A Novel Parallel Method for Speckle Masking Reconstruction Using the OpenMP

XUEBAO LI and YANFANG ZHENG

College of Electrical and Information Engineering, Jiangsu University of Science and Technology, Zhangjiagang 215600, China; zhengyf@ynao.ac.cn

Received July 6, 2016; accepted August 4, 2016

Abstract: High resolution reconstruction technology is developed to help enhance the spatial resolution of observational images for ground-based solar telescopes, such as speckle masking. Near real-time reconstruction performance is achieved on a high performance cluster using the Message Passing Interface (MPI). However, much time is spent in reconstructing solar subimages in such a speckle reconstruction. We design and implement a novel parallel method for speckle masking reconstruction of solar subimage on a shared memory machine using the OpenMP. Real tests are performed to verify the correctness of our codes. We present the details of several parallel reconstruction steps. The parallel implementation between various modules shows a great speed increase as compared to single thread serial implementation, and a speedup of about 2.5 is achieved in one subimage reconstruction. The timing result for reconstructing one subimage with 256×256 pixels shows a clear advantage with greater number of threads. This novel parallel method can be valuable in real-time reconstruction of solar images, especially after porting to a high performance cluster.

Key words: solar telescope: image processing: parallel computing: OpenMP

1. Introduction

The spatial resolution of a large aperture ground-based solar telescope is degraded by the atmospheric turbulence. Raw observational images from the telescope have characteristics of motion, blurring, and geometrical distortions. Many post-processing reconstruction methods have been developed to help enhance spatial resolution of observational images, such as shift-and-add (SAA) (Li et al. 2014; Sudo et al. 2006), phase diversity (Gonsalves 1982; Paxman et al. 1992) and speckle masking (Lohmann et al. 1983; von der Lühe 1993). Due to reliable performance of phase reconstruction, the speckle masking is most commonly used, even though it is computationally expensive. The speckle masking method was applied to solar image reconstruction for the New Vacuum Solar Telescope (NVST) (Li et al. 2014). Near real-time reconstruction performance was achieved for the NVST on a high performance cluster using the Message Passing Interface (MPI). The parallel program we developed reconstructs one image with 2368×1920 pixels from one burst with 100 short exposure images in about 48 seconds (Li et al. 2015). However, much time (one-third of the computation time) was spent in reconstructing all subimages in such a speckle reconstruction. Real-time image reconstruction can not only enhance the efficiency of observational data analysis, but also relieve the pressure of data storage and processing. Therefore, in order to further improve the acceleration performance, it is absolutely necessary to parallelize and speedup subimage reconstruction.

Some aspects of using speckle masking and MPI to reconstruct solar image in near time have been already researched by Denker et al. (2001), Wöger et al. (2008), and Cao et al. (2010). However, there is little literature on the parallelization of the speckle masking algorithm. Only Wöger & Ferayorni (2012) researched the acceleration of speckle masking algorithm in one solar subimage reconstruction on Graphics Processing Units (GPU). Upon analyzing speckle masking algorithm in subimage reconstruction, we found that the average bispectrum computation is most time-consuming. The algorithm described in this paper takes a different approach to accelerate the average bispectrum computation using the Open Multiple Processing (OpenMP). OpenMP is the most popular solution and has been widely used in real-time astronomical data processing. For example, Giant Metrewave Radio Telescope (GMRT) used the OpenMP to accelerate the flagging and calibration of radio interferometric data (Jayanti & Jayaram 2012). The parallel computation of the nonlinear force-free field extrapolation of the coronal magnetic field using the data obtained by the Hinode satellite was realized through OpenMP on a multi-CPU machine (He et al. 2011).

In this paper, we report on a parallel implementation of the speckle masking algorithm for solar subimage reconstruction using the OpenMP. Our method mainly differs from Wöger & Ferayorni (2012) in that the method uses the OpenMP to accelerate the average bispectrum computation algorithm, and promotes the computation speed of solar subimage reconstruction significantly. In Section 2, we describe the novel parallel method for speckle masking algorithm. In Section 3, we present results and analysis. Finally, we present our
conclusions and future work in Section 4.

2. THE NOVEL PARALLEL METHOD FOR SPECKLE MASKING ALGORITHM

In general, a high resolution image is statistically reconstructed from one burst with at least 100 short exposure images. Because speckle masking is only valid for a small region, the isoplanatic patch, one burst with at least 100 preprocessed images are divided into a number of partially overlapping subimage bursts with $100 \times 256^2$ pixels (Li et al. 2015). These subimage bursts are reconstructed into subimages using speckle masking separately, then all reconstructed subimages are combined to form an entire high resolution image (Liu et al. 2014). The speckle masking algorithm, based on Fourier phase reconstruction by use of the bispectrum, is defined by

$$B(u, v) = \langle I_i(u) I_i(v) I_i^*(u + v) \rangle,$$
1: #define ELEM(r,c,colnum) (r*colnum+c)
2: Batched 2D subimage preprocessing on a single core CPU
3: omp_set_num_threads(10);
4: ...
5: //Batched 2D subimage FFT, SampleSize=100
6: for(i=0;i<SampleSize;i++)
7: { for(j=0;j<subimg_nelements;j++)
8: datatmp[j]=subimgs[i][j]*bppf[j];
9: NVST_FFT_FORWARD(subimg_rownum, subimg_colnum, datatmp, datafft[i]);
10: }
11: ...
12: //Batched 2D spectrum shift
13: #pragma omp parallel for
14: for(i=0;i<SampleSize;i++)
15: shift_complex(datafft[i], subimg_rownum, subimg_colnum, subimg_rownum/2, subimg_colnum/2);
16: ...
17: //Batched 2D spectrum subimage extract,
18: //Mfrx=subimg_colnum/2=128, Mfry=subimg_rownum/2=128
19: #pragma omp parallel for private(j,ii,jj)
20: for(i=0;i<SampleSize;i++)
21: { j=0;
22: for(ii=subimg_rownum/2-Mfry;ii<=subimg_rownum/2+Mfry-1;ii++)
23: for(jj=subimg_colnum/2-Mfrx;jj<=subimg_colnum/2+Mfrx-1;jj++)
24: datafft_temp[i][j++]=datafft[i][ELEM(ii,jj,subimg_colnum)];
25: }
26: //5D bispectrum computation
27: #pragma omp parallel for
28: for(i=0;i<SampleSize;i++)
29: NVST_BIP_CAL(datafft_temp[i],M5,bp1[i], bp2[i], 2*Mfry, 2*Mfrx);
30: //average 5D bispectrum computation
31: #pragma omp parallel for private(ii, jj, i, j, kk)
32: for(k=0;k<SampleSize;k++)
33: for(ii=0;ii<M5+1;ii++)
34: for(jj=0;jj<M5+1;jj++)
35: for(i=0;i<Mfry;i++)
36: for(j=0;j<Mfrx;j++)
37: { kk=omp_get_thread_num();
38: bp1_add[kk][ii][jj][i][j]=gsl_complex_add(bp1_add[kk][ii][jj][i][j],bp1[k][ii][jj][i][j]);
39: bp2_add[kk][ii][jj][i][j]=gsl_complex_add(bp2_add[kk][ii][jj][i][j],bp2[k][ii][jj][i][j]);
40: }
41: //average 4D bispectrum computation, coreNum=10, M5=5
42: #pragma omp parallel for private(i, j, k)
43: for(k=0;k<coreNum;k++)
44: for(i=0;i<M5+1;i++)
45: for(j=0;j<Mfrx;j++)
46: { bp1_add_all[i][j][ii][jj][k]=gsl_complex_add(bp1_add_all[i][j][ii][jj][k],bp1[k][ii][jj][i][j]);
47: bp2_add_all[i][j][ii][jj][k]=gsl_complex_add(bp2_add_all[i][j][ii][jj][k],bp2[k][ii][jj][i][j]);
48: }
49: 50: Fourier phase and modulus recovery on a single core CPU
51: One subimage is reconstructed on a single core CPU

Figure 2. The pseudo code for subimage reconstruction using speckle masking based on OpenMP.

\[ O_i(u)O_i(v)O_i^*(u + v) \cdot < H_i(u)H_i(v)H_i^*(u + v) >_i, \]  
where \( u \) and \( v \) is the two-dimensional spatial frequency. \(< I_i(u)I_i(v)I_i^*(u + v) >_i \) represents the average speckle masking bispectrum. \(< H_i(u)H_i(v)H_i^*(u + v) >_i \) denotes the average speckle masking transfer function. After obtaining average bispectrum, the object’s phase can be recovered from average bispectrum. A detailed description of the technical parts of the phase reconstruction method is given by Pehlemann & von der Lühe (1989).

OpenMP is an open standard for portable high performance computing. Using the OpenMP standard, it is easy to solve the traditional programming problem of load balancing because OpenMP does it automatically, we need not consider which part of calculation should be done on which CPU core (Chapman et al. 2007). In
Table 1

<table>
<thead>
<tr>
<th>Module</th>
<th>Serial program(s)</th>
<th>Parallel program(s)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batched 2D spectrum shift</td>
<td>0.35</td>
<td>0.04</td>
<td>8.75</td>
</tr>
<tr>
<td>Batched 2D spectrum extract</td>
<td>0.20</td>
<td>0.02</td>
<td>10</td>
</tr>
<tr>
<td>Average bispectrum computation</td>
<td>5.10</td>
<td>1.10</td>
<td>4.64</td>
</tr>
<tr>
<td>Subimage reconstruction</td>
<td>6.70</td>
<td>2.70</td>
<td>2.48</td>
</tr>
</tbody>
</table>

Figure 3. (a) One subimage of one subimage burst of 100 × 256^2 pixels before reconstruction. (b) The parallel reconstruction subimage using speckle masking based on OpenMP.

In our work, we propose a novel parallel method for speckle masking algorithm based on OpenMP. In the OpenMP model, multi-threads accelerate the computational intensity and time-consuming codes, while single thread continues to execute serial codes. Upon analyzing speckle masking algorithm, we find that computation of the average bispectrum is the most time-consuming process. Therefore, in the algorithm, the main functions such as batched two-dimensional (2D) Fourier spectrum shift, batched 2D spectrum subimage extract, and five-dimensional (5D) bispectrum computation are implemented in C language through OpenMP multi-threads. The flow chart of the novel parallel method for speckle masking algorithm based on OpenMP is shown in Figure 1. Several functions are implemented layer by layer, and inside the parallel region, each function is performed by multi-threads simultaneously and independently.

One subimage burst was composed of 100 batch of 2D 256×256 pixels subimages. The implementation of parallel method for one subimage reconstruction covers the following steps, and the pseudo code for subimage reconstruction using speckle masking based on OpenMP is shown in Figure 2. Firstly, the single thread continues to preprocess the subimage burst, executes the batched 2D FFT operation, and then acquires 100 batch of 2D spectrum. Secondly, the outer FOR loop controls the statistical sample size within one subimage reconstruction. For the functions of batched 2D spectrum shift and batched 2D spectrum subimage extract, data dependencies do not exist, so this loop could be unrolled, and then these two functions are implemented by multi-threads respectively and simultaneously. Thirdly, differently from the direct use of average four-dimensional (4D) bispectrum computation in serial codes: 5D bispectrum is added, and multi-threads are employed to perform 5D bispectrum computation and average 5D bispectrum computation respectively and simultaneously. Fourthly, since the loop index is reduced to 10 from 100, the average 4D bispectrum could be quickly acquired from average 5D bispectrum, and the object’s phase is reconstructed from the average 4D
The data was composed of 100 batch of 256×256 pixels subimages in the photosphere TiO channel. Figure 3a shows one raw subimage of one subimage burst before reconstruction. Figure 3b shows the parallel reconstruction subimage using speckle masking through OpenMP, and its field of view is slightly smaller than that of raw image. The spatial resolution of reconstructed subimage is enhanced significantly compared to that of raw subimage. The result on reconstruction accuracy using multi-threads parallel implementation is the same as in the single thread serial implementation. All experiments were conducted under Linux system on a shared memory machine equipped with an Intel Xeon E5-2620 CPU at 2.00 GHz (total 16 CPU cores), 32 GB random access memory (RAM). For the tests, we used a burst of 100 subimages, and this data set was reconstructed with 166464 average 4D bispectrum values.

Table 1 shows the execution time comparison of one subimage reconstruction between various modules with single thread serial implementation and 10 threads parallel implementation. As shown in the table, the runtime of reconstructing one entire subimage in parallel reduces to around 2.7 seconds, outperforming the single thread serial implementation by a factor of about 2.5. The most time-consuming module – average bispectrum computation, including 5D bispectrum computation, average 5D bispectrum computation, and average 4D bispectrum computation, shows a great speed increase by a factor of about 4.6. The processing speed of the other two modules is about 8-10 times faster than that of single thread serial implementation. From the comparison results above, we can see that the processing speed of solar subimage reconstruction together with several reconstruction steps is promoted significantly.

Figure 4 shows the execution time and relative speedup ratio of one solar subimage reconstruction for the number of threads ranging from 1 to 16. A clear advantage is seen for increasing number of threads. As shown in Figure 4, as the number of threads increases, the execution time reduces and the relative speedup ratio increases. However, there is a saturation at a minimum of around 2.7 seconds in Figure 4a (at a maximum of around 5 in Figure 4b) when the number of threads exceeds 10. The probable reason is that there are some serial programs in the parallel implementation, and some overhead associated with the startup, revo- cation, and synchronization of multi-threads. Therefore, for accelerating one subimage reconstruction in the OpenMP model, 10 threads on each processor may prove to be an optimal choice.
Wöger, F., & Ferayorni, A. 2012, Accelerated Speckle Imaging with the ATST Visible Broadband Imager, SPIE, 8451, 84511C-1