An Implementation of the Median Filter and Its Effectiveness on Different Kinds of Images

Kevin Liu

Thomas Jefferson High School for Science and Technology

Computer Systems Lab 2006-2007

June 13, 2007

Abstract

This project explores digital image filtering techniques by determining the effectiveness of the median filter with different inputs. Implementations of the median filter were developed with JAVA and noise-affected images such as scenery, objects, and people were used as sample inputs to determine the kind of image the filter is most effective for. Noise is introduced manually and is representative of the different kinds of noise experienced in data transmission.

Keywords: digital image, image filter, median filter, noise reduction, noise elimination, noise introduction

Introduction

The field of image processing has wide and important uses. Any field of work that involves images or videos has uses for research in this area. The results of this project will influence how images are processed and enhanced. Applications stemming from the results of this project will be important to image and video enhancement applications because this research project provides insights on the best techniques in filtering and enhancing each kind of image.

This project explores the effects of the median filter on different types of images. The fields explored include its effects on digital images of people, scenery, and objects.

The quality of filtered images will be determined in a subjective manner. The effects of human errors and subjectivity does not detract from this study because image filtering is done for subjective and aesthetic qualities, so this method of evaluation makes sense as well.

Background

Digital image processing such as filtering was first developed in the 1960's. As computers became cheaper and faster, real-time image processing became available and its applications boomed. Digital filtering attempts to clear out noise, or useless and distracting information, in pictures. Examples of noise include missing pixels and wrong pixels. Noise is inevitable when converting analog information into a digital form. Such a conversion occurs inside a digital camera, when the camera takes the analog picture from the lens and stores it as a digital file.

To produce high quality images, for both aesthetic and additional high level processing purposes such as edge detection, noise reduction is very helpful and

often required. The median filter is one such filter.
The median filter deals with each pixel and assures it fits with the pixels around it. Therefore, it is very useful in filtering out missing or damaged pixels. It is especially effective for pictures with salt and pepper noises, which are often results of electronic noise during transmission. Because of the sheer volume of data that normally needs to be filtered, the main problem in designing the median filter is efficiency and time consumption.

This project will implement the median filter and evaluate it for its effectiveness in filtering different kinds of pictures. The results of this project may help determine the situations in which the median filter is optimal for image enhancement.

Development

One. The Interface Module

The interface module is the module that provides the interface between the user and the program. This module also serves the purpose of processing the input image by representing the image in a format that is easy to manipulate. In the case of this program, the two dimensional integer array was chosen.

The interface between the user and program is currently very simple. The user accesses the module through the command prompt window and specifies the image to be filtered in the command prompt.

The conversion to a two dimensional integer array structure is done by reading the image in its pgm format and filling in a two dimensional array during that process. The code for this process is reproduced below.

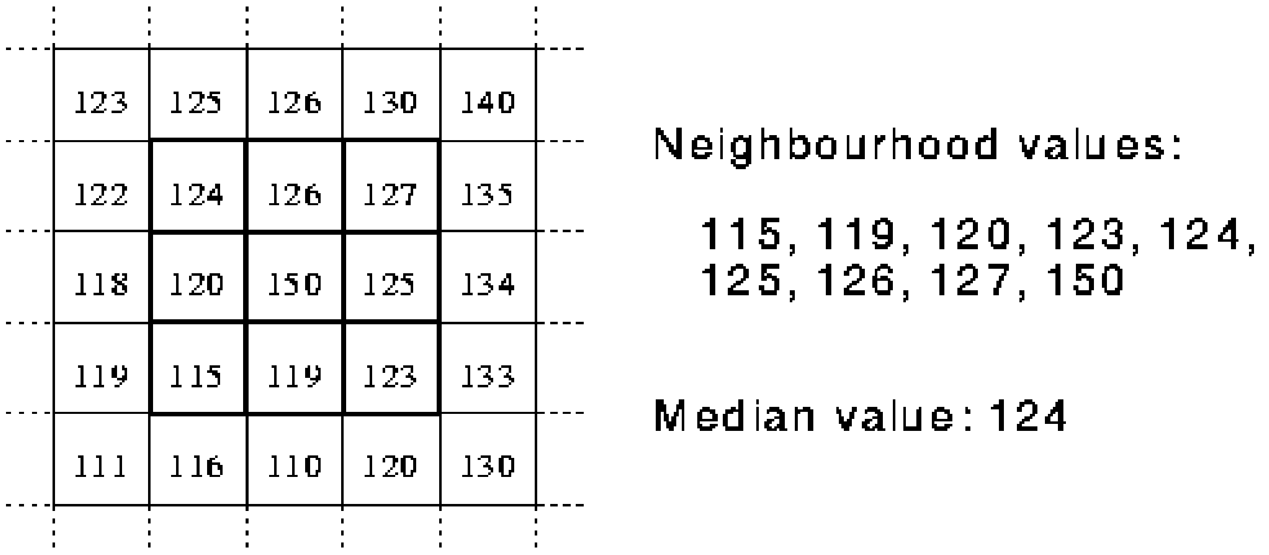
```
FileInputStream fstream = new
                              FileInputStream(args[0]);

                    // Convert our input stream to a
                    // DataInputStream
                       DataInputStream in =
                         new DataInputStream(fstream);

                    // Continue to read lines while
                    // there are still some left to read

                       String type = in.readLine();

                       String size = in.readLine();

                       size = in.readLine(); //skip the second line

                       int numCol = Integer.valueOf( size.substring(0,
size.indexOf(" ")) ).intValue();

                    int numRow = Integer.valueOf(
size.substring(size.indexOf(" ")+1) ).intValue();

                    System.out.println(numRow);
                    System.out.println(numCol);

                    String maxValue = in.readLine();

                    if( type.equals("P2") ) //if file is pgm image
                    {

                         int[][] data = new int[numRow][numCol];

                         int row;

                         for(int column = 0; column < numCol;
column++)
                         {
                              for(row = 0; row < numRow; row++)
                              {
                                   data[row][column] =
Integer.valueOf(in.readLine()).intValue();
                              }
                         }
```

After the image is in that appropriate format, it calls the median filter to filter the image or the noise introduction module .

Two. The Median Filter

The median filter module is the module that encompasses all the internal processing for the median filter. Due to its relative simplicity, the median filter may be encapsulated into one module without any problems of over-complexity or size. The median filter is a Gaussian filter that slides a window of a certain size across each pixel of the image. The size of the window in this program is three by three. At each position of the window, the nine pixels values inside that window are copied and sorted. The value of the center pixel of the window is replaced with the median value of the nine pixels in the window. The implementation of this program does not do anything with the pixels on the edges.

An illustration of the algorithm at one position of the sliding window is produced on the next page.

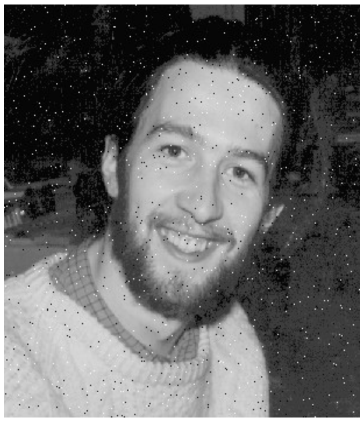
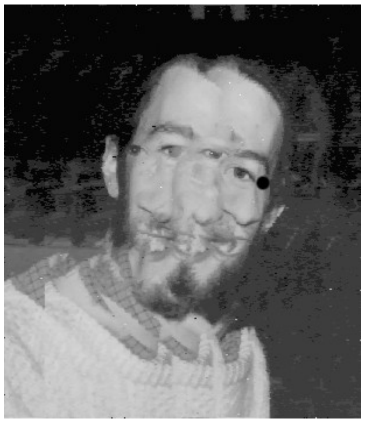


A piece of the code for this process is reproduced below.

// slides the window across the image and copies the nine pixels inside into an array called "surround"

```
          for(row = 0; row < numRow; row++)
          {
               for(column = 0; column < numCol; column++)
               {
                    if(row == 0 || row == numRow-1 || column == 0 ||
column == numCol-1)
                    {
                         result[row][column] = data[row][column];
                         continue;
                    }
                    iterator = 0;
                    for(r=row-1; r < row+2; r++)
                         for(c=column-1; c < column+2; c++)
                         {
                              surround[iterator] = data[r][c];
                              iterator++;
                         }
                    result[row][column] =
insertionSortMedian(surround, 9);      //calls the sorting method
and enters the median value

               }
          }
```

During the course of developing the median filter, a problem was encountered in which the filter eliminates the the salt and pepper noise as intended, but the pixels were shifted as well.

The output produced is placed next to the original. As obviously seen, even though the filter eliminated the noise, it shifted pixels and destroyed the original image.

The problem was found to be caused by the nomenclature in the pgm format. This format lists the number of columns first and the number of rows second as opposed to the typical convention of row first column second. This caused the matrix representation of the images to be flipped.

Three. Noise Introduction

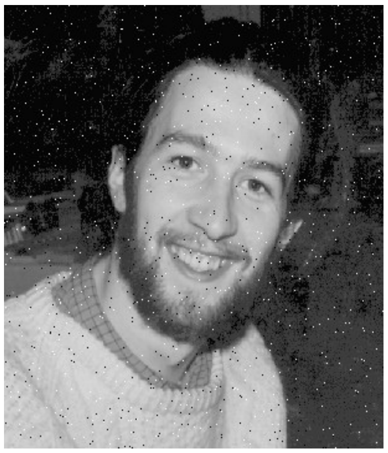
A noise introduction module was written to introduce salt and pepper noise into images to facilitate testing. The amount of noise may be determined by the user as a percentage of probability. The module goes through each pixel of the input image and at each pixel there is a chance that the value of the pixel will be changed to a number close to 0 (white), or 256 (black). This creates the salt and pepper noise representative of corrupted images.

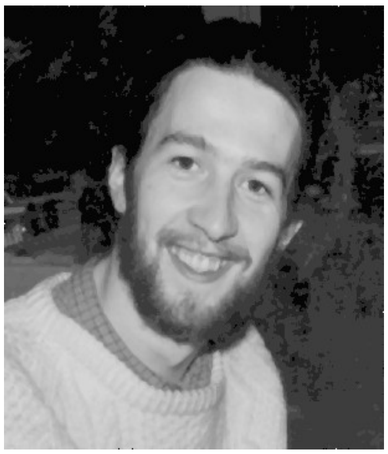
A piece of code from this module is reproduced below:

```
generator is a random number generator
for(int row = 0; row < numRow; row++)

         {

            for(column = 0; column < numCol; column++)

            {

               generated = generator.nextInt(100);

               if(generated <= probability)
               {

                  if(generator.nextInt(2) == 0)

                     result[row][column] = generator.nextInt(20);

                  else

                     result[row][column] = 236 +
generator.nextInt(20);

               }

               else

                  result[row][column] = data[row][column];

            }

         }
```

Results

Sample before and after images illustrating noise reduction:





Sample before and after images illustrating boundary blurring:

Conclusions

The noise reduction was found to be equally effective with corrupted images of people, objects, and scenery.

The edge blurring effects, however, were found to be the most severe with pictures of scenery. Images of people are second in the severity of edge blurring, followed lastly by objects. This may be because the edges of objects have distinct boundaries that do not blur easily, while images of people and scenery have soft and indistinct boundaries that are blurred very easily.

Bibliography

Davies, E. R. (2005). Shifts Introduced by Median Filters. In E. R. Davies (Ed.), *Machine Vision* (3rd ed., pp. 66 - 78). San Francisco, CA: Morgan Kaufmann Publishers.

The median filter is used more often than its counterparts, including the mean and mode filter because the median filter suppresses noise without blurring the image. The median filter, however, may shift the edges of objects in certain situations. Straight edges are not shifted as a result of median filtering, but curved edges are shifted inward (toward the center of the curve) and bumps tend to be smoothed out as a result. The worst case scenario that produces the maximum amount of filtering shift is a circular object.

The median filter may also remove smalls details that the filter mistakes for noise. A line one pixel wide, for example, would be completely wiped out by a one-by-one median filter.

Fourier Transforms & the Frequency Domain. (n.d.). Retrieved January 11, 2007, from University of California at Berkeley Web site: http://grus.berkeley.edu/~jrg/ngst/fft/concepts.html

Signals can be expressed in two different domains, spatial and frequency domains. Considering a graph of a signal in either domain, the horizontal variable in the spatial domain is time; and the vertical variable is amplitude. For a graph in the frequency domain, the horizontal variable is frequency; and the vertical variable is amplitude. Signals are expressed in either of these domains to make processing and calculations easier. To convert a representation from the spatial to the frequency domain or vice versa, we use the Fourier Transformation. The

Fourier Transformation has applications in communications, astrology, geology, and optics. The Fourier Transformation is needed for my project because the frequency filter works on images on the frequency domain, therefore incoming images must be converted from its usual spatial domain representation to the frequency domain.