K V RAMPRASAD, International Journal of Computers Electrical and Advanced Communication Engineering [IJCEACE]™
Volume 1, Issue 06, PP: 194 – 197, NOV' 2014.

APPLICATION OF MEDIAN FILTER ON TRUE COLOR IMAGES

K V RAMPRASAD 1*

1. Professor, Dept of ECE, KALLAM HARANADHA REDDY INSTITUTE OF TECHNOLOGY, GUNTUR, AP, INDIA.
Email: kvrp1976@gmail.com

ABSTARCT:

Median filters are commonly used when the objective is to achieve noise reduction with a minimum amount of blurring. A median filter replaces the pixel at the center of a mask with the median of the set of pixels under the mask. Median filters are in the class of order filters. These are nonlinear filters. In this paper, the effect of a median filter on a noisy step edge is compared with the effect of a smoothing filter. It is observed that, the smoothing filter tends to distort the edge transition[1].

Median filters are most effective against impulse (salt & pepper) noise. The smoothing filters tend to smear and lower the pulses. The median filter with N=3 is ineffective on doublets. The median filter with N=5 removes doublets.

The main objective of this paper is to remove the noise present in 24 bit true color images using median filters.

Keywords: **DATA, COMPRESSION, MEDIAN FILTERS**

I. INTRODUCTION

A digital image is a collection of light information emanating from an object in terms of discrete integer numbers that we call as pixels (picture elements). For an image to be formed or acquired, a source of light (electromagnetic radiation) should be incident on the object of interest. Any object, of whatever nature, is capable of absorbing a certain amount of electromagnetic radiation. The part of the radiation that is absorbed is purely dependant on the object. The visible band of the electromagnetic radiation spans the range: red to violet (400–700 nm approx). The natural source of light viz. sunlight consists of several frequencies within this band. We normally describe the frequency gamut of sunlight with a finite set of colors such as: violet, indigo, blue, green, yellow, orange and red. This division is necessary arbitrary and not accurate. In practice, there are numerous frequencies present within the visible band. For a qualitative description, however, this classification will suffice. The colors that we see in an image actually come from the incident light[2,3].

Consider an example that sunlight as the incident source and the imaged object as a patch of green vegetation. Green leaves have the tendency to absorb all the incident except the green component. What is actually reflected is green light, so the leaves appear green. This principle is true for any kind of object. An object surface that totally absorbs all the incident radiation will appear black, since nothing is reflected back. An object will appear white if all the incident radiation is reflected with 0% absorption. The fact that the colors that we perceive in an image actually come from the incident light can be easily verified by a simple experiment. Provide a monochromatic source of light, such as sodium vapor lamp, in a closed room and observe the colors of objects within the room. All the objects will appear either as black or yellow! Objects that appear black absorb the incident radiation (single frequency present in the sodium light) and objects that appear yellow reflect the incident light without absorption[5].

II. RELATED WORK

Image data is usually stored in a permanent magnetic medium such as computer disk files, magnetic tapes, Compact Discs etc. In order that the image data may be used by the image processing community in a standard way without any ambiguity, the image data is usually stored in some standard format. There are several standards in existence. We will discuss the following industry standard formats for image files:

- Bitmap files storing raw image data (compatible with Windows & OS2)
- JPEG files storing compressed image data (compatible with Windows & OS2)

- TIFF files storing compressed or uncompressed data (compatible with Windows, Unix and various other operating systems)

These file formats store the geometry and other details of the image in a reserved area of the file called **Header.** The most important details are the following:

- Width of the image
- Height of the image
- Horizontal resolution
- Vertical resolution
- Color depth of the image (bits per pixel)
- Color lookup table (if present)
- List of colors that need that special treatment

Some file formats like **TIFF** allow storing of multiple images in a single file. Additionally, images can also be stored in the form of strips (sections of image). Digital cameras and scanners give the user the option of storing the image data in any of these formats.

In order to process image data, the raw data has to be retrieved from these files. In order to retrieve the image data, the structure and format of these files should be known. Since these are industry standard formats, documentation pertaining the file structures are available in the public domain[6,7].

In our image processing project, we will be mainly concerned with data stored in the **BMP** format, since raw image data is easily retrievable. The following paragraphs describe the organization of **BMP** files:

Fig. 1: BMP structure Header

The header consists of the following fields. Note that, the following assumptions are made, short int of 2 bytes, int of 4 bytes, and long int of 8 bytes. Further, it is also assumed that byte ordering as for typical (Intel) machines. The header is 14 bytes in length.

The useful fields in this structure are the type field (should be 'BM') which is a simple check that this is likely to be a legitimate BMP file, and the offset field which gives the number of bytes before the actual pixel data (this is relative to the start of the file). Note that this struct is not a multiple of 4 bytes for those machines/compilers that might assume this, these machines will generally pad this struct by 2 bytes to 16 which will unalign the future fread() calls - be warned.

Information

The image info data that follows is 40 bytes in length, it is described in the struct given below. The fields of most interest below are the image width and height, the number of bits per pixel (should be 1, 4, 8 or 24), the number of planes (assumed to be 1 here), and the compression type (assumed to be 0 here).

The compression types supported by BMP are listed below :

- 0 - no compression
- 1 - 8 bit run length encoding
- 2 - 4 bit run length encoding
- 3 - RGB bitmap with mask

Only type 0 (no compression will be discussed here.

24 bit Image Data

The simplest data to read is 24 bit true color images. In this case the image data follows immediately after the information header, that is, there is no color palette. It consists of three bytes per pixel in b,g,r order. Each byte gives the saturation for that color component, 0 for black and 1 for white (fully saturated).

Indexed Colour Data

If the image is indexed color then immediately after the information header there will be a table of infoheader.ncolors colors, each of 4 bytes. The first three bytes correspond to b,g,r components, the last byte is reserved/unused but could obviously represent the alpha channel. For 8 bit grayscale images this color index will generally just be a grayscale ramp. If

you do the sums....then the length of the header plus the length of the information block plus 4 times the number of palette colors should equal the image data offset. In other words
14 + 40 + 4 * infoheader.ncolors = header.offset

The .bmp file is the abbreviation for bitmap image file format. The images are stored in binary format. Before storing the images, scanning Gallery plus adds "Header" to the image file, which consists of various types of information. The header consists of the size of image, rosolution(vertical and horizontal) etc.

1) Byte no. 0 and 1 contains characters 'B' and 'M'
2) Byte no. 2 to 5(4 bytes) specifies the size of the file in bytes
3) Byte no. 6 to 7 contains zeros
4) Byte no. 8 to 9 contains zeros
5) Byte no. 10 to 13 specifies the offset from the beginning of the file to the bitmap data
6) Byte no. 14 to 17 specifies the size of the header
7) Byte no. 18 to 21 specifies the width of the image in pixels
8) Byte no. 22 to 25 specifies the height of the image in pixels
9) Byte no. 26 and 27 specifies the no. of planes
10) Byte no. 28 and 29 specifies the bits per pixel
11) Byte no. 30 to 33 specifies the type of compression
12) Byte no. 34 to 37 specifies the size of image in bytes
13) Byte no. 38 to 41 specifies the horizontal pixels per meter on the designated target device.
14) Byte no. 42 to 45 specifies the vertical pixels per meter on the designated target device.
15) Byte no. 46 to 49 specifies the no. of colors used in the bitmap
16) Byte no. 50 to 53 specifies the no. of colors that are important for the bitmap

Implementation of Median Filters

In median filtering, the input pixel is replaced by the median of the pixels contained in a window around the pixel, that is,

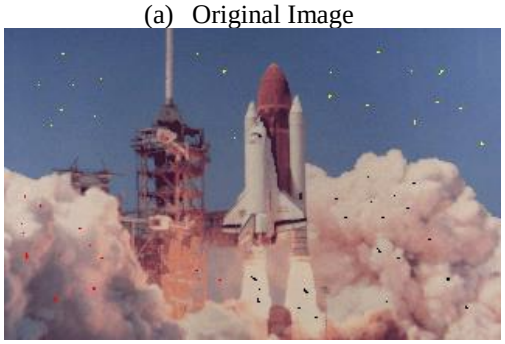
$$V(m,n) = \text{median}\{ m-k, n-l), (k,l) \in W$$

where W is a suitably chosen window. The algorithm for median filtering requires arranging the pixel values in the window in increasing or decreasing order and picking the middle value. Generally, the window size is chosen so that Nw is odd. If Nw is even, then the median is taken as the average of the two values in the middle[8].

III. RESULTS & DISCUSSIONS

When the images are processed, the noise in the images is eliminated using median filters and the results are shown below.

(a) Original Image

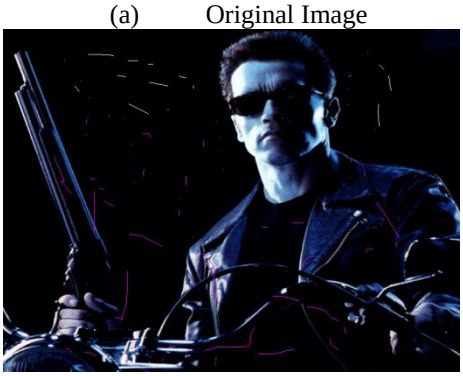


(b) Corrupted with noise

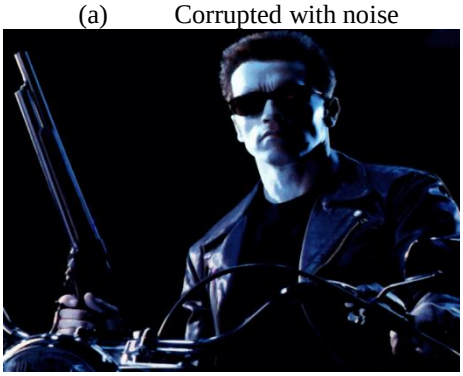
(c) Smoothened with Median Filter

Fig. 2: Processing Results of shuttele.bmp

(a) Original Image



(a) Corrupted with noise



(b) Smoothened with Median Filter

Fig. 3: Processing Results of arnie.bmp

IV. CONCLUSIONS

When the noisy images are processed using averaging or any other linear filters to remove the impulse noise, some times they tend to blur the edges, which are nothing but the sharp transitions in the image, but it is observed that such problem can be minimized using the application of median filters.

V. REFERENCES

[1] Digital Image Processing by Rafael C. González, Richard Eugene Woods

[2]Digital Image processing using MATLAB by Rafael C. Gonzalez, Richard Eugene Woods, Steven L. Eddins

[3]Digital image processing, Volume 1 by Bernd Jähne

[4] D. L. Donoho, M. Vetterli, R. A. DeVore, and I. Daubechies, "Data compression and harmonic analysis," IEEE Trans. Inform. Th., vol. 44,no. 6, pp. 2435–2476, October 1998

[5] S. Mallat, A Wavelet Tour of Signal Processing, 2nd ed. Academic Press, 1999.

[6] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," IEEE Signal Processing Magazine, vol. 18, pp. 36–58, Sep. 2001.

[7] R. H. Bamberger and M. J. T. Smith, "A filter bank for the directional decomposition of images: Theory and design," IEEE Trans. Signal Proc., vol. 40, no. 4, pp. 882–893, April 1992.

[8] P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact image code," IEEE Trans. Commun., vol. 31, no. 4, pp. 532–540, April 1983.