Median Filter in Agriculture

A.Senthil Rajan, E. Ramaraj

Abstract: Noise is caused by malfunctioning pixels in camera sensors, faulty memory locations in hardware, or transmission in a noisy channel. Two common types of impulse noise are the salt-and-pepper noise and the random-valued noise. There are many works on the restoration of images corrupted by impulse noise. The median filter was once the most popular nonlinear filter for removing impulse noise, because of its good denoising power and computational efficiency. Open MP is an extensive and powerful Application Programming Interface (API) that supports much functionality required for median filter. Median filter using OpenMP, the corrupted insect pest in paddy field can be analyzed and effective measures can take immediately to eradicate the harmful insect.

Keywords: insect, median filter, noise, openMP

I. OPENMP

Open Multiprocessing (OpenMP) for the use of open multithread median filter applications to take advantage of multicore general-propose CPUs. Open MP is an extensive and powerful Application Programming Interface (API) that supports much functionality required for parallel programming. More sophisticated applications could be build on similar principle.

A key challenge in parallel computing has been the lack of a broadly supported, simple to implement parallel programming model. Software programmers subsequently found it difficult to adapt applications to take advantage of multicore hardware advances OpenMP was designed to bridge this gap, providing an industry standard, parallel Application Programming Interface API for shared memory multiprocessors, including multicore processors. Support for OpenMP is currently available in most modern Fortran and C/C++ compilers as well as numerous operating systems, including Microsoft Windows, Linux and Apple Macintosh OS X. Version 1.0 of OpenMp. OpenMp is certainly not the only way of achieving parallelism on multicore systems. Other implementation models, such as CLICL, Pthreads, and MPI[3][4] exist and may be a good choice depending on the hardware, application, and the preference of the programmer.

A. Senthil Rajan is the Head of the Department Master of Computer Application, Jyoti Nivas College, Bangalore, India. IEEE and IAENG Member.
(e-mail: agni_senthil@yahoo.com) .

E. Ramaraj is the Director, Computer Department, Alagappa University, Karaikudi, India.

A. USING OPENMP

OpenMp works as a set of preprocessor directives, run-time library routines, and environment variables provided to the programmer, who instructs the complier how a section of code can be multithreaded. FORTRAN, the directives appear as comments, while in C/C++ they are implemented as pragmas. In this way, compilers that do not support the standard will process and potentially optimize the codeMP API is independent of the machine/operating system, properly written OpenMP code for one platform can easily be recompiled and run on another platform.

An open MP application always begins with a single thread of control, called the master thread. Which exists for the duration of the program? The set of variables available to any particular thread is called the thread's execution context. During execution, the master thread may encounter parallel regions, at which the master thread will fork new threads, each with its own stack and execution context. At the end of the parallel region, the forked threads will terminate, and the master thread continues execution. Nested parallelism, for which forked threads further threads, is supported.

B. LOOP LEVEL PARALLELISM

Parallelism is added to an application by including pragmas, which, in C++, have the following form:

#pragama omp < directive > [clauses]

There are numerous directives, but here we focus on the parallel for directive, which offers a simple way to achieve loop-level parallelism, often existing in signal and image processing algorithms. The optional clauses modify the behavior of the directive.

The parallelization of loops is the most common use of OpenMP.

C. VARIABLE SCOPE

Every thread has its own execution stack that contains variables in the scope of the thread. When parallelizing code, it is very important to identify which variables are shared between the threads, and which are private. In the parallelized sine wave example above, the variables x,A,w, N were shared, while n was private, that is each thread has its own n but shares all the other variables.

OpenMp provides explicit constructs to specify shared and private variables in the execution stack. By default, all variables are shared except

1) The loop index.
2) Variables local (declared within) the loop.
3) Variables listed in private clauses.

ISBN: 978-988-17012-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Sharing the private variables using directive clauses.

```
#pragma omp parallel for
Private (n)
Shared(A, x, w)
for((int  n=0; n< N; n++)
x[n] = A * sin (w *n);
```

copies of the variables in the private clause will be placed in to each thread's execution context

D. SCHEDULING

Earlier static scheduling, which divides work of the loop evenly among the different threads. Static scheduling is the default work sharing constructs and works well for balanced loops that have a relatively constant cost per iterations. However, some loops are unbalanced, with some iterations taking much longer than others. For such loops, static scheduling is not ideal, as fast threads will complete their work early and block until slow threads have completed their work. With OpenMP, it is possible to specify the scheduling mechanism example using the static and dynamic clause. In dynamic schedule, the numbers of iterations for each thread can be vary depending on the workload. When free, each thread requests more iteration until loop is complete. Dynamic scheduling is more flexible but does add additional overhead in coordinating the distribution of work amongst the threads.

By default, the system will decide how many threads to fork during run time. The number of spawned threads can be retrieved using

Integer omp_get_num_threads(void).

In addition, the number of threads may be set using omp_set_num_threads (integer)

II. MEDIAN FILTERING

Median filtering is a commonly applied nonlinear filtering technique that is particularly useful in removing speckle and salt-and-pepper noise [1][2]. Simply put, an image neighborhood surrounding each pixel is defined, and the median value of this neighborhood is calculated and is used to replace the original pixel in the output image.

$$I_{med}[x,y] = median(I_{orig}[i,j], i,j \in nbor[x,y]) \qquad (1)$$

In this example, a square neighborhood around each pixel, defined using the half width of the neighborhood, i.e., for a half width of n, the number of pixels in the neighborhood would be $(2n+1)^2$. At each pixel, the functions GetNbors retrieve the neighbors; neighbors some that lies outside the image domain are assigned to be that of the nearest pixel within the image boundary. These neighbors are then sorted using the shear sort or quick sort in C++ program and the median selected.

On a 512 X 512 insect image, and using a quad-core 2.4 GHz CPU, the result of median using half width of three, i.e., the number of neighbors= $(2*3+1)^2 = 49$

III. SYSTEM EVALUATION

The experiment was carried out on a Intel core TM core Processor Q6600 2.40 GHZ system with 2 GB shared RAM, 500GB SATA hard disk , and double D-link 1Gbps network adapters. The normal application is a PC-based Linux router function to forward packets at above 50Mbps. The performance of this application is maintained throughout the whole experiment. We use the image data sets which are replayed at controlled transmission rate.

The system needs to capture the incoming image from the network adapter and analyze these images or compare the image without any noise (Processing speed), and the speed of incoming images (network speed), the system may be able to process all images, or have to drop some images. If the processing speed is slower than the network speed (this may happen when the system is under DDoS attack), the system may drop some images thus may lose some attacking information, which will increase false positive and negative rate.

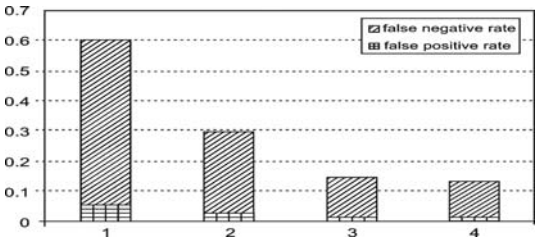

Figure 1 Error rate by different number of course

False positive and negative rate by different number of cores used

The figure1 shows the error rate when different numbers of cores are used when the transmission interval is 0.001 second. The trend in this figure also shows when the parallelized in to different cores, the error rates can be significantly reduced. As a large portion of images are dropped with errors in a single core scenario, the false negative rate is high as 54.3 % and false positive rate is high as 5.49%. If more cores are utilized, the images with errors are reduced. For examples, a 4 core scenario, the false negative rate can be reduced to 11.8% and the false positive rate can be reduced to 1.09%

*Par*allelized code for median filtering using a half width of three

```
int x y, halfwidth, nborSize;
PixelType nbors[MAX_NBOR_SIZE];
halfWidth = 3;
nborSize = 2*halfwidth +1;
nborSize *=nborsize;
#pragma omp parallel for
\Shared(inputImage,outputImage,structuring
Element, width, height)\
```

ISBN: 978-988-17012-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

```
Private(x,y,nbors)firstprivate(halfwidth,nborsize)
schedule (static)
for (y=0; y<height; y++){
for (x=0; x<width; x++){

GetNbors(inputImage,x,y,width,height,halfwidth,nbors);
Sort(&nbors[0],&nbors[nborSize]);
int index = x + y * width;
outputImage[index]= nbors[nborSize/2];
}
)
```

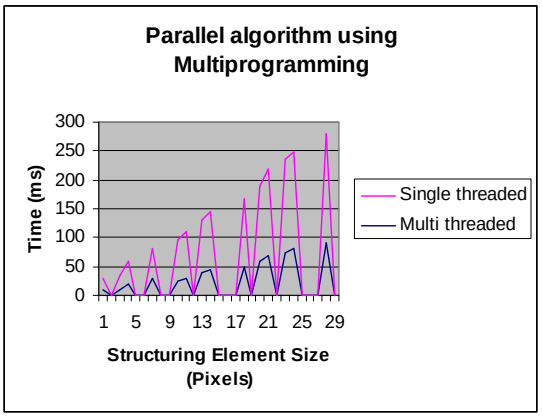

Figure 2: Parallel algorithm using multi programming

IV. PADDY PEST

Paddy is the most staple crop in many countries and many insect pests cause great damage to this crop by attacking the roots, stem, leaves and even the young grains when they are in milk stage.

As yellow stem borer (Tryporyza incertulas), swarming caterpillar (Spodoptera mauritia), Pamripoka (Dicladispa armigera) etc. are important major pests of paddy. As green bug (*Schizaphis graminum*) becomes minor pest of paddy in some exceptional cases.

Stem borer and leaf folder are major insects whose are affecting paddy productivity by attacking the crop at vegetative and reproductive stage. In some cases, impact of stem borer and leaf folder attack is so severe that it reduces paddy yield by 35-40 per cent. Farmers are applying unidentified and sub-standard chemicals, which do not control stem borer and leaf folder very effectively. Application of recommended doses of insecticides for control of stem borer and leaf folder will increase the productivity and returns from paddy.

Methods

1. Getting the image of the affected paddy crops or getting the image of the insects using the cell phones and sent to the administrator analyzer.
2. The administrator receives the image and removes the noise portion of the image using median filter Open MP.
3. Analyze the image using the image comparer and with database.
4. Extract the solution to solve the problem.
5. Sending back the solution to the user.
6. Get back the feedback from the user for every ten days.

The following table shows the successful results for one acre.

TABLE 1: Results for one acre

Paddy Type	Amount Spent for cultivation in (Rs.)	Total Amount gain by traditional	Total Amount gain by Median Filters
ADT 45	12,000	18,900	28,000
Culture	13,000	27,000	38,000
IR20	12,500	14,700	27,000
Deluxe Poni	15,500	30,000	45,000

V. CONCLUSION

Open MP is an extensive and powerful application programming interface (API) that supports much functionality required for median filter. Multicore processor using the parallel algorithm, median filter filters the noise efficiently with less processing time. Median filter using OpenMP, the corrupted insect pest in paddy field can be analyzed and effective measures can take immediately to eradicate the harmful insect.

REFERENCES

[1]. Dr. Ramaraj and A. Senthiil rajan “High density impulse noise removal in color images using ROIMCAR weighted median filter. IAENG,March.,vol2,pp.1481-1485,2010.
[2]. Dr. Ramaraj and A. Senthiil Rajan,” Using multi-core processor to support network parallel image processing application”,”IEEE signal processing, May.,vol1, pp232-235,2009
[3]. G.Blake,R.G.Deslinski,and T.Mudge,”A Survey of multicore processors; Areview of their common attributes,”IEEE Signal Processing Mag.,vol.26,n0.6,pp.26-37,2009.
[4]. H.Kim and R.Bond,” Multicore software technologies: Asurvey,”IEEE Singnal Processing May., vol26,n0.6,pp.80-89,2009.

ISBN: 978-988-17012-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

[5]. R.Chandra,l.Dagum,D.Kohr,D.Maydan,J.McDonald , and R.Menon,Parallel Programming in OpenMp, 1st ed. San Mateo,CA;Morgan Kaufmann,2001.
[6]. C.Johnson and J.Welser, "Future Processors:Flexible and Modular", Proceeding of 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, pp.4-6,2005
[7]. H.Sutter and J.Larus,"software and Concurrency Revolution", ACM Queue, vol.3, no 7, pp.54-62, 2005.
[8]. O.Villa, D.P.Scarpazza and F.Petrini, "Accelerating Real-Time String Searching with Multi-core Processors ".IEEE Computer, Vol.41, no.4,oo.42-50, 2008.
[9]. S.Dhamaparikar,P.Krishnamurthy, T.S.Sproull, J.W.Lockwood, "Deep packet Inspection Using Parallel Bloom Filters " , IEEE Micro, vol.24, no. 1 , pp.52-61, 2004.
[10]. H.Liu, .Zheng,B.Liu,X.Zhangand Y.Liu, "A Memory-Efficient Parallel String Matching Architecture for High-Speed Intrusion Detection", IEEE Jouranl on Selected Areas in Communications, vol.24, no.10,pp.1793-1804, 2006.
[11]. C. L. Hayes and Y.Luo, " DIPICO: A High Speed Deep Packed Inspection Engine Using Compact Automata " , Proceedings of ACM/IEEE ANCS "07, pp. 195-203, 2007.
[12]. Preap,V ,GC Jahn, K.Hin,N.Siheng, 2005. Fish and rice management system to enable agricultural diversification. Paper presented at the 5th Asia Pacific Congress of Entomology, 18-21 October 2005, Jeju, Korea.

ISBN: 978-988-17012-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)