ISSN: 2277-9655
(ISRA), Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Implementation of Parallel Algorithms for Image Enhancement Using Matlab

Shaimaa Ibrahem [*], I.A.Ismail [1], Magdy Ahmed [2], Ahmed A. El-shami [2]

[*] Computer science, NCA academy, Cairo, [1] October 6 University, Giza, [2] Suez Canal University, Ismailia, Egypt

en_sh1_cs@yahoo.com

Abstract

This paper presents an efficient implementation of algorithms which are used for image enhancement process, which filter and restore images of big size easy and faster. Application with sequential algorithm can no longer work to improve the program performance. In the image enhancement technique we need to work with large image data which takes a lot of time. Parallel computing is an efficient way to handle large size images and to reduce the processing time. This paper focuses on calculating the parallel execution time spend in parallel filtering program and compares it with corresponding sequential execution time, moreover we discuss the results of filtering under different filter types.

Key words: Image enhancements, parallel algorithms, Fast Fourier transform.

Introduction

Parallel programs consist of a collection of tasks that are executed by processes or threads on multiple processors [1].Image enhancement is one of the most interesting and visually appealing areas of image processing [2].Image enhancement is to process an image so that the result is much better had we used the original image altogether for specific application. Enhancement process can be implementing on images with large size in parallel to improve efficiency and response time by speed up computations. Our objectives in this paper are as following. 1) We assume that a group of images is corrupted by different types of noise. 2) We use lowpass and highpath filters in parallel way to enhance the corrupted images and estimate an image that is near as long as to the original image. 3) We set a comparison between execution time process in the proposed parallel algorithms in this work and the corresponding execution time in sequential algorithms.4) We discuss the filtered images under different filter types.

This approach is tested over grayscale images which can be easily extended to colored images.

Image enhancement approaches fall into two broad categories: spatial domain methods and frequency domain methods [2].In this approach we are filtering images in frequency domain, its techniques are based on modifying the Fourier transform of an image, then after implementing the filter we compute the inverse fast Fourier transform to the filtered image. In the image enhancement we can implement the filter to an image more than one time to obtain the best enhancement image.

In Parallel programming we divide the program into multiple tasks that work individually, its tasks are assigned to threads (scheduling operation) and then the threads assigned to physical computation units for execution (mapping operation)[2].Parallel programming is commonly easy to use today in many programming language, in this approach we use Matlab program. Matlab program provides The 'parallel computing toolbox' which creates multiple workers (called 'labs' in Matlab) on the local machine, and 'parfor' (Parallel For-loops) a simple way to making FOR loops run in parallel [3].We must use 'matlabpool' to run our programs in parallel and we can upgrade the 'local' configuration so as to run on 4 or 8 or larger threads.

Creating noisy image

We suppose here that the image is corrupted by four different kinds of noise

1-**Gaussian noise**

Its model is used frequently in practice. The PDF (probability density function) of Gaussian random variables z, is given by

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\mu)^2/2\sigma^2}$$

Where z represents gray level, μ is the mean or average value of z, and σ is its standard deviation, $\sigma^2$is the variance of z [2], here we using μ= 0.0 & σ^2 = 0.1 to noise Chest ribs image fig1 ,the result show in fig 2.

Fig 1: original image

Fig 2: noisy image

2- Impulse (Salt-and-pepper) noise

Salt and pepper noise is a type of noise resemble. Its granules randomly distributed over the image, it can be negative or positive. Negative impulses appear as black (pepper) points in an image; positive impulses appear white (salt) noise [2]. The PDF of impulse noise is given by

$$p(z) = \begin{cases} p_a & for\ z=a \\ p_b & for\ z=b \\ 0 & otherwise \end{cases}$$

Where a and b are values in the digitized image, for making noise on Backbone image fig3, we using d=0.05 the result show in fig 4.

Fig 3:original image

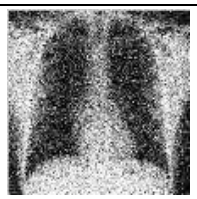


Fig 4:noisy image

3-periodic (sinusoidal) noise

Discrete sinusoidal noise is a model of periodic noise that appears on Image from electrical and/or electromechanical interference during image acquisition. We obtain to the 2-D noise from equation:

$$r(x,y) = A\sin[\,2\pi u_0(x + B_x)/\,M + 2\pi v_0(y + B_y)/\,N]$$

For x = 0,1,2,…,M - 1 and y = 0,1,2,…,N – 1, where A is the amplitude, u_0 and v_0 determine the sinusoidal frequencies with respect to the x- and y-axis, respectively, and B_x and B_y are phase displacements with respect to the origin [4].Fig 5 showing the sinusoidal noise image we generated by using an arbitrary number of impulse locations, fig 6 Chest ribs image and fig7 show the chest image after adding the noise .

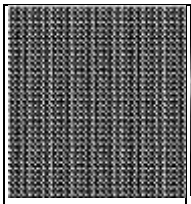


fig5 :noise

Fig 6: Original image

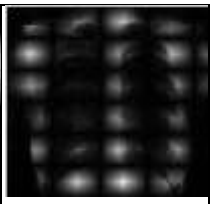


Fig 7: noisy image

4-random noise

Random noise can be generated by using random function to generate matrix with random numbers and add to the original image. Fig8 shows skull image and fig9 shows the result of noisy

Fig 8: Original image

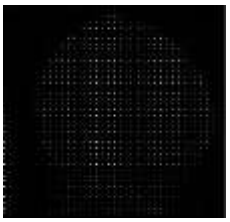


Fig 9: Noisy image

Fourier transform and Fast Fourier transform to an image

Fourier Transform is the standard algorithm for transforming data from spatial to frequency domain. When dealing discrete data (like sampled digital data), then we use the Discrete Fourier Transform (DFT):

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}dx\,dy$$

This formula involves $O(N^2)$ operations. Fortunately, a faster algorithm was invented, called Fast Fourier Transform (FFT) that performs the calculation in $O(N\cdot lg_2N)$, or better, operations. It takes advantage of recurring terms in the DFT and avoids the re-calculations.

A two dimensional transform is performed in two steps - first transform each row using a regular FFT, then transform each column with a regular FFT.

Implementation filtering noisy image using parallel method

Steps for filtering in the frequency domain

1. Reading noisy image from its source.
2. Compute F(u,v), DFT of the image from (1) using FFT2(f) function in matlab where f is the input noisy image.
3. Generate a filter function of size suitable to the noisy image(H(u,v)) .
4. Multiply the transform F(u,v) by a filter function H(u,v) , Array multiplication (outer product) .
5. Obtain the IFFT2 (inverse fast Fourier transform) to the result from step 4.

1- Filtering using lowpass filter

Lowpass filter is a type of filters that adopts smoothing effect on the images. It removes the small details from the image prior to image extraction, and bridging of small gaps in lines or curves [2].

Here we use the **Gaussian lowpass filter** (GLPFs) to filtering the noisy image, the form of this filter is given by:

$$H(u,v) = e^{-D^2(u,v)/2D_0^2}$$

Where $D(u,v)$ is the Euclidean distance from the original of the Fourier transform, which has been shifted to the center of the frequency rectangle using the procedure:

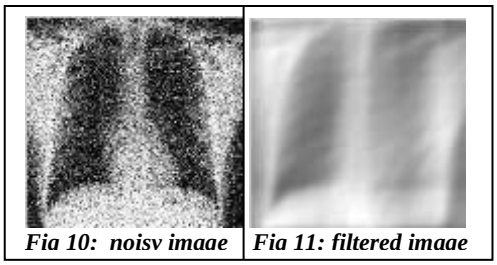
$$D(u,v) = [(u - M/2)^2 + (v - N/2)^2]^{1/2}$$

And D◦ the cutoff frequency in our example we choose D◦ = D(u,v) = 100 to be changed at will and the size of the noisy image is 200-by-200.

We implement the previous algorithm on an image corrupted by different types of noise. Using in our implementation matlab program with 8 threads in the filtering operation in parallel way and the speed of computer processor equal to 2.00 GH, Take in consideration; if we increase the threads number we obtain a minimum number of execution time.

I. Filtering Gaussian noisy image:

We are filtering Chest ribs image has corrupted by Gaussian noise show in fig 10, and D◦ = D(u,v) = 100 in the Gaussian lowpass filter. The result of filtering is shows in fig 11.



Fia 10: noisv imaae *Fia 11: filtered imaae*

Comparing between sequential execution time & parallel execution time:

The filtering time in the sequential program is 5.3 ms and the filtering time in the parallel program is 0.1357 ms So we now verify that the filtering time in parallel way is faster than the sequential way about 8 times.

Computing standard deviation:

Standard deviation (σ) between the original image and the image we get after filtering is 0.2167.

II. Filtering salt & pepper noisy image:

Next example we filtering Chest ribs image noising by salt and pepper noise (fig 12), we are using Gaussian lowpass filter two times with D◦ = D(u,v) = 150 and the result of the filter is shows in fig 13:

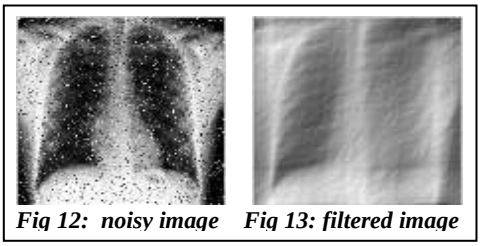


Fig 12: noisy image *Fig 13: filtered image*

Comparing between sequential execution time & parallel execution time:

The filtering time in the sequential program is 525.7 ms for two time filtering and the filtering time in the parallel program is 0.38077 ms for two time filtering.

Computing standard deviation:

Standard deviation (σ) between the original image and the image we get after filtering is 0.1868

III. Filtering sinusoidal noisy image:

In this example we using Chest ribs image noising by additive periodic noise (fig 14) and Gaussian lowpass filter with D◦ = D(u,v) = 150.Result is shows in fig 15:

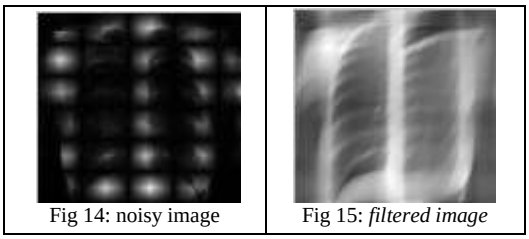


Fig 14: noisy image Fig 15: *filtered image*

Comparing between sequential time & parallel time :

The filtering time in the sequential program is 2.1 ms and filtering time in the parallel program is 0.16091ms

Computing standard deviation:

Standard deviation (σ) between the original image and the image we get after filtering is 0.3180

IV. Filtering random noisy image:

Here we use image noising by random noise (fig 16) and Gaussian lowpass filter with D◦ = D(u,v) = 150 result shows in fig 17.

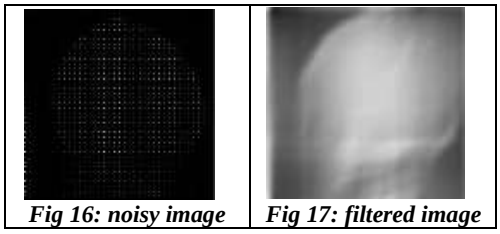


Fig 16: noisy image *Fig 17: filtered image*

Comparing between sequential time & parallel time:

The filtering time in the sequential program is1.5ms and the filtering time in the parallel program are 0.119ms.

Computing standard division:

Standard C (σ) between the original image and the image we get after filtering is 0. 1890

Summarization of Filtering using Gaussian lowpass filter

In the previous implementation we use different value with D◦ to get the best filtering and the lowest standard deviation between original image and the filtering image.

Fig 18 shows the different value of D◦ and the corresponding standard deviation we get it.

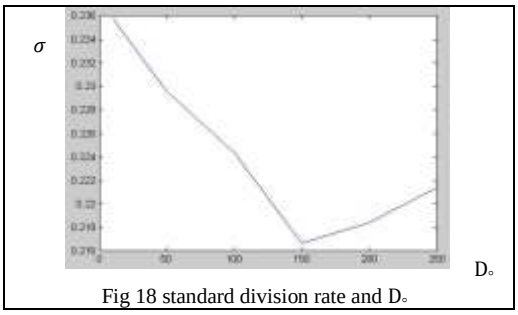


Fig 18 standard division rate and D◦

We get that the best value of D◦ with Gaussian lowpass filter is range between 100 and 150.

2-Filtering using Highpass filter

Highpass filter is a type of filters that help to improve sharpening effect on the images. It can attenuate the low frequency components without disturbing high-frequency information in the Fourier transform [2].The transfer function of the Gaussian highpass filter (GHPF) with cutoff frequency locus at a distance D◦ from the origin is given by:

$$H(u,v) = 1 - e^{-D^2(u,v)/2D_\circ^2}$$

Where $D(u,v)$ is the distance from point (u,v) to the center of the frequency rectangle, if the image of size M×N so the frequency rectangle is at $(u,v) =$ (M/2, N/2) that are given by:

$$D(u,v) = \left[\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2\right]^{\frac{1}{2}}$$

In examples we choose D◦ = 100 to be changed at will and the size of the noisy image is 200-by-200.

We are implement the Gaussian highpass filter on images has corrupted by different types of noise that we have discussed before.

I. Filtering Gaussian noisy image:

Here we are use highpass filter with $D(u,v) =$ D◦ =100 and Chest ribs image was corrupted by Gaussian noise shows in fig 19, the result shows in fig 20

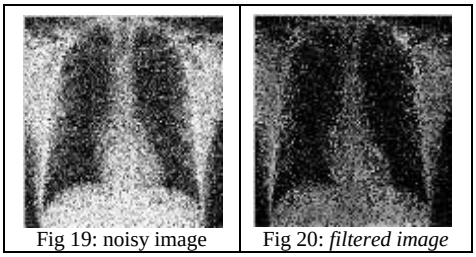


Fig 19: noisy image | Fig 20: *filtered image*

Comparing between sequential time & parallel time:

The filtering time in the sequential program is 319.3 ms and the filtering time in the parallel program is 0.13577 ms.

Computing standard deviation:

Standard deviation (σ) between the original image and the image we get after filtering is 0.5424

II. Filtering salt & pepper noisy image:

next example of filtering Chest ribs image is noising by salt and pepper noise shows in fig 21, and the Gaussian Highpass filter is implement two times with cutoff frequency $D(u,v) =$ D◦ = 100 then the result is shows in fig 22:

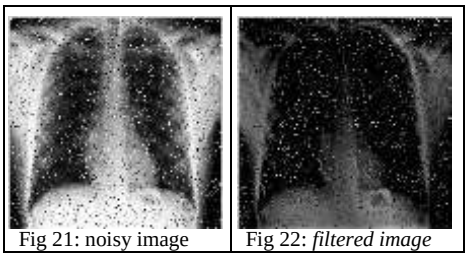


Fig 21: noisy image | Fig 22: *filtered image*

Comparing between sequential time & parallel time:

The filtering time in the sequential program is 4.2ms and the filtering time in the parallel program is 0.503 ms.

Computing standard deviation:

Standard deviation (σ) between the original image and the image we get after filtering is 0.5290

III. Filtering sinusoidal noisy image

In this example we using Chest ribs image noising by additive periodic noise shows in fig 23, and the Gaussian Highpass filter with cutoff frequency $D(u,v) =$ D◦ = 100 the result is shows in fig 24:

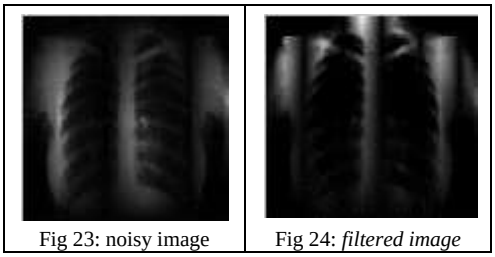


Fig 23: noisy image | Fig 24: *filtered image*

Comparing between sequential time & parallel time:

The filtering time in the sequential program is 1.4 ms and the filtering time in the parallel program is 0.47 ms

Computing standard deviation:

Standard deviation (σ) between the original image and the image we get after filtering is 0.4547

IV. Filtering random noisy image:

Here we are filtering a skull image noising by two dimensions random matrix shows in fig 25 by Gaussian Highpass filter with cutoff frequency $D(u,v) =$ D◦ = 100 and the result is shows in fig 26:

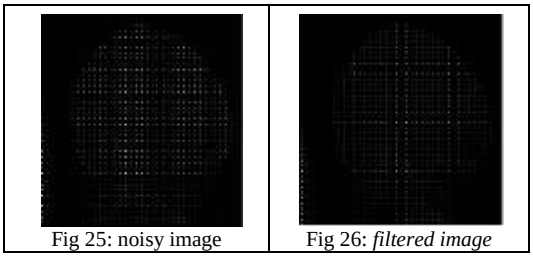


Fig 25: noisy image | Fig 26: *filtered image*

Comparing between sequential time & parallel time:

The filtering time in the sequential program is1.8ms and the filtering time in the parallel program are 0.576 ms

Computing standard deviation:

Standard deviation (σ) between the original image and the image we get after filtering is 0.1078

Summarization of Filtering using Gaussian highpass filter

In the previous implementations we use different value with D$_0$ to get the best filtering and the lowest standard deviation between original image and the filtering image.

Fig 27 shows the different value of D$_0$ and the corresponding standard deviation we get it.

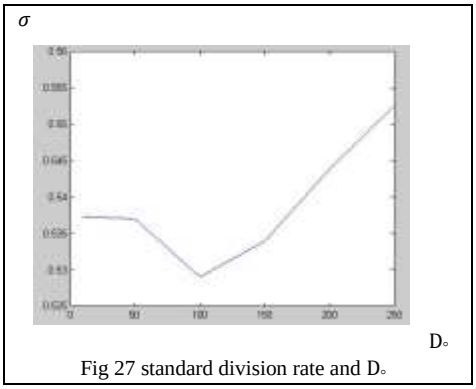


Fig 27 standard division rate and D$_0$

So we get the best value of D$_0$ with Gaussian highpass filter is 100.

Conclusion and future work

In this paper, we discussed filtering noisy images given different types of filtering and different types of noise.

The **Gaussian lowpass** filter is improve smoothing (blurring) effect on the noisy image in the frequency domain it is efficient with sharp noisy image. **Gaussian Highpass** filter improves sharpening effect on the noisy image in the frequency domain it is attenuates the low-frequency component without disturbing high-frequency information in the Fourier transform.

In this paper we use parallel programming for implementing all filtering and we prove that the parallel execution time of filtering is much minimum than sequential execution time about 8 times that we have using 8 threads and if we increase the number of used thread the execution time is decreased, but it remains constant after that, fig 28 Fig 28 is showing the result of serial time and parallel time with different numbers of threads computed by running the image enhancement algorithms by taking the different images at different size.

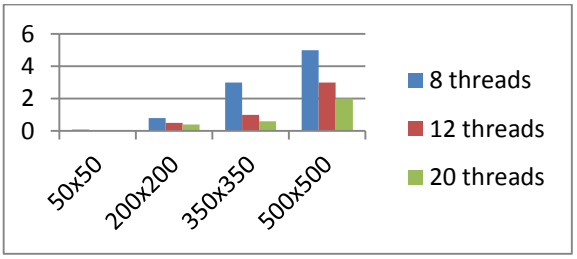


Fig 28 relation between the number of threads & the execution time

In this paper we implementing filtering on black & White images and the work is ongoing to apply filtering on coloring image in future.

References

[1] Thomas Rauber and Gudula Runger,"Parallel Programing For Multicore and Cluster System",Springer-Verlag Berlin Heidelberg,2010

[2] Rafael C.Gonzalez and Richard E.Woods, "Digital Image Processing",Prentic Hall,Inc, 2002.

[3] William Smith, Matlab's Parallel Computation Toolbox and the Parallel Interpolation of Commodity Futures Curves, March 2010

[4] Rafael C.Gonzalez,Richard E.Woods and Steven L.Eddins,"Digital Image Processing Using Matlab",Tata McGraw Hill Education Private Limited, 2e,2010.

[5] mathworks,"Parallel Computing Toolbox" www.mathworks.com/products/parallel computing