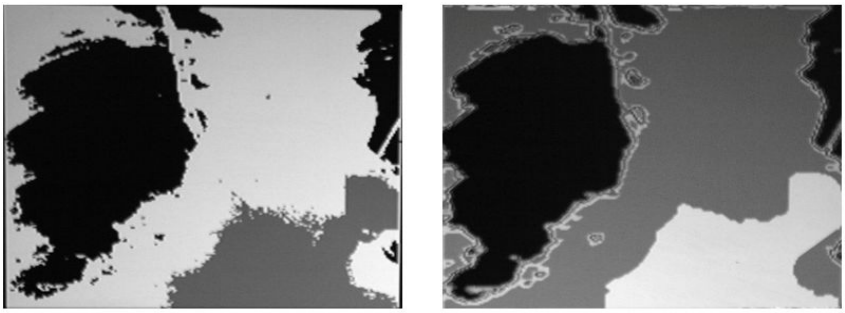

Figure 12: (a). Original Noisy Radiographic Chest x-Ray Image (b). Low-Passed Filtered Image Applying Mask 3 (3x3 Dimensions)

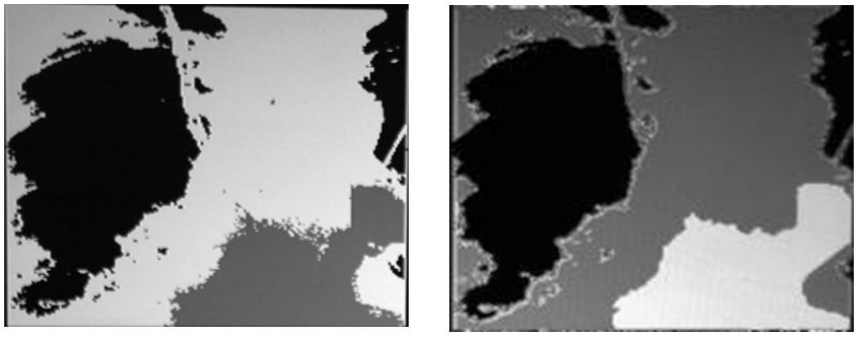

Figure 13: (a). Original Noisy Radiographic Chest x-Ray Image (b). Low-Passed Filtered Image Applying Mask 2 (5x5 Dimensions)

Figure 14: (a). Original Noisy Baby Radiographic Image (b) Low-Passed Filter Image Applying Masks 2. (c) The Resulting Image (Adding a to b), The Resulting Image Shows A Milder Low-Pass Effect

REFERENCES

1. Brown Dw, Kirch DL, Reyerson TW, et al. Computer processing of scans using Fourier and other transformations. J Nucl Med 1971; 12: 287-91.
2. John J. Erickson. Nuclear medicine computers – software. J Nuc Med Technol 1985; 13(3): 140-149.
3. Gonzalez RC, Witz PA. Digital image processing, 1st edition. London: Addison-Welsey: 1977: 1-503.
4. Jain A. Fundamentals of digital image processing. Englewood Cliffs NJ: Prentice-Hall; 1989:1-569.
5. Rafael C. Gonzalez and Richard E. Woods. Digital image processing, Second edition. Pearson Education, Prentice Hall, Upper Saddle River, NJ 07458: 2002: 1-793.
6. Gottschalk A, ed. Diagnostic nuclear medicine, second edition. Baltimore, MD: Williams and Wilkins: 1988: 1-1154.
7. Erikson JJ, Rollo FD. Digital nuclear medicine. Philadelphia: J.B. Lippincott: 1983: 1-240.
8. Tracy L. Faber and Russel D Folks. Computer processing methods for nuclear medicine images. J Nuc Med Technol 1994; 22: 145-162.
9. Gregory A. Baxes. Digital Image Processing, Part VI, Processing in Action, Image Operation Studies. John wiley & Sons, Inc., 1994.
10. Miller TR and Sampath kumaran KS. Digital filtering in nuclear medicine. J Nuc Med 1982; 23: 66-72.
11. Gregory A. Baxes. Digital Image Processing, Chaptr 4, Part II, Processing Concepts, Image Enhancement and restoration . John wiley & Sons, Inc., 1994.
12. Rosenfeld, A., and Kak, A.C. Digital Picture Processing, vols. 1 and 2 2nd ed., Academic Press, New York. 1982.
13. R. Boyle and R. Thomas Computer Vision: A First Course, Blackwell Scientific Publications, 1988, pp 32 - 34.
14. E. Davies Machine Vision: Theory, Algorithms and Practicalities, Academic Press, 1990, Chap. 3.
15. D. Vernon Machine Vision, Prentice-Hall, 1991, Chap. 4.

APPENDIX 1

```
//          This program includes the implementation and testing
//          of  all the constructed low-pass filters

   #include <graphics.h>

   #include <dos.h>

   #include <alloc.h>

   #include <stdio.h>

   #include <conio.h>
```

```
#include <stdlib.h>
#include <math.h>
#include <fstream.h>
//double sqrt  (double x)
//double fabs  (double x)
//double x;

FILE *source,*dest;
char far *buf;
char file[20];
unsigned char ch;
unsigned int  c1,c2;
int count=0;
long position=0;
unsigned size;
int jj;
const int im_size=200;

char matrix[im_size][im_size];

int i=im_size,j=0;
enum flag{T=1,F=0}flag=T;
//=================================================
// lowfreq and noiseclean are included in the same function
// edge2 and laplacian50 are included in the same function
void Smooth25(int i, int j);
void LPFA25 (int i, int j);
```

```
void LPFA(int i, int j);
void add_matrices(char a[][im_size],char b[][im_size],char c[][im_size]);
//============================
void main()
{
//   printf("Enter The file to be shwon: ");
//   gets(file);
   ofstream  matrix3;
   matrix3.open("matrix3.dat");
   int gdriver = DETECT, gmode, errorcode;
   int x=0, y=480, color, maxx, maxy,
         maxcolor, seed;
 /* initialize graphics and local variables */
  initgraph(&gdriver, &gmode, "c:\\borlandc\\bgi");

/* read result of initialization */
   errorcode = graphresult();
/* an error occurred */
   if (errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
/* terminate with an error code */
      exit(1);
   }
  if ( (source=fopen("c:\\borlandc\\bmp\\bab.bmp","rt"))== NULL)
```

```
    {
        closegraph();
     fprintf(stderr, "Cannot open input file.\n");
    }
  position=ftell(source)+118;
  fseek(source, position, SEEK_SET);
  while ( ! feof(source) )
     {
      fread(&ch,sizeof(char) , 1, source);
      count++;c1=c2=0;
      c1=int(ch) >> 4;
      ch=ch<<4;
      c2=int(ch) >> 4;
      if(flag==T)
            if(i>=0 && j<=im_size)
                  {
                  matrix[i][j]=char(c1);
                  matrix[i][++j]=char(c2);

                  if(++j>=im_size)
                        {j=0,i=i-1;flag=F;}
                  }

//    putpixel(x, y, c1);
//    putpixel(++x, y, c2);
      ++x;
      if(++x>getmaxx())
```

```
        {x=0;y--;flag=T;}
  }  //while


  // copy the original matrix in a file called matrix3.dat
  //  ----------------------------------
        for(j=0;j<=im_size;++j)
         for(i=0;i<=im_size;++i)
         matrix3<<matrix[i][j];
    matrix3.close();
  //-------------------------------------


  //        if(matrix[i][j]<0 ) matrix[i][j]=0;


  //getch();
  ifstream matrix2;
  matrix2.open("matrix3.dat");//re-open file matrix3.dat in read mode


  cleardevice();
   for(j=0;j<=im_size;++j)
         for(i=0;i<=im_size;++i)
                  putpixel(j+100, i+80, int(matrix[i][j]));
      for(i=0;i<=im_size;++i)
        for(j=0;j<=im_size;++j)
                  {
                   if(i<=0) matrix[i][j]=0;
                   if(j<=0 || j>=im_size-1) matrix[i][j]=0;
                   if(i>0 && j>0 && j<im_size-1)
```

```
//================================================

//   (i-1,j-1)  (i-1,j)   (i-1,j+1)

//    (i,j-1)    (i,j)     (i,j+1)

//    (i-1,j)   (i+1,j)    (i+1,j)

//================================================

// Smooth25 (i,j);

// LPFA25 (i,j);

// LPFA (i,j);

//============================

          if(matrix[i][j]<0 ) matrix[i][j]=0;

          if(matrix[i][j]>255 ) matrix[i][j]=255;

//                  if(matrix[i][j]>63) matrix[i][j]=63;

//                  if(matrix[i][j]<0 ) matrix[i][j]=0;


//              matrix[i][j]*=.5;

                  }

//     x=100;

//     y=0;

     for(j=0;j<=im_size;++j)

       for(i=0;i<=im_size;++i)

                putpixel(j+320, i+30, int(matrix[i][j]));

//                          putpixel(j+50, i+80, int(matrix[i][j]));


// add the original matrix with the processed one and save

//   it in the matrix (matrix)

//-------------------------------------------
```

```
        for(j=0;j<=im_size;++j)
          for(i=0;i<=im_size;++i)

          { matrix2>>ch;
            matrix[i][j]= char(int(ch)-int(matrix[i][j]));

          if(matrix[i][j]<0 ) matrix[i][j]=0;
          if(matrix[i][j]>255 ) matrix[i][j]=255;

         };
//-------------------------------------------
getch();
//cleardevice();
//============================================================
  // draw the image of the produced matrix
  //----------------------------------------
        for(j=0;j<=im_size;++j)
        for(i=0;i<=im_size;++i)
                putpixel(j+320, i+250, int(matrix[i][j]));

//

//============================================================
getch();
 matrix2.close();

 fcloseall();
```

```
 closegraph();
 exit(1);
} ;
//=========================================
//   Functions Declaration
//=========================================
  void    Smooth25 ( int i, int j)
{
       matrix[i][j]=
      (6*matrix[i-2][j-2]+25*matrix[i-2][j-1]+37*matrix[i-2][j]
      +25*matrix[i-2][j+1]+6*matrix[i-2][j+2]+25*matrix[i-1][j-2]
      +100*matrix[i-1][j-1]+150*matrix[i-1][j]+100*matrix[i-1][j+1]
      +25*matrix[i-1][j+1]+37*matrix[i][j-2]+150*matrix[i][j-1]
      +225*matrix[i][j]+150*matrix[i][j+1]+37*matrix[i][j+2]
      +25*matrix[i+1][j-2]+100*matrix[i+1][j-1]+150*matrix[i+1][j]
      +100*matrix[i+1][j+1]+25*matrix[i+1][j+2]+6*matrix[i+2][j-2]
      +25*matrix[i+2][j-1]+37*matrix[i+2][j]+25*matrix[i+2][j+1]
      +6*matrix[i+2][j+2])/1597 ;
};

//*******************************************************
//      Filter 5*5   instead of 3*3   divided by 25 instead of 9

void  LPFA25 ( int i, int j)
{
      matrix[i][j]=
      (matrix[i-2][j-2]+matrix[i-2][j-1]+matrix[i-2][j]
```

```
        +matrix[i-2][j+1]+matrix[i-2][j+2]+matrix[i-1][j-2]
        +matrix[i-1][j-1]+matrix[i-1][j]+matrix[i-1][j+1]
        +matrix[i-1][j+1]+matrix[i][j-2]+matrix[i][j-1]
        +matrix[i][j]+matrix[i][j+1]+matrix[i][j+2]
        +matrix[i+1][j-2]+matrix[i+1][j-1]+matrix[i+1][j]
        +matrix[i+1][j+1]+matrix[i+1][j+2]+matrix[i+2][j-2]
        +matrix[i+2][j-1]+matrix[i+2][j]+matrix[i+2][j+1]
        +matrix[i+2][j+2])/25 ;
};
void LPFA ( int i, int j)
{       matrix[i][j]=
//      (2*matrix[i-1][j-1]+3*matrix[i-1][j]+2*matrix[i-1][j+1]
//      +3*matrix[i][j-1]+5.0*matrix[i][j]+3*matrix[i][j+1]
//      +2*matrix[i+1][j-1]+3*matrix[i+1][j]+2*matrix[i+1][j+1])/25 ;

//      (1*matrix[i-1][j-1]+1*matrix[i-1][j]+1*matrix[i-1][j+1]
//      +1*matrix[i][j-1]+1*matrix[i][j]+1*matrix[i][j+1]
//      +1*matrix[i+1][j-1]+1*matrix[i+1][j]+1*matrix[i+1][j+1])/9;

//      (1*matrix[i-1][j-1]+1*matrix[i-1][j]+1*matrix[i-1][j+1]
//      +1*matrix[i][j-1]+2*matrix[i][j]+1*matrix[i][j+1]
//      +1*matrix[i+1][j-1]+1*matrix[i+1][j]+1*matrix[i+1][j+1])/10;


//      (1*matrix[i-1][j-1]+2*matrix[i-1][j]+1*matrix[i-1][j+1]
//      +2*matrix[i][j-1]+4*matrix[i][j]+2*matrix[i][j+1]
//      +1*matrix[i+1][j-1]+2*matrix[i+1][j]+1*matrix[i+1][j+1])/16;
```

```
//       Mild low pass filter


//      (3*matrix[i-1][j-1]+6*matrix[i-1][j]+3*matrix[i-1][j+1]
//      +6*matrix[i][j-1]+12*matrix[i][j]+6*matrix[i][j+1]
//      +3*matrix[i+1][j-1]+6*matrix[i+1][j]+3*matrix[i+1][j+1])/48;


        (matrix[i-1][j-1]+matrix[i-1][j]+matrix[i-1][j+1]
        +matrix[i][j-1]+16*matrix[i][j]+matrix[i][j+1]
        +matrix[i+1][j-1]+matrix[i+1][j]+matrix[i+1][j+1])/24;


//      (matrix[i-1][j-1]+matrix[i-1][j]+matrix[i-1][j+1]
//      +matrix[i][j-1]+20*matrix[i][j]+matrix[i][j+1]
//      +matrix[i+1][j-1]+matrix[i+1][j]+matrix[i+1][j+1])/28;


};
//
void add_matrices(char a[][im_size],char b[][im_size],char c[][im_size])
{
    for(j=0;j<=im_size;++j)
       for(i=0;i<=im_size;++i)
          c[i][j]=char(int(a[i][j])+int(b[i][j]));
 };
```

APPENDIX 2

```
C
C          Pross88.For ===== PSP input data ---- PP2 output data
C
C   THIS PROGRAM CONVOLVES ANY OPERATOR OR FILTER
```

```
C    WITH A MATRIX (I,J) WITH ANY DIMENSIONS OF I AND J.
C
C
C    The filter was tested for (7,7) matrix.
C
     DIMENSION RR(7,7),DD(7,7)
C     OPEN(8,FILE='LPT1:')
           OPEN(3,FILE='d:\Issa\psp.d',STATUS='unknown')
           OPEN(4,FILE='d:\Issa\mmm.txt',STATUS='unknown')
     READ(3,*)RR
     WRITE(4,9)RR
     WRITE(4,8)
 8   FORMAT(/)
     WRITE(*,5)RR
 9   FORMAT(7(F7.1))
 5   FORMAT(7(F6.1))
     WRITE(*,7)
c     WRITE(4,7)
 7   FORMAT(/)


C
C
     N=7
     K=7
     DO 27 I=1,N
       DO 28 J=1,K

c
     DD(I,J)=(RR(I,J)+RR(I-1,J)+RR(I+1,J)
    $       +RR(I,J-1)+RR(I,J+1)+RR(I+1,J-1)
    $       +RR(I+1,J+1)+RR(I-1,J-1)+RR(I-1,J+1))/9

 28  CONTINUE
 27  CONTINUE



c    OPEN(1,FILE='c:\Issa\PP2.O',STATUS='OLD')
           WRITE(4,9)DD
           WRITE(*,5)DD

C
     STOP
     END
```