Interactive Software Package For Image Enhancement In Sonic Infrared Nde Technology

Loai M. Ahmed-Babikir
Wayne State University,

Follow this and additional works at: http://digitalcommons.wayne.edu/oa_theses

Recommended Citation

This Open Access Thesis is brought to you for free and open access by DigitalCommons@WayneState. It has been accepted for inclusion in Wayne State University Theses by an authorized administrator of DigitalCommons@WayneState.
INTERACTIVE SOFTWARE PACKAGE FOR IMAGE ENHANCEMENT IN SONIC INFRARED NDE TECHNOLOGY

by

LOAI M AHMED-BABIKIR

THESIS

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

2010

MAJOR: COMPUTER ENGINEERING

Approved by:

____________________________________
Advisor: Date:

____________________________________

____________________________________

____________________________________

____________________________________
DEDICATION

This thesis is dedicated to my mother ‘Zeinab’ who always valued education.
ACKNOWLEDGMENTS

I knew that I wanted to take the thesis option for my master’s degree but I was at a loss on what and how to go about it. My dilemma was put to rest when I took a class with Dr. Xiaoyan who not only taught me the fundamentals of image processing but also instilled in me the love of the subject. Three classes later with her and a thesis on image processing is a testimony on what a caring professor can instil on a student. My first acknowledgment goes to her.

My colleague Lou Galbiati at Cadence Design Systems cannot be thanked enough. When I first joined Cadence, he took me under his wings and set for me an example that I tried to emulate throughout my career. His love of education inspired me to pursue my masters. His continuous encouragement when the going was rough helped me to stay the coarse. My second acknowledgment goes to him, Dr. Lou.

To my own family and the family at Wayne State I give my sincere thanks and gratitude.
# TABLE OF CONTENTS

Dedication ........................................................................................................... i  
Acknowledgments .............................................................................................. ii 
List of Figures ...................................................................................................... iv 
Chapter 1 Introduction ......................................................................................... 1  
  Traditional Techniques .................................................................................. 2  
  Interactive Technique .................................................................................... 6  
Chapter 2 Algorithm & Results ........................................................................... 7  
  2.1 Elliptical Algorithm ............................................................................... 9  
  2.2 Elliptical Results .................................................................................... 15  
  2.3 Rectangular Algorithm ........................................................................ 19  
  2.4 Rectangular Results ............................................................................. 24  
Chapter 3 Interface ............................................................................................ 28  
Chapter 4 A Custom Made Image Story ............................................................. 30  
  Traditional Filters vs Proposed Algorithm .................................................... 33  
  An Elaborate Example ................................................................................. 39  
Conclusion .......................................................................................................... 43  
References .......................................................................................................... 45  
Abstract .............................................................................................................. 46  
Autobiographical Statement .............................................................................. 47
LIST OF FIGURES

Figure 1.2a: pepper noise ................................................................. 2
Figure 1.2b: result with contraharmonic Q=1.5 ................................. 2
Figure 1.2c: salt noise ................................................................. 3
Figure 1.2d: result with contraharmonic Q=–1.5 ................................. 3
Figure 1.2e: salt & pepper noise .................................................. 3
Figure 1.2f: result with median filter ............................................. 3
Figure 1.2g: noisy finger print ...................................................... 5
Figure 1.2h: result after erosion .................................................... 5
Figure 1.2i: result after dilation ................................................... 5
Figure 1.3: An image with an additive noise .................................... 7
Figure 2: i,j coordinate system ....................................................... 9
Figure 3: x/y coordinates representing noise pixels ......................... 10
Figure 4a: noise pixel representation: BW=band width, DB=distance from border, blending factor p=DB/BW ............................................. 12
Figure 4b: showing that at the border the noise pixel has more weight on the blending algorithm. As we move towards the center, the patch pixel starts to have more weight on the blending........................................... 13
Figure 5: noise area and patch area identified ................................. 16
Figure 5a: Original image with noise area selected .......................... 17
Figure 5b: a BW of 0.0 leads to complete override of the noise area ........ 17
Figure 5c: a BW of 0.1 leads to blending at the boundary while the rest is overridden by the patch ...................................................... 17
Figure 5d: a BW of 0.2 leads to further blending away from the boundary .......... 17
Figure 5e: a BW of 0.3 leads to noticeable blending .......................... 18
Figure 5f: a BW of 0.4 ................................................................. 18
Figure 5g: a BW of 0.5, 50% of the image is blended ....................... 18
Figure 5h: a BW of 0.6 patch influence is very noticeable ................... 18
Figure 5i: a BW of 0.7 .................................................................. 19
Figure 5j: a BW of 0.8 showing a smooth blend .............................. 19
Figure 5k: a BW of 0.9 showing very smooth blend .......................... 19
Figure 5l: BW of 1.0 showing complete blend ................................. 19
Figure 6: rectangular noise area ..................................................... 20
Figure 7: noise pixel location .......................................................... 20
Figure 8: noise pixel plus BW .......................................................... 22
Figure 9: noise and patch areas identified ......................................... 24
Figure 9a: Original image with noise area selected .......................... 25
Figure 9b: a BW of 0.0 leads to complete override of the noise area ...... 25
Figure 9c: a BW of 0.1 leads to blending at the boundary while the rest is overridden by the patch ................................................................. 25
Figure 9d: a BW of 0.2 leads to further blending away from the boundary .... 25
Figure 9e: a BW of 0.3 leads to noticeable blending .......................... 26
Figure 9f: a BW of 0.4 .................................................................. 26
Figure 9g: a BW of 0.5, 50% of the image is blended ........................ 26
Figure 9h: a BW of 0.6 patch influence is very noticeable ................... 26
Figure 9i: a BW of 0.7 .................................................................. 26
Figure 9j: a BW of 0.8 showing a smooth blend .............................. 26
Figure 9k: a BW of 0.9 showing very smooth blend .......................... 26
Figure 9l: BW of 1.0 showing complete blend .................................. 26
Figure 10: blending algorithm interface .......................................................... 28
Figure 10b: blending algorithm interface Menu’s and subMenu’s ....................... 29
Figure 11: new image with just the background ................................................. 31
Figure 12: new image updated with items from the original image ...................... 32
Figures 13a & 13b: red eye image .................................................................. 39
Figure 13c: noise selection .............................................................................. 40
Figure 13d: noise and patch selections plus Histogram of patch selection .......... 41
Figure 13e: intensity adjusting of the patch image .......................................... 42
Figure 13f: patch image after adjusting intensity values ................................... 42
Figure 13g: corrected left eye ......................................................................... 43
Chapter 1

Introduction

Noise is a general word used to express an unwanted distortion to an otherwise harmonious setting. It can manifest itself in various forms that span both the audible world (what can be heard), and the visible world (what can be seen). Our eye is the door for the noise into the vision world. It is through our eyes, miraculously within a blink, that we decide if all the things within a picture harmoniously blend or not. Anything that does not blend in the image has to be extracted away from the image with the least distortion possible. The best way to have the least distortion possible is to use the image itself in replacing any extracted component. In contrast with plastic surgery, to cover exposed areas of skin it is best to use skin from other areas of the same body for either complete replacement or gradual blending. A successful and elegant surgery is one that makes it hard for our eyes to know the area of operation.

Noise in an image can be a by-product of image capture. A reflection of the flash from the surface of an image is one of the most common types of noise. Unfortunately there are cases where the use of a flash is necessary despite its noise-additive property so the workaround of not using a flash is really not an option. Hence a technique must be developed to give the user the ability to manipulate an image to remove any unwanted or additive noise like the flash. Such a technique is at least made up of three steps:

1. Identity the noise area
2. Identify the replacement area or patch area
3. Completely replace or gradually blend the noise area with the patch area
Traditional Techniques

Common techniques used to diminish the effect of noise involves the use of spatial filters such as: mean filters (arithmetic, geometric, harmonic, contraharmonic), order-statistic filters (median, max, min, midpoint) and morphological filters.¹ However, the stated techniques apply to the whole image, hence affecting non-noise areas as well. Figure 1.2a shows a circuit board image corrupted by pepper noise. Figure 1.2b shows the result after filtering the image with a 3 X 3 contraharmonic filter with Q = 1.5 where Q is the order of the filter.

Figure 1.2a pepper noise                                      Figure 1.2b result with contraharmonic Q=1.5

Figure 1.2c shows a circuit board image corrupted by salt noise. Figure 1.2d shows the result after filtering the image with a 3 X 3 contraharmonic filter with Q = -1.5.
Figure 1.2c salt noise  
Figure 1.2d result with contraharmonic $Q=-1.5$

Figure 1.2e shows the circuit board image corrupted by both salt and pepper noise. Figure 1.2f shows the result after using a median filter.

Figure 1.2e salt & pepper noise  
Figure 1.2f result with median filter
Figure 1.2g shows a noisy finger print image. The noise can be eliminated through morphological techniques: erosion and dilation. To compute the erosion of an image a nXn structure element is chosen. Using this structuring element, we consider each of the *foreground* pixels in the input image in turn. For each foreground pixel (which we will call the *input pixel*) we superimpose the structuring element on top of the input image so that the origin of the structuring element coincides with the input pixel coordinates. If for *every* pixel in the structuring element, the corresponding pixel in the image underneath is a foreground pixel, then the input pixel is left as it is. If any of the corresponding pixels in the image are background, however, the input pixel is also set to background value.

To compute the dilation of an image by this structuring element, we consider each of the *background* pixels in the input image in turn. For each background pixel (which we will call the *input pixel*) we superimpose the structuring element on top of the input image so that the origin of the structuring element coincides with the input pixel position. If *at least one* pixel in the structuring element coincides with a foreground pixel in the image underneath, then the input pixel is set to the foreground value. If all the corresponding pixels in the image are background, however, the input pixel is left at the background value.

Figure 1.2h shows how the noise was eliminated using dilation with a 3X3 structure element (consisting of only 1’s). The net effect of dilation was to eliminate the noise components in both the background and the fingerprint itself. To restore the original image, dilation is performed, however, new gaps were created as shown in Figure 1.2i.
An Interactive Technique

Spatial filters are ideal if the noise is scattered throughout the image but if the noise is localized to a specific area an interactive technique would be more appropriate. The image blending algorithms introduced in this paper, gives the user control on specifying the noise area which limits the algorithm changes to partial part of the image only. In addition to specifying the noise area, the user has control over choosing a patch
area that should be blended with the noise. How much of a blending to be done is controlled by a Blend Width parameter.

Figure 1.3 shows an image with an additive noise caused by the image capture hardware. To remove such a noise, a possible patch area is located from within the image itself. An algorithm must then be developed to function as our elegant surgeon to patch the noise area with the chosen patch area.

In certain cases, the best way to deal with a noise is to ignore its existence completely. How? By building a new image based on only the items of interest, we end up with the ideal image we wished for. An example of such a case is where the noise is present on the background of the image only with no direct effect present on the main features. In such a case, a new image background can be built from the unaffected background in the main image followed by the placement of the wanted image items into their corresponding places.
Figure 1.3, An image with an additive noise.
Chapter 2
Algorithm & Results

The noise area (surgery area) can be selected as either an elliptical or a rectangular area. The patch area must be of similar shape (elliptical or rectangular). Following our elegant surgery approach mentioned above, the two areas have to be blended as smoothly as possible over a zone defined by the user. The user selects the noise area by pressing down the mouse button and moving it over the noise area. By default, the area covered while the mouse is pressed down, will be rectangular (the user can cause the selection to be elliptical via button selection). Once the noise area is selected, the mouse button is released. The mouse cursor will now take the shape of a transparent rectangle of the same size as the noise area. The user can then drag the rectangle mouse over the desired patch area and then with one mouse click get a patch of the same size as the noise. However, the user can choose to have a variable size of the patch area by following the same steps for selecting the noise area. If the patch area selected by the user is smaller than the noise area, then the algorithm will replicate the patch area until it is equal to the noise area. On the other hand, if the patch area was bigger than the noise area, then the algorithm will just use an area equal to the noise area and discard the excess area. Since there are two shapes involved, two algorithms will be discussed; elliptical and rectangular.
2.1 Elliptical Algorithm

To cover the selected elliptical noise area, a similar elliptical area (patch area) has to be located somewhere in the image itself. Once the elliptical patch area is located, it is to be blended with the noise image according to the algorithms discussed below.

Even though the mouse draws an ellipse, there is actually a virtual (not seen) rectangle that touches the far end points of the ellipse (left, right, top and bottom). The top left corner of the rectangle is our origin (0,0) point. Any point within the ellipse is expressed in terms of the distance horizontally (i coordinate) and vertically (j coordinate) from the origin. The i,j coordinate system can be represented graphically as shown in Figure 2.

For our elliptical noise area we can redefine the location of any pixel with i,j coordinates using a new set of x/y coordinates as follows:

- the center of the ellipse is our origin (0,0) point.
- x is a horizontal coordinate, with 0 at the center of our ellipse, -1 at the left side, and +1 at the right side.
The x-coordinate of any pixel with i,j location can now be described within the ellipse as:

\[ x = \frac{(i-x_{center})}{x_{center}}; \]

where \( x_{center} \) is the i coordinate of the center of the ellipse.

\( y \) is vertical coordinate, with 0 at the center, -1 at the top, and +1 at the bottom. The y coordinate of any pixel with i,j coordinates can be described within the ellipse as:

\[ y = \frac{(j-y_{center})}{y_{center}}; \]

where \( y_{center} \) is the j coordinate of the center of the ellipse.

Figure 3 below shows the x/y scale discussed above:

![Figure 3. x/y coordinates representing noise pixels.](image)

The ellipse is defined mathematically as:

\[ \left( \frac{(i-x_{center})}{x_{center}} \right)^2 + \left( \frac{(j-y_{center})}{y_{center}} \right)^2 = 1 \]

Hence in the new coordinate system, the ellipse becomes the unit circle:

\[ x^2 + y^2 = 1. \]
The above equation is true for points lying on the edge of the ellipse. However we are more concerned with points within the ellipse (noise area) and their distance from the center of the ellipse. For any pixel within the noise area, its distance from the center can be calculated as follows:

\[ \text{distance\_from\_center} = \sqrt{x^2+y^2}; \]

Figure 4a shows the noise area (surgery area). In the white region in the center, we will simply replace the original image by the patch image. The grey region is the Blending Zone. In this region, we will replace the original image with a blend of the original image and the patch image. Beyond the outer edge of the grey region, we retain the original image.

The user defines the width of the Blending Zone by specifying the Blend Width parameter BW. BW is a number between 0 and 1. It gives the width of the Blending Zone, as a distance in the x,y coordinate system. If BW is 0.1, the Blending Zone is a narrow region near the outer edge of the surgery area. As BW increases, the Blending Zone gets wider. If BW is 1, the Blending Zone is the entire surgery area. The black dot in Figure 4a is a pixel in the Blending Zone. We calculate its distance DB from the outer boundary. Next we calculate a blending factor \( p \) for this pixel as the ratio of distance from the boundary, vs. BW:

\[ p = \frac{DB}{BW} \]

Finally, we calculate the new pixel value as a combination of the patch pixel and original pixel value:

\[ \text{New pixel} = \text{patchPixelValue} \times p + (1-p) \times \text{NoisePixelValue}; \]
Figure 4a noise pixel representation: BW=blend width, DB=distance from border, blending factor p=DB/BW

Figure 4b below shows that at the boundary, the noise pixels have more influence than the patch pixels on the blended pixel value. As we approach the end of the blend width zone, heading towards the center, the opposite is true, i.e. the patch pixel becomes more dominant. The gradual shift from the noise effect to the patch image gives an overall smoother transition exactly mimicking a skilful surgery.
Figure 4b showing that at the border the noise pixel has more weight on the blending algorithm. As we move towards the center, the patch pixel starts to have more weight on the blending.

The C code implementation of the elliptical blending algorithm is as follows:

The variables used in the algorithm are:

image1: main image

init1_cropX: i location in the main image of the top left corner of the rectangle enclosing the ellipse

init1_cropY: j location in the main image of the top left corner of the rectangle enclosing the ellipse

width: the width of the rectangle enclosing the ellipse.

height: the height of the rectangle enclosing the ellipse.

blendwidth: the distance from the ellipse boundary within which the blending is performed on the noise pixels.
i, j: coordinates of the noise pixels within the enclosing rectangle.
image5: the patch image region.
image6: the noise image region.
noisePixelValue: the gray scale pixel value of the noise point.
patchPixelValue: the gray scale pixel value of the patch point.
nNewPixelValue: the new pixel value from the blending algorithm.
pwidth: the width of the patch image
pheight: the height of the patch image

Note: pwidth and pheight are used in the case that the patch image size was smaller than the noise image. In such a case, the patch image is replicated till it is of the same size as the noise.

double xcenter = width/2;
double ycenter = height/2;
for (int j=0; j<height; j++) {
    for (int i=0; i<width; i++) {
        patchPixelValue = image5->GetPixel(i%pwidth, j%pheight);
        noisePixelValue = image6->GetPixel(i, j);
        double x = (i-xcenter)/xcenter;
        // x is horizontal coordinate, with 0 at the center, -1 at left side, and +1 at the right side.
        double y = (j-ycenter)/ycenter;
        // y is vertical coordinate, with 0 at the center, -1 at top, and +1 at the bottom.
        double distance_from_center = sqrt(x*x+y*y);
double distance_from_boundary;
if (distance_from_center > 1.0) {
    distance_from_boundary = 0.0;
} else {
    distance_from_boundary = (distance_from_center > 1.0-blendwidth) ? 1.0 - distance_from_center : 1;
}

// for a point inside the ellipse, distance_from_boundary tells how far this point is from the boundary (scale from 0 to 1).
// A point outside the ellipse is considered to be on the boundary.
if (blendwidth<0.1) {
    p=(distance_from_center > 1.0) ? 0.0 : 1.0;
} else if (blendwidth==1.0 && distance_from_center==0.0) {
    p=1;
} else {
    // larger blendwidth gives more gradual blending
    p = (distance_from_boundary>blendwidth) ? 1.0 : distance_from_boundary/blendwidth;
}

int newPixelValue = p*patchPixelValue + (1-p)*noisePixelValue;
image1->SetPixel(
    i+init1_cropX,j+init1_cropY,System::Drawing::Color::FromArgb (alpha, newPixelValue, newPixelValue, newPixelValue));
2.2 Elliptical Results

Figure 5 shows the image with the noise and patch areas identified.

The elliptical noise area is selected as shown in Figure 5a. With a blend width (BW) of 0.0, there is no blending done and the noise image is completely overridden by the patch image as can be seen in Figure 5b. Since no blending was done, the patch area stands out with its boundaries clearly visible giving results similar to a crude surgery.
As the BW is increased to 0.1, 0.2, 0.3 and 0.4 (as shown in Figure 5c, Figure 5d, Figure 5e and Figure 5f) we see that the blending is becoming more apparent as the BW is increasing. We can see that the sharp contrast at the boundary is slowly giving way to smoother transition from the image area to the blending area. However, we see that further blending is required.
With a BW of 0.5, we can hardly determine the boundaries of the ellipse in Figure 5g. At a BW level of 0.6, Figure 5h, the blending is really paying off as the only thing that now stands out is the place where no blending was done (show patch values only).

At a BW level of 0.7, 0.8, shown in Figure 5i and Figure 5j, majority of the noise area blend smoothly with the rest of the image and the noise area can no longer be detected visually.
Figure 5i, a BW of 0.7

Figure 5j, a BW of 0.8 showing a smooth blend

At a BW level of 0.9, Figure 5k, the noise area is long gone and its previous area harmoniously blend with the rest of the image. Complete blending is achieved with a BW of 1.0 as seen in Figure 5l.

Figure 5k, a BW of 0.9 showing very smooth blend
Figure 5l, a BW of 1.0 showing complete blend

2.3 Rectangular Algorithm

To cover the selected rectangular noise area, a similar rectangular area (patch area) has to be located somewhere in the image itself. Once the rectangular patch area is located, it is to be blended with the noise image according to the algorithms discussed below.
The rectangular noise area can be defined in terms of both width and height as shown in Figure 6 below:

**Figure 6, rectangular noise area**

The center of the rectangular noise area (width x height) is calculated as follows:

\[ x_{\text{center}} = \text{width}/2 \quad y_{\text{center}} = \text{height}/2 \]

A pixel location within the rectangular area can be defined from its relation relative to the width (how far to the left and how far to the right) and height (how far from the top and how far from the bottom).

A pixel with location \( i,j \) can be described in terms of the width and height as follows:

Distance left (dl) = \( i \)

Distance right (dr) = width – \( i \);  Distance top (dt) = \( j \);

Distance bottom (db) = height – \( j \);

Figure 7 shows a pixel location relative to the width and height.

**Figure 7, noise pixel location**
The user defines the width of the Blending Zone by specifying the Blend Width parameter BW. BW is a number between 0 and 1. It gives the width of the Blending Zone, as a distance from the boundary to the center of the rectangle starting from 0 at the boundary to 1.0 at the center. If BW is 0.1, the Blending Zone or surgery area is 0.1 from both the width and height of the rectangle.

The black dot in Figure 8 is a pixel in the Blending Zone. To find the distance ‘dboundary’ of the pixel from the boundary, the minimum values between dl,dr and dt,db is determined:

\[ dboundary = \min(\min(dl,dr),\min(dt,db)) \]

The distance of the center from the boundary ‘dbm2’ is calculated as the minimum between the width/2 and height/2:

\[ dbm2 = \min(\text{width},\text{height})/2.0 \]

Next we calculate a blending factor ‘p’ for this pixel as the ratio of distance from the boundary/dbm2, vs. BW:

\[ p = \frac{dboundary}{dbm2}/BW \]

However, if (dboundary/dbm2)/BW is more than 1, we take the blending factor p to be 1”.

Finally, we calculate the new pixel value as a combination of the patch pixel and original pixel value:

\[ \text{New pixel} = \text{patchPixelValue} \times p + (1-p) \times \text{NoisePixelValue}; \]
The C code implementation of the rectangular blending algorithm is as follows:

The variables used in the algorithm are:

- image1: main image
- init1_cropX: i location in the main image of the top left corner of the rectangle enclosing the ellipse
- init1_cropY: j location in the main image of the top left corner of the rectangle enclosing the ellipse
- width: the width of the noise rectangle.
- height: the height of the noise rectangle.
- blendwidth: the distance from the ellipse boundary within which the blending is performed on the noise pixels.
- i, j: coordinates of the noise pixels within the noise rectangle.
- image5: the patch image region.
- image6: the noise image region.
- noisePixelValue: the gray scale pixel value of the noise point.
patchPixelValue: the gray scale pixel value of the patch point.
newPixelValue: the new pixel value from the blending algorithm.
pwidth: the width of the patch image
pheight: the height of the patch image
dbm2: minimum value between the width and height
dboundary: minimum distance of the noise pixel to the width and height
dl: distance of the pixel from the left boundary of the rectangle
dr: distance of the pixel from the right boundary of the rectangle
dt: distance of the pixel from the top boundary of the rectangle
db: distance of the pixel from the top bottom of the rectangle

Note: pwidth and pheight are used in the case that the patch image size was smaller than the noise image. In such a case, the patch image is replicated till it is of the same size as the noise.

double dbm2=min(width, height)/2.0;
double dboundary=min(min(dl,dr),min(dt,db));
for (int j=0; j<height; j++) {
    for (int i=0; i<width; i++) {
        patchPixelValue = image5->GetPixel(i%pwidth, j%pheight);
        noisePixelValue = image6->GetPixel(i, j);
        int dl=i; int dr=width-i;
        int dt=j; int db=height-j;

        if (blendwidth<0.1) {
            p=1.0;
        } else {
            // rest of the code
        }
    }
}
p = (dboundary/dbm2>/blendwidth) ?
    1.0 :
    (dboundary/dbm2)/blendwidth;
}

int newPixelValue = p*patchPixelValue + (1-p)*noisePixelValue;
image1->SetPixel(
    i+init1_cropX,j+init1_cropY,System::Drawing::Color::FromArgb(
        alpha, newPixelValue, newPixelValue, newPixelValue));
}

2.4 Rectangular Results

Patch area                       noise

Figure 9, noise and patch areas identified
As the BW is increased to 0.1, 0.2, 0.3 and 0.4 (as shown in Figure 9c, Figure 9d, Figure 9e and Figure 9f) we see that the blending is becoming more apparent as the BW is increasing. We can see that the sharp contrast at the boundary is slowly giving way to smoother transition from the image area to the blending area. However, we see that further blending is required.
With a BW of 0.5, we can hardly determine the boundaries of the rectangle in Figure 9g.

At a BW level of 0.6, Figure 9h, the blending is really paying off as the only thing that now stands out is the place where no blending was done (show patch values only).

At a BW level of 0.7, 0.8, shown in Figure 9i and Figure 9j, majority of the noise area blend smoothly with the rest of the image and the noise area can no longer be detected visually.
At a BW level of 0.9, Figure 9k, the noise area is long gone and its previous area harmoniously blend with the rest of the image. Complete blending is achieved with a BW of 1.0 as seen in Figure 9l.
Chapter 3

Interface

The Image surgery program is written using Visual C++ windows form. The program has been integrated with programs from previous classes (ECE 6690, ECE 7690) taken with Dr. Xiaoyan. A snapshot of the program GUI is shown in Figure 10a.

Figure 10a blending algorithm interface
To remove a noise from an image, the user first loads the image:

- **File -> Open -> image**
  
The image will be loaded into the **Active Image** window.

- Select the noise area: click and keep the mouse pressed down, then drag over the noise area, by default, the selection will be *rectangular*; release the mouse.

  The noise area will be displayed in the **noise** window.

  *To make the mouse selection show ellipse rather than rectangle, click on ‘□ circular cut’ button.*

- Select the patch area: select ‘□ getPatchArea’, click and keep the mouse pressed down, then drag over the patch area; release the mouse.

  The patch area will be displayed in the **patch** window.

- To blend the noise and patch image, the BW value must be set in ‘Blend%’ numeric up/down counter in the GUI (by default it is set to 0.5).

- To perform the blending algorithm, click on ‘Operate’ button. The **Active Image** window will be automatically updated with the blended image.
Chapter 4

A Custom Made Image Story

The rectangular and elliptical algorithms enabled our surgeon to skillfully blend a noise area with its patch area. However, both algorithms confined the surgeon to work on the same body and be content with the surgery results. Imagine, giving the surgeon the ability to create an ideal new image based on his own selections from the image.

The GUI provides an interface to building a new image based on selections from an existing image. Once the original image is loaded into the Active Image window, the user can start building a new image from it. First a background for the new image is selected from the original image. To select the background image, click on ‘setBKground’, choose a background area by clicking and dragging the mouse button over it. Once the mouse button is released, a new image is created with the same dimension as the original image by duplicating the background selection. The new image is created in the background of the Active Image window. To bring the new image into the foreground, click on ‘ShowNewImage’ button. Figure 11 shows an image created from the patch selection described earlier in the elliptical and rectangular algorithms.
To restore the original image to the foreground, click on ‘Restore’ button. To copy items of interest from the original image to the new image, click on ‘○ copy’, choose an item by clicking and dragging the mouse button over it. Once the mouse button is released, the new image is updated at the same location with the new item. Figure 12 shows the new image updated with some items from the original image.

Note: You can have either have elliptical or rectangular (default) selections depending on whether the ‘□ circular cut’ button is selected or not. To facilitate fast and easy copying of items of the same size, the mouse always retains the selection size so the next item can be copied by just clicking on it.
Figure 12, new image updated with items from the original image

To clear the new image to black color, click on ‘clear’ button. To copy the original image in its entirety to the new image, click on ‘save’.
Traditional Filters vs Proposed Algorithm

In the first column below are several images where the area of interest is enclosed within a red eclipse. The second column is the result after applying some of the traditional filters (the filter type is described beneath the image), the third column is the result after using the proposed algorithm. As can be seen clearly, the proposed algorithm gave the best extraction of the area within the ellipse.
High pass d=15

Notch filter 4,2
High pass, d=4

High pass, d=9
Notch filter 4,2 followed by
Geometric Mean 7
Notch filter 4.2 followed by
Geometric Mean 7
Notch filter 4,2 followed by
Geometric Mean 7
**An Elaborate Example**

Figure 13a shows an image distorted with red eye. Red eye is caused by a reflection of the camera flash in the retina. A gray scale representation of the image is shown in Figure 13b.

![Figures 13a & 13b red eye image](image_url)

For the purpose of compare and contrast only the red eye on the left eye will be removed.

- Load the image into the **Active Image** window.

- Select the noise area: the noise area is the circular red eye region.

Since the noise is circular it is best to make the mouse selection elliptical by clicking on ‘□ circular cut’ button. Figure 13c shows the noise selection on the left eye.
The noise area will be displayed in the noise window.

- Select the patch area: select ‘getPatchArea’, click on the same noise area. The patch area will be displayed in the patch window. Now both the patch and noise windows should almost have the same image. Ideally the noise area (shown as white color in the gray image) should be blended with the dark region just outside of its circumference gradually ending with a dark point in the center (the pupil). When the ‘getPatchArea’ is selected, the Histogram window shows the color distribution of the image in the patch window, i.e. the frequency of occurrence of each color in the image. As Figure 13d shows, the color ranges from 60 – 255 on a scale from 0 – 255 where 0 is black and 255 is white.
To adjust the light colors in the patch window to a darker value we can use the intensityAdjust application under the AdvanceFeature tab. As shown in Figure 13e values in the range 140 – 255 are set to 100.
Figure 13e: Intensity adjusting of the patch image

Figure 13f: Shows the new patch image after adjusting the pixel values:

Figure 13f: Patch image after adjusting intensity values
- To blend the noise and patch image, a BW value of 0.7 is chosen. Figure 13g shows a nicely corrected left eye image that appears to be free from any abnormalities.

![Figure 13g corrected left eye](image)

**Conclusion**

To remove noise from an image, it is best to locate from within the same image a patch area that, when blended with the noise, dampens the overall influence of the noise. The dampening effect depends on the blend width specified by the user. A higher blend width (BW) value causes the image to smoothly disappear within the patch area. Using the mouse in selection of the noise and patch areas enables the user to only work with part of the image (noise area) rather than the whole image leading to greater efficiency and less distortion to the whole image. Both of the elliptical and rectangular algorithms blend the noise and patch areas together, however, one algorithm might be more suitable
over the other depending on the shape of the noise area. In certain cases, it might more suitable to build a custom made new image based on certain selections from the user.
REFERENCES


ABSTRACT

INTERACTIVE SOFTWARE PACKAGE FOR IMAGE ENHANCEMENT IN SONIC INFRARED NDE TECHNOLOGY

by

LOAI M AHMED-BABIKIR

December 2010

Advisor: Dr. Xiaoyan Han

Major: Computer Engineering

Degree: Master of Science

Noise in an image can be introduced from the image capture mechanism. It can be thought of as an area of a skin that is burned and needs skillful surgery to make it blend smoothly with the surrounding skin. The best place to get a patch skin for the surgery is to use the skin from the same body. For the eyes not to recognize any abnormality in the surgery, the patch and noise areas have to be blended as gradually as possible. A visual C++ GUI interface is developed for ease of using the mouse in selecting the noise and patch areas to be blended. How much of a blending should be done is controlled by the blend width (BW) parameter. As the BW increases, the greater is the area for blending and the better the surgery would look. Two blending algorithms, elliptical and rectangular, are used to blend the noise area and the patch area to the extent specified by the blend width.

When surgery is not an option, the user can build a new image by selecting the components needed from the old image, hence, bypassing dealing with the noise area completely.
AUTOBIOGRAPHICAL STATEMENT

Growing in a large family of ten in Sudan was full of fun but was also full of motivation for pursuing education. On the helm of sailing the ship was my mother who valued education more than anything else. With my mother backing I finished high school in the very competitive Comboni College Khartoum where I sat for my O-level exams administered by Oxford in England. Having finished high school, my education ship sailed towards an electrical engineering degree in the States which I obtained in the summer of 1996 at San Diego State University. I joined Cadence Design Systems in 1998 and had a very interesting and rewarding career working there. However, I always felt that my education was not complete without a master’s degree but really did not have the time to fulfill this dream till we moved to Michigan in 2007. I can’t believe that I am almost there in getting my long awaited degree.