USE OF A GPGPU MEANS FOR THE DEVELOPMENT OF SEARCH PROGRAMS OF DEFECTS OF MONOCHROME HALF-TONE PICTURES

V.A. Dudnik,* V.I. Kudryavtsev, T.M. Sereda, S.A. Us, M.V. Shestakov

National Science Center "Kharkov Institute of Physics and Technology", 61108, Kharkov, Ukraine

(Received January 16, 2012)

Application of a GPGPU means for the development of search programs of defects of monochrome half-tone pictures is described. The description of realization of algorithm of search of images' defects by the means of technology CUDA (Compute Unified Device Architecture - the unified hardware-software decision for parallel calculations on GPU) companies NVIDIA is resulted. It is done the comparison of the temporary characteristics of performance of images' updating without application GPU and with use of opportunities of graphic processor GeForce 8800.

PACS: 89.80.+h, 89.70.+c, 01.10.Hx

1. INTRODUCTION

Monochrome images are turned out as a result of processing the data given by the sensor controls used for medicine, flaw detection, special cartography and etc. In this case the image is turned out on an output of X-ray camera used for the control of temperature parameters in the chamber of a high pressure press. Prominent feature of such images is presence of specific defects which are very bright points.

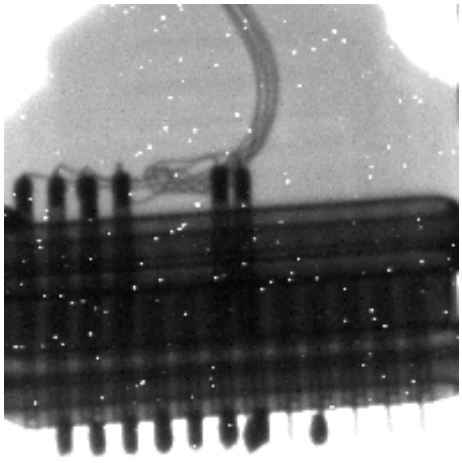

Fig.1. *A fragment of the original image with defects*

Such defects are a result from hitting of the absent-minded quantum of X-ray radiation directly on a matrix of the sensor control. In this case because of rather bigger size of the camera the voltage on the anode of a used source of radiation reaches 250 kilovolt. It leads to the occurrence of rather big quantity of quantum of the absent-minded radiation. Within the analysis of similar images it is necessary the application of the programs-filters removing the similar defects. It slows down the formation of images essentially (in several times). Acceleration of processing is necessary for improvement of the analysis of such images. Basic time when setting up of a window of visibility is occupied by the filtration of images. It is necessary for removing of such defects.

2. IMAGE FILTERING

Filters are based on the operation of convolution [1,3,4]. An operation of calculation of a new value of the chosen pixel is applied for the image of convolution. It considers the values of the pixels surrounding it. For this case the matrix (a kernel of convolution) by the size 3×3 is used:

P(i-1,j-1)	P(i,j-1)	P(i+1,j-1)
P(i-1,j)	P(i,j)	P(i+1,j)
P(i-1,j+1)	P(i,j+1)	P(i+1,j+1)

If we apply convolution to each pixel of the image the certain effect is turned out which depends on the chosen kernel of convolution. Generally there are two types of the filters on usage of pixels' values: recursive and non- recursive. In recursive filters the values calculated on the previous step are used for the calculation of the subsequent pixels' values. In non-recursive filters for calculations it is always used the original pixels' values. It is necessary to note that it is rather difficult to do paralleling of calculation for the recursive filters. It is impossible for CUDA. That is why the application of such filters is not considered here.

3. MEDIAN FILTER

For the filtration of impulsive disturbances (in this case they are bright points) median filters are usually used. It is one of a kind of digital filters which

*Corresponding author. E-mail address: dudnik@mail.ru

is widely used in digital processing of signals and images for the reduction of the noise level. It is based on finding of a median which is an average element of sequence. For this the value of readout inside of the window of the filter are sorted in ascending order (decrease). The value which is in the middle of the ordered list comes to the output of the filter. The window moves along filtered signal and calculations are repeated. In this way very bright values of elements of the image are eliminated and replaced by the values of similar magnitudes of brightness of the next elements.

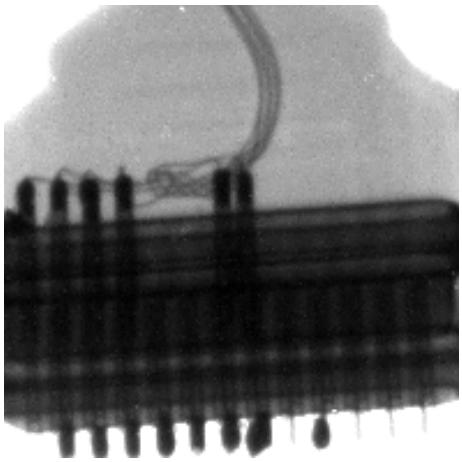

Fig.2. *A fragment of the filtered image with the removed defects*

Use of the graphic accelerators as a fast calculators allows to accelerate processing of the half-tone pictures which have size more of 10 000 elements. Realization of the programs of images processing in the form of DLL is recommended. It allows to use them for the various program's platforms. The gap in productivity between GPU and CPU is sharply reduced (it is approximately proportional to a quantity of kernels) with the usage of the multinuclear (2?6 kernels) processors and increasing of calculations' complexity for paralleling processes.

4. PROGRAM REALIZATION OF ALGORITHM

The algorithm has been realized by the means of system engineering CUDA in the form of Windows DLL. It provided a convenient reference to it from the programs created by the means of various software (for example: MS Visual Studio and Borland C ++ Builder). Moving of the data processing image to the GPU device is done during the reference to the program of filtration. Processing of the received data by the means of median filter and returning of the filtered image to the specified area of computer's memory is also carried out [2,5]. Below the fragment of the code which is carrying out described above operation is resulted:

```
 #include <stdio.h>
 #include <stdio.h>
 #include <assert.h>
 #include <assert.h>
 #include <cuda.h>
 #include <cuda.h>
 int main(void)
 int main(void)
{
// pointers to host memory
float *a_h, *b_h; float *a_h, *b_h; float *a_d, *b_d;
// pointers to device memory
float *a_d, *b_d; int i; int i;
// allocate arrays on host
a_h = (float *)malloc(sizeof(float)*N); a_h = (float
*)malloc(sizeof(float)*N); b_h = (float *)malloc(sizeof(float)*N);
b_h = (float *)malloc(sizeof(float)*N);
// allocate arrays on device
cudaMalloc((void **) &a_d, sizeof(float)*N); cudaMalloc((void **)
&a_d, sizeof(float)*N); cudaMalloc((void **) &b_d, sizeof(float)*N);
cudaMalloc((void **) &b_d, sizeof(float)*N);
// send data from host to device: a_h to a_d
cudaMemcpy(a_d, a_h, sizeof(float)*N, cudaMemcpyHostToDevice);
cudaMemcpy(a_d, a_h, sizeof(float)*N, cudaMemcpyHostToDevice);
// copy data within device: a_d to b_d
cudaMemcpy(b_d, a_d, sizeof(float)*N, cudaMemcpyDeviceToDevice);
cudaMemcpy(b_d, a_d, sizeof(float)*N, cudaMemcpyDeviceToDevice);
// Delete defects
MedianFilter(b_d);
// retrieve data from device: b_d to b_h
cudaMemcpy(b_h, b_d, sizeof(float)*N, cudaMemcpyDeviceToHost);
cudaMemcpy(b_h, b_d, sizeof(float)*N, cudaMemcpyDeviceToHost);
```

It is necessary to note that an overhead charge for the transfer and returning of a file of processing data, start of the process of processing occupy an essential part of the general operating time of the program. That is why an essential gain in time starts to be shown within the size of processing files of 20 thousand of elements and more. In this case the same algorithm has been realized with the help of the base means of parallel processing which is available within Windows. The comparison analysis of programs' work speed has shown that the usage of modern multinuclear processors with general purpose allows making a break in speed of processing unimpressive (2/5 times in comparison with 10/20 for the one-nuclear processor). On the other hand it has been noted by us that the increase in processing speed rather sharp decreases within increasing the quantity of kernels more than seven. (From our point of view it is due to increasing the overhead charge of Windows system for the management of quantization of time and switching of the processes, and as due to interference of the data in the buffer of the processor).

5. CONCLUSIONS

Use of the graphic accelerators as a fast calculators allows to accelerate processing of the half-tone pictures which have size more of 10 000 elements. Realization of the programs of images processing in the form of DLL is recommended. It allows to use them for the various program's platforms. The gap in productivity between GPU and CPU is sharply reduced (it is approximately proportional to a quantity of kernels) with the usage of the multinuclear (2?6 kernels) processors and increasing of calculations' complexity for paralleling processes.

References

1. William K. Pratt. *Digital Image Processing 3-rd Edition.* On 2011. Wiley-Interscience; 2 edition (April 1991).
2. Aleksej Berillo. (NVIDIA CUDA) *Non graphic calculation on graphic processors.* http://www.ixbt.com/video3/cuda-1.shtml
3. Kenneth R. Castleman. *Digital Image Processing.* Prentice Hall; 2nd edition (September 2, 1995).
4. Geoff Dougherty. *Digital Image Processing for Medical Applications.* Cambridge University Press; 1 edition (May 11, 2009).
5. Federico Dal Castello. *Advanced System Technology.* STMicroelectronics, Italy Douglas Miles, The Portland Group: Parallel Random Number Generation Using OpenMP, OpenCL and PGI Accelerator Directives. http://www.pgroup.com/lit/articles/insider/ v2n2a4.htm
6. Don Breazeal,Craig Toepfer: Tuning Application Performance Using Hardware Event Counters in the PGPROF Profiler http://www.pgroup.com/lit/articles/insider/ v2n4a3.htm en-us/File/ *larrabee-"manycore.pdf"*

ИСПОЛЬЗОВАНИЕ СРЕДСТВ GPGPU ДЛЯ РАЗРАБОТКИ ПРОГРАММ ПОИСКА ДЕФЕКТОВ МОНОХРОМНЫХ ПОЛУТОНОВЫХ ИЗОБРАЖЕНИЙ

В.А. Дудник, В.И. Кудрявцев, Т.М. Середа, С.А. Ус, М.В. Шестаков

Описано применение средств GPGPU для разработки программ поиска дефектов монохромных полутоновых изображений. Приведено описание реализации алгоритма поиска дефектов изображений средствами технологии CUDA (Compute Unified Device Architecture – унифицированного программно-аппаратного решения для параллельных вычислений на GPU) компании NVIDIA. Проведено сравнение временных характеристик выполнения корректировки изображений без применения GPU и с использованием возможностей графического процессора GeForce 8800.

ВИКОРИСТАННЯ ЗАСОБІВ GPGPU ДЛЯ РОЗРОБКИ ПРОГРАМ ПОШУКУ ДЕФЕКТІВ МОНОХРОМНИХ ПІВТОНОВИХ ЗОБРАЖЕНЬ

В.О. Дуднік, В.І. Кудрявцев, Т.М. Середа, С.О. Ус, М.В. Шестаков

Описано застосування засобів GPGPU для розробки програм пошуку дефектів монохромних півтонових зображень. Приведено опис реалізації алгоритму пошуку дефектів зображень засобами технології CUDA (Compute Unified Device Architecture – уніфікованого програмно-апаратного рішення для паралельних обчислень на GPU) компанії NVIDIA. Прведено порівняння тимчасових характеристик виконання коректування зображень без застосування GPU і з використанням можливостей графічного процесора GeForce 8800.