Application of parallel version two-dimensional fast Fourier transform algorithm, analog of the Cooley-Tukey algorithm, for digital image processing of satellite data

Mikhail Noskov [1*], *Valeriy* Tutatchikov[1], *Mikhail* Lapchik[2], *Marina* Ragulina[2], and *Tatiana* Yamskikh [1]
[1] Siberian Federal University, Institute of Space and Information Technology, Svobodny prospect 79, Krasnoyarsk, 660041, Russian Federation
[2] Omsk State Pedagogical University, Naberezhnaya Tukhachevskogo 14, Omsk, 644099, Russian Federation

Abstract. In modern systems of remote sensing two-dimensional fast Fourier transform (FFT) has been widely used for digital processing of satellite images and subsequent image filtering. This article provides a parallel version two-dimensional fast Fourier transform algorithm, analog of the Cooley-Tukey algorithm and its implementation for processing the satellite image of Krasnoyarsk and its suburban areas.

Introduction

At present, Earth remote sensing is closely related to digital image processing, as the aerospace images that are rich in detail are commonly represented in digital form of a raster type. Adjustment of contrast and brightness, spatial filtering, Fourier transform and subsequent frequency filtering are often used to improve image quality [1]. The traditionally applied algorithm for computing two-dimensional fast Fourier transform (FFT) is the sequential application of a one-dimensional FFT, first for all rows, then for all columns. The article describes a version two-dimensional fast Fourier transform algorithm, analog of the Cooley-Tukey algorithm**,** with the reduced number of complex operations compared to the traditionally used algorithm. A variant of the algorithm parallelization to accelerate calculations is shown in the article. The use of the algorithms to perform image processing on digital images is considered.

The filtering procedure is implemented in several stages:
• Data reading and swapping
• Parallel computation of the Cooley-Tukey algorithm
• Filtering the obtained Fourier transform
• Reverse FFT Calculation
• Image acquisition

* Corresponding author: mvnoskov@yandex.ru

Detailed description of these stages can be found in [1]. A two-dimensional analogue of the Cooley-Tukey algorithm is considered in [2], its multidimensional variant in [3] and a more detailed description of the algorithm is provided in [4]. In this paper we present a parallel two-dimensional version of the Cooley-Tukey algorithm. First, we consider two dimensional analog of the Cooley-Tukey algorithm for computing the fast Fourier transform.

The Cooley-Tukey algorithm

Given a function *f(x,y)* of two-dimensional periodic signal with values in the complex space, where *x,y = 0, 1, ..., 2^s-1*, *s* is a positive integer. Grayscale image that is 2^s pixels in height and width can be taken as an example of such a function. The value of the function *f(x,y)* is a complex number, which real component is equivalent to the brightness values of the corresponding pixel with coordinates *(x,y)* in the range 0-255, and the imaginary component of the number is zero. Then the Fourier transform *F(u,v)* of this signal can be represented as follows:

$$F(u,v)=\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} f(x,y)\cdot e^{\frac{2\pi i(ux+vy)}{N}} \tag{1}$$

where $N=2^s$, *x,y = 0,1,...,N-1*.

We convert formula (1) as follows: the coordinates x and y are decomposed into even and odd parts. Then, the initial sum can be divided into four equations:

$$\begin{aligned}
F(u,v)=&\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\Bigg(f(2x1,2y1)\cdot e^{\frac{2\pi i(u\cdot 2x1+v\cdot 2y1)}{N}} + f(2x1+1,2y1)\cdot e^{\frac{2\pi i(u\cdot(2x1+1)+v\cdot 2y1)}{N}} + \\
&+ f(2x1,2y1+1)\cdot e^{\frac{2\pi i(u\cdot 2x1+v\cdot(2y1+1))}{N}} + f(2x1+1,2y1+1)\cdot e^{\frac{2\pi i(u\cdot(2x1+1)+v\cdot(2y1+1))}{N}}\Bigg)= \\
=&\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} f(2x1,2y1)\cdot e^{\frac{2\pi i(u\cdot 2x1+v\cdot 2y1)}{N}} + e^{\frac{2\pi iu}{N}}\cdot\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} f(2x1+1,2y1)\cdot e^{\frac{2\pi i(u\cdot 2x1+v\cdot 2y1)}{N}} + \\
&+ e^{\frac{2\pi iv}{N}}\cdot\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} f(2x1,2y1+1)\cdot e^{\frac{2\pi i(u\cdot 2x1+v\cdot 2y1)}{N}} + \\
&+ e^{\frac{2\pi iu}{N}}\cdot e^{\frac{2\pi iv}{N}}\cdot\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} f(2x1+1,2y1+1)\cdot e^{\frac{2\pi i(u\cdot 2x1+v\cdot 2y1)}{N}} = \\
=&\, g_{00}(u,v)+e^{\frac{2\pi iu}{N}}\cdot g_{10}(u,v)+e^{\frac{2\pi iv}{N}}\cdot g_{01}(u,v)+e^{\frac{2\pi iu}{N}}\cdot e^{\frac{2\pi iv}{N}}\cdot g_{11}(u,v)
\end{aligned} \tag{2}$$

Each of the sums *g(u,v)* obtained in (2) is a FFT signal with the elements $2^{s-1}\times 2^{s-1}$ for even or odd coordinates of the source signal *f(x,y)*

Then, in the next step , we divide the coordinates *u* and *v* into two equal-sized subsets *2u1*, *2v1* and *2u1+N/2*, *2v1+N/2*, respectively, *u1,v1 = 0,1,...,N/2-1*. Then for the factors $e^{\frac{2\pi iu}{N}}$ and $e^{\frac{2\pi iv}{N}}$ from the second subset we get:

$$\begin{aligned}
e^{\frac{2\pi i\cdot(2u1+N/2)}{N}} &= e^{\frac{2\pi i\cdot(2u1)}{N}}\cdot e^{\frac{2\pi i\cdot(N/2)}{N}} = e^{\frac{2\pi i\cdot(2u1)}{N}}\cdot e^{\pi i} = -e^{\frac{2\pi i\cdot(2u1)}{N}}, \\
e^{\frac{2\pi i\cdot(2v1+N/2)}{N}} &= e^{\frac{2\pi i\cdot(2v1)}{N}}\cdot e^{\frac{2\pi i\cdot(N/2)}{N}} = e^{\frac{2\pi i\cdot(2v1)}{N}}\cdot e^{\pi i} = -e^{\frac{2\pi i\cdot(2v1)}{N}}.
\end{aligned} \tag{3}$$

We plug (3) into the formula (2) to get:

$$
\begin{aligned}
&F(u1,v1) = g_{00}(u1,v1) + e^{\frac{2\pi i \cdot u1}{N}} \cdot g_{10}(u1,v1) + e^{\frac{2\pi i \cdot v1}{N}} \cdot g_{01}(u1,v1) + \\
&+ e^{\frac{2\pi i \cdot u1}{N}} \cdot e^{\frac{2\pi i \cdot v1}{N}} \cdot g_{11}(u1,v1), \\
&F(u1+N/2,v1) = g_{00}(u1+N/2,v1) - e^{\frac{2\pi i \cdot u1}{N}} \cdot g_{10}(u1+N/2,v1) + \\
&+ e^{\frac{2\pi i \cdot v1}{N}} \cdot g_{01}(u1+N/2,v1) - e^{\frac{2\pi i \cdot u1}{N}} \cdot e^{\frac{2\pi i \cdot v1}{N}} \cdot g_{11}(u1+N/2,v1), \\
&F(u1,v1+N/2) = g_{00}(u1,v1+N/2) + e^{\frac{2\pi i \cdot u1}{N}} \cdot g_{10}(u1,v1+N/2) - \\
&- e^{\frac{2\pi i \cdot v1}{N}} \cdot g_{01}(u1,v1+N/2) - e^{\frac{2\pi i \cdot u1}{N}} \cdot e^{\frac{2\pi i \cdot v1}{N}} \cdot g_{11}(u1,v1+N/2), \\
&F(u1+N/2,v1+N/2) = g_{00}(u1+N/2,v1+N/2) - \\
&- e^{\frac{2\pi i \cdot u1}{N}} \cdot g_{10}(u1+N/2,v1+N/2) - e^{\frac{2\pi i \cdot v1}{N}} \cdot g_{01}(u1+N/2,v1+N/2) + \\
&+ e^{\frac{2\pi i \cdot u1}{N}} \cdot e^{\frac{2\pi i \cdot v1}{N}} \cdot g_{11}(u1+N/2,v1+N/2),
\end{aligned}
\tag{4}
$$

where *u1,v1 = 0,1,...,N/2-1*.

Formula (4) describes the two-dimensional fast Fourier transform batterfly in analog of Cooley-Tukey algorithm. It is schematically depicted in Figure 1. With this butterfly we can split the source signal and the Fourier transform with elements into four sub-signals each with the number of elements $2^{s-1} \times 2^{s-1}$ [4]. This reduces the number of multiplications and additions of complex numbers required to compute FFT.

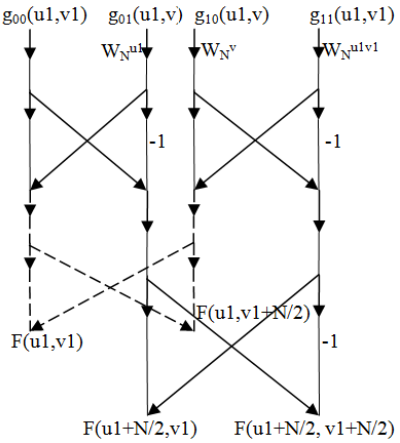

Fig. 1. Two-dimensional fast Fourier transform batterfly in analog of Cooley-Tukey algorithm

Decomposition (4) can be applied to each sub-signal *g(u,v)* several times until we get sub-signals of 4 elements, which FFTs are being computed directly. Then for the signal *f(x,y)* of the elements $N \times N$ the total number of multiplications required to compute complex numbers will be $\frac{3}{4} N^2 \log_2 N$, and the number of additions - $2N^2 \log_2 N$ [3]. For comparison, the standard method of computing two-dimensional FFT by rows and columns will require $N^2 \log_2 N$ multiplications and $2N^2 \log_2 N$ additions.

The parallel Cooley-Tukey algorithm

To test the running time of an algorithm for calculating two-dimensional fast Fourier transform algorithm, analog of the Cooley-Tukey algorithm, a simulation program was written in the C++ programming language [4]. Two principal methods of algorithm parallelization were used*:* OpenMP, targeted toward use on shared memory systems*,* and MPI for distributed memory systems. Testing was carried out on the cluster node of the SFU supercomputer with IBM HS21 XM Xeon Quad core E5450 3.0 GHz, 64 Gb RAM [5]. The result of testing for a system with distributed memory on a single cluster node is presented in Table 1.

The two-dimensional analogue of the Cooley-Tukey algorithm processes the signal with the number of samples $2^s \times 2^s$ in s iterations. In the first iteration, due to the preliminary permutation of the elements, the signal data is divided into quads of related (vertically and horizontally) elements spaced by one element from each other; in the second iteration, into quads of elements separated by two elements from each other; in the third iteration - by the 2^2 element; in the last s- iteration by- 2^{s-1} element. For such implementation of the algorithm, in shared memory systems parallel data structures are designed by splitting multiple data in each iteration into arrays of interconnected sets of the elements for each separate thread. In a distributed memory systems, a similar partitioning into sets of related data occurs at each iteration with subsequent transfer of data between the processes for independent computations.

Table 1. The result of testing a two-dimensional parallel analogue of the Cooley-Tukey algorithm in a system with distributed memory, in seconds

Size	The number of processes	2D FFT, for rows and columns	2D Cooley-Tukey FFT
1024*1024	1	0,490	0,310
	2	0,310	0,310
	4	0,220	0,270
	8	0,180	0,270
	16	0,180	0,350
2048*2048	1	2,300	1,330
	2	1,550	1,260
	4	1,060	0,950
	8	0,840	0,850
	16	0,850	1,000
4096*4096	1	9,880	5,850
	2	6,240	4,590
	4	4,370	3,590
	8	3,430	3,060
	16	3,740	3,390
8192*8192	1	43,210	25,190
	2	26,990	19,160
	4	18,550	14,870
	8	14,520	13,130
	16	14,090	11,700

The data are presented graphically in Figure 2, where the FFT algorithm by rows and columns is denoted by FFT RC, and the analogue of the Cooley-Tukey algorithm by FFT CT.

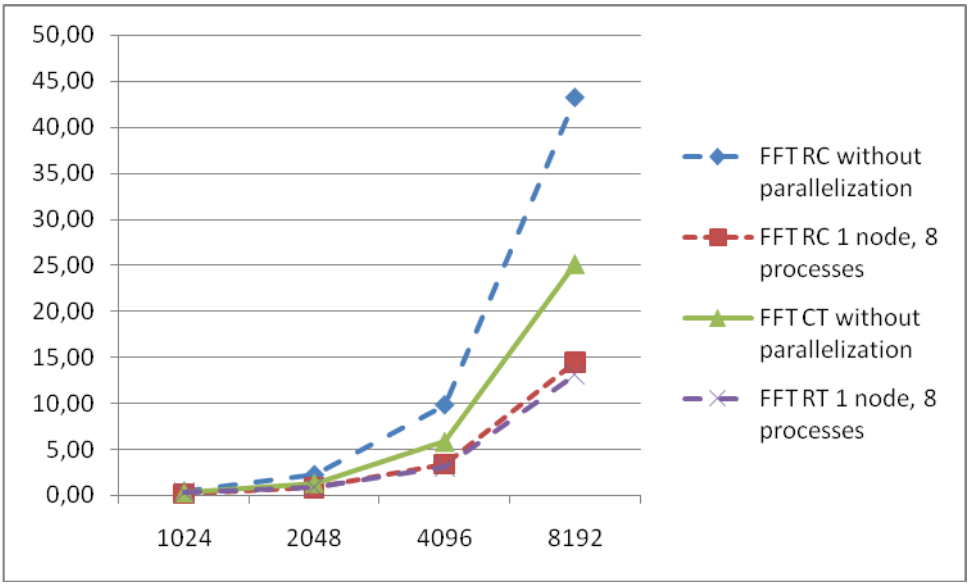

Fig. 2. Comparison of running time for various algorithms

Image filtering

LandSat-8 image of Krasnoyarsk and its surrounding areas dated April 7, 2016 [6], which is shown in Figure 1 on the left, was used as a test signal. The original image resolution is 8081 * 8171 pixels, it was converted to the nearest power of two**:** and then scaled for the powers 10-15. On the right side of the figure 3 the result from high-pass filtering is shown. In this case, the contours are more vivid: the river line, the boundaries of the rocky areas.

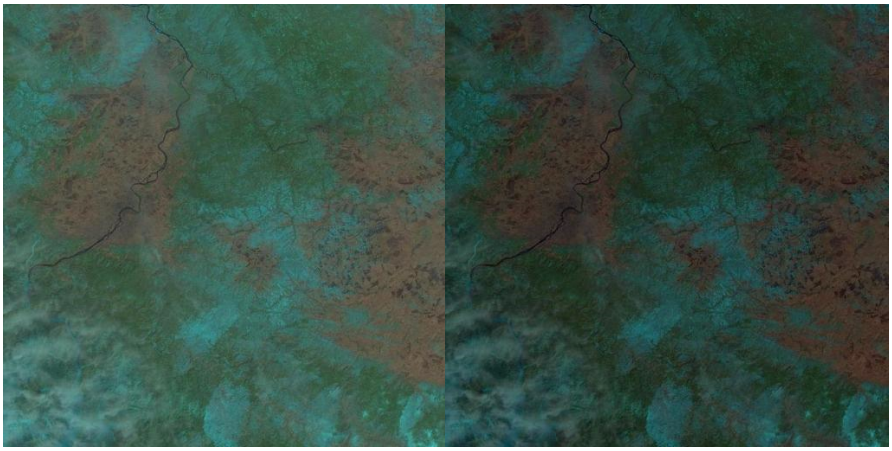

Fig. 3. Original image - The result from high-pass filtering

The original image is shown on the left side of the figure 4. The result from low-pass filtering is on the right side. In this case, small sharp changes in mountainous terrain are not so noticeable on the general background, that is, small details have been removed.

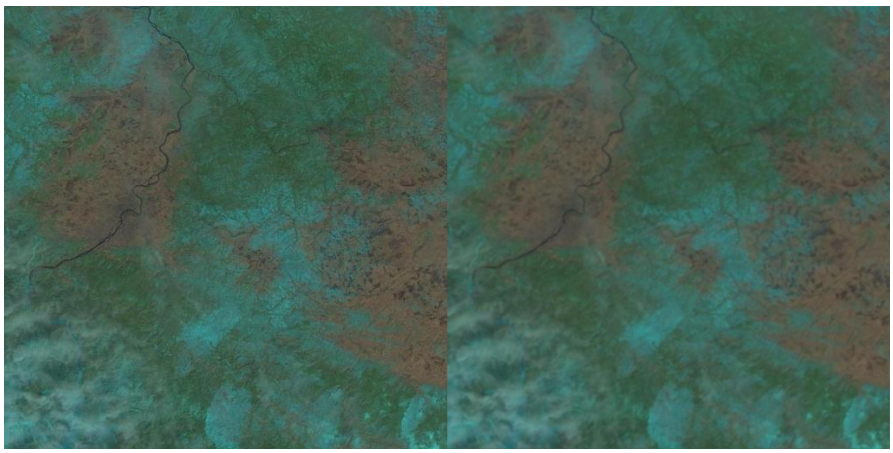

Fig. 4. Original Image - Result from low pass filtering

Conclusion

It is shown that the parallel analogue of the Cooley-Tukey algorithm for computing two-dimensional FFT is executed on average 4 times faster than the standard method for computing two-dimensional fast Fourier transform by rows and columns.

References

1. R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Second Edition, Pearson Education, Inc (2004)
2. R. Blahut, *Fast algorithms for digital signal processing*, Addison-Wesley Publishing Company (1985)
3. V.S. Tutatchikov, O.I. Kiselev, M.V. Noskov, *Calculating the n-Dimensional Fast Fourier Transform,* PRIA, **23(3)**, 429-433 (2013)
4. V.S. Tutatchikov, *Two-dimensional fast Fourier transform, Proceeding of 11th International Forum on Strategic Technology (IFOST-2016)*, 495 – 498 (2016)
5. Supercomputer complex of SFU Access mode: http://cluster.sfu-kras.ru/
6. LandSat-8 image of Krasnoyarsk and its surrounding areas Access mode: http://earthexplorer.usgs.gov/