High-speed Parallel Feature Extraction Algorithm of Wind Tunnel Image Based on GPU

Zhengyu Zhang¹, Xiaochao Xu²*, and Hua Yan²

¹China Aerodynamics Research and Development Center, Mianyang 621000, China
²College of Electronics and information Engineering, Sichuan University, Chengdu 610065, China

*Corresponding author email: 1084891228@qq.com

Abstract. In wind tunnel test, the serial feature extraction algorithm of wind tunnel image based on CPU is too slow to meet the execution speed requirements of wind tunnel test. To solve this problem, a high-speed parallel feature extraction algorithm of wind tunnel image based on GPU is proposed. The proposed algorithm is optimized in parallel from the CUDA kernel level and CUDA stream level to speed up feature extraction execution. In image pre-processing, a pixel is processed by a CUDA thread to achieve parallelization. In feature extraction, the image segmentation is introduced to parallelize 8-connected boundary tracking algorithm. And a CUDA thread is used to process an image block to parallelize feature extraction process. Furthermore, the proposed algorithm uses CUDA stream to asynchronous data transmission and data processing to achieve CUDA stream level parallelism. To verify the efficiency and effectiveness of the GPU-based algorithm, comparative experiment between CPU and GPU is conducted. The experimental results show that the performance of GPU-based parallel algorithm is far better than that of CPU-based serial algorithm. The GPU-based parallel algorithm can greatly improve the execution speed of feature extraction while ensuring the accuracy of feature data.

1. Introduction

With the development of industrial aerodynamics, people pay more and more attention to wind tunnel test [1]. Wind tunnel test refers to an aerodynamic experimental method of placing a model of object in a wind tunnel to study the gas flow, interaction with the model and understand the aerodynamic characteristics of the object. The method called VM (Vedeogrammetric Measurement) [2, 3] is often used in wind tunnel test to measure the deformation and estimate the attitude of the target model. Firstly, one or more cameras are used to take the image of the marked points on the experimental model at the same time. And then the feature information of the mark points is extracted by the feature extraction algorithm so that we can measure the deformation and estimate the attitude of the target model. In the wind tunnel test, the experimental image is a gray-scale image containing a large number of white marked points, which are approximately circular irregular figures. The features extracted from the experimental image are perimeter, area and gray centroid of these white marked points. Figure 1 below shows an experimental sample whose resolution is 5120×5120. In the past few years, feature extraction algorithm based on CPU was often used. Due to the influence of CPU architecture, the algorithm can only be processed in a serialized or low concurrency way, which makes the image processing speed very limited. With the development of wind tunnel technology, the amount of wind tunnel test data is getting larger and larger. The processing speed of traditional feature extraction algorithm based on CPU has...
been unable to keep up with the speed of the camera's image collection. Therefore, it is particularly meaningful to introduce a new method to accelerate the feature extraction algorithm.

Figure 1. Experimental sample in wind tunnel test.

This paper studies the traditional feature extraction algorithm of wind tunnel image based on CPU and conducts the parallel analysis of it. By using GPU and CUDA programming model [4, 5, 6] to accelerate and optimize the traditional feature extraction algorithm, a high-speed parallel feature extraction algorithm of wind tunnel image based on GPU is propose.

2. Feature Extraction Algorithm of Wind Tunnel Images Based on CPU

The process of wind tunnel image feature extraction includes two steps: image preprocessing and feature extraction. Image preprocessing includes three steps: binarization, expansion and corrosion. The purpose of image preprocessing is to generate the contour map of marker points according to the grayscale image.

Binarization [7] is to transform grayscale image into binary image. Assuming that the threshold value of binarization is $T$, $f(x, y)$ is input and $g(x, y)$ is the output result of binarization, the binarization algorithm can be expressed as follows:

$$
g(x, y) = \begin{cases} 
0 & f(x, y) < T \\
255 & f(x, y) \geq T 
\end{cases}
$$

(1)

Then, edge detection is needed for the generated binary image to generate the contour map. Edge detection algorithm includes two steps: expansion and corrosion [8]. The expansion is to use 8-connected neighbourhood to check the internal hollow pixel of the marker points so that we can get regularizing contours of marked points. The expansion can be expressed as follows:

$$
e(x, y) = g(x, y) \lor g(i, j) \quad x - 1 \leq i \leq x + 1, \quad y - 1 \leq j \leq y + 1
$$

(2)

After the expansion, the corrosion uses a 4-domain cross core to extract the contour map of the marked points. The corrosion can be expressed as follows:

$$
c(x, y) = e(x, y) \lor e(i, j) \quad (i, j) \subseteq \{(x - 1, y), (x + 1, y), (x, y - 1), (x, y + 1)\}
$$

(3)

The feature extraction process includes two steps: 8-connected boundary tracking algorithm [9, 10] to extract bounding boxes and feature extraction.

8-connected boundary tracking algorithm is used to track the boundary information of the contour map. Its idea is to start from a certain boundary pixel point of the contour and search out other pixel points of the contour according to a certain "detection criterion" until returning to the starting boundary point, so as to track the target boundary. When tracking the boundary, the 8-connected boundary tracking
algorithm needs to set the tracking starting point, the initial tracking direction and the tracking direction rotation mode (clockwise or counter clockwise) [11]. In the tracking process, we save the maximum and minimum values of the pixels in the x and y directions to extract the bounding box of marked points. After getting the bounding box, we extract the area and gray centroid of the marked points on grayscale image according to the information of the bounding box. The area of marked points can be calculated by counting the number of pixels whose value is greater than the threshold value of binarization in the bounding box area of grayscale image. The area calculation formula is as follows.

\[
S = \begin{cases} 
S & f(x,y) < T \\ 
S + 1 & f(x,y) \geq T 
\end{cases} 
\quad (x,y) \subseteq \text{BoundingBox}
\]  

(4)

The gray centroid algorithm [12] is used to calculate the gray centroid of the marked points. The gray centroid algorithm divides the weighted sum of coordinates and pixel values by the sum of pixels in bounding box area. Assuming that \( w(x,y) \) represents pixel weight, \( f(x,y) \) represents pixel value, \( D \) represents calculation area, the detailed calculation formula is as follows.

\[
\bar{x} = \frac{\sum \sum i \cdot w(i,j)}{\sum \sum w(i,j)}, \quad \bar{y} = \frac{\sum \sum j \cdot w(i,j)}{\sum \sum w(i,j)}
\]

\[
w(x,y) = \begin{cases} 
0 & (x,y) \notin D \\ f(x,y) & (x,y) \in D 
\end{cases}
\]

(5)

3. Parallel Feature Extraction Algorithm of Wind Tunnel Images Based on GPU

The feature extraction algorithm based on CPU can only perform in a serial or low concurrent way. To improve the speed performance of the feature extraction algorithm, this paper adopts CUDA to design a parallel feature extraction algorithm of wind tunnel image based on GPU.

In image preprocessing, each pixel is processed in exactly the same way while the processing between pixels is completely irrelevant. Obviously, they are highly parallelizable. Therefore, when using GPU for image preprocessing, a CUDA thread is used to process a pixel correspondingly so that we can process image data in parallel. We configure 128 CUDA threads into one CUDA block which processes an image block composed of 128 pixels in a row. Figure 2 below shows the correspondence between the CUDA thread and the image processing unit.

![Figure 2](Image)

**Figure 2.** Correspondence between the CUDA thread and the image processing unit.

In feature extraction, the key to parallel feature extraction is parallel extraction of bounding box. After the bounding box extraction is completed, each bounding box is processed in the same way. According to the 8-connected boundary tracking algorithm, the key to parallel extraction of bounding box is to find the starting pixel point of the contour of the marker point in parallel. In this paper, a parallel feature extraction algorithm based on image segmentation is proposed. The algorithm divides the image into...
image blocks of the same size, and uses CUDA threads to parallelize image blocks at the same time. Figure 3 below shows the principle of parallel feature extraction.

![Figure 3](image)

**Figure 3.** The principle of parallel feature extraction.

Parallel 8-connected boundary tracking algorithm based on image segmentation divides image contour map into 16×16 image blocks. Each CUDA thread traverse the whole image block to find the starting pixel point of the marked point contour. When the starting pixel point of the contour is found, the bounding box of the marked points is extracted by the 8-connected boundary tracking algorithm. Figure 4 shows the principle of parallel extraction of bounding box based on image segmentation.

![Figure 4](image)

**Figure 4.** The principle of parallel extraction of bounding box based on image segmentation.

Parallel 8-connected boundary tracking algorithm may lead to the problem of data repetition of bounding box. When a contour of a marker point is divided into different image blocks, each image block will extract same bounding box. To solve this problem efficiently, we use preliminary screening and precise screening which are both based on the distribution characteristics of the repeated bounding box data in the GPU memory. Preliminary screening is to compare the current bounding box with that of its adjacent blocks which includes upper right blocks, right blocks, lower right blocks, and lower block. If the same data exists, the current bounding box will be discarded. Precise screening is to compare the current bounding box with the bounding boxes extracted from the image block in the lower half of its 9×9 neighbourhood. If the same data exists, the current bounding box will be discarded.

After getting the bounding box, each bounding box is used to extract the area and gray centroid of the marked point on grayscale image by a CUDA thread so that we can extract the feature of all marked points in parallel.

When using GPU to speed up the feature extraction process, the data transmission between CPU and GPU is very time-consuming. We can introduce CUDA stream [13] to solve this problem, improve the processing speed and realize the CUDA stream level parallelism. CUDA stream can asynchoronize data transmission and data processing, and different CUDA streams can be executed concurrently to improve the utilization efficiency of GPU. By creating a reasonable number of CUDA streams, we can hide the
transmission delay between GPU and CPU. Figure 5 below shows the principle of CUDA stream parallelism.

Figure 5. The principle of CUDA stream parallelism.

4. Design and Analysis of Experiment

In order to verify the effectiveness of the parallel feature extraction algorithm based on CUDA, repeated test experiments on four kinds of wind tunnel test images were conducted. Figure 6 below shows 4 kinds of sample images and their information.

Figure 5. Four kinds of sample images.

The experimental environment is a personal laptop (Dell Alienware area-51m) whose CPU is Intel(R) Core i7-9700K(CPU) and GPU is NVIDIA GeForce RTX 2070. The experiments are carried out on Visual studio 2017 and CUDA 10.0.

In order to verify the efficiency of parallel feature extraction algorithm based on GPU, we test the execution time when the number of the sample is 1, 10, 20, 50 and 100. And then we can calculate execution speed of different samples by Formula 6.

\[
\text{Speed} = \frac{(1+10+20+50+100) \times \text{imgsize}}{t_i + t_{10} + t_{20} + t_{50} + t_{100}}
\] (6)

Figure 6 shows the execution speed comparison between CPU and GPU. We can see from the figure 6, the execution speed of GPU is far faster than that of CPU. The average execution speed of GPU reaches 1524.14M/s and the average acceleration ratio of GPU to CPU is 12.71.
In order to explore whether the number of marked points in a sample affects the performance of the parallel feature extraction algorithm based on GPU, we conducted test experiments on sample 1 and sample 3. Firstly, we set the 1/2, 3/4, and 4/4 of image data to zero to generate samples containing different numbers of marked points as shown in Figure 7.

And then, we can get the time line chart shown in Figure 8 by testing the execution time of the pre-processing and feature extraction when the number of the samples is 100. It can be seen from Figure 8 that the relationship between the parallel feature extraction algorithm on the GPU and the number of marked points in the sample image can be described as follows:

1) The number of feature points in a sample has almost no effect on the image pre-processing, because no matter whether there are marked points or not, the corresponding preprocessing operations must be performed.

2) The number of feature points in sample image has an effect on the feature extraction part. The more marked points in a sample, the slower the execution speed of the feature extraction. Because more
marked points in a sample, more thread warps are needed in GPU, and the execution time of the feature extraction is longer.

![Figure 8](image_url)  
**Figure 8.** The Figure shows the time of the pre-processing and feature extraction when the number of the samples is 100. The test samples are shown in Figure 7.

In order to explore the effect of CUDA stream on the performance of the algorithm based on GPU, we test the execution speed of four sample images when the number of CUDA stream is one, two and five. Figure 9 below shows the execution speed with different number of CUDA streams.

![Figure 9](image_url)  
**Figure 9.** Execution speed of four sample images with different CUDA Streams.  
It can be seen from Figure 9 that CUDA stream can improve the performance of the algorithm based on GPU. Different samples have different optimization effect, which is related to the proportion of data transmission time in the image execution process. If the proportion of data transmission time is too low, the transmission delay that can be hidden accounts for a low proportion of the total time. And the performance of the algorithm can be improved a little. If the proportion of data transmission is too high, the feature extraction process takes a short time and the transmission delay cannot be completely hidden.
In order to verify the effectiveness of the algorithm based on GPU, we compare the feature data extracted by the CPU with the feature data extracted by the GPU. To calculate the number of feature points, mean square error of area, perimeter and gray centroid, we can get the algorithm validity test result as shown in Table 1. It can be seen that the GPU algorithm execution result is exactly the same as the CPU execution result, which can prove the effectiveness of the algorithm based on GPU.

### Table 1. The result of algorithm validity test.

| Sample  | The number of marked points (GPU) | The number of marked points (CPU) | mean square error |
|---------|-----------------------------------|-----------------------------------|------------------|
|         |                                   |                                   | area  | perimeter | gray centroid |
| Sample 1| 14828                             | 14825                             | 0     | 0         | 0             |
| Sample 2| 1618                              | 1618                              | 0     | 0         | 0             |
| Sample 3| 400                               | 400                               | 0     | 0         | 0             |
| Sample 4| 2187                              | 2187                              | 0     | 0         | 0             |

5. Conclusion

In this paper, the feature extraction algorithm of wind tunnel image is studied. Aiming at solving the problem of low performance of serial feature extraction algorithm based on CPU, a high-speed parallel feature extraction algorithm of wind tunnel image based on GPU is proposed. The feature extraction process is accelerated by CUDA kernel-level and CUDA stream-level parallelization. The experimental verification shows that the performance of high-speed parallel feature extraction algorithm of wind tunnel image based on GPU is far superior to that of the serial feature extraction algorithm based on CPU. The average processing speed of the algorithm based on GPU reaches 1534M/s, which is 12.73 times higher than that of the algorithm based on CPU.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (11872069).

References

[1] Birch A. Wind Tunnel Test [J]. Building Design, 2010, 28(1):33-39.
[2] Liu T, Burner A W, Jones T W, et al. Photogrammetric Techniques for Aerospace Applications[J]. Progress in Aerospace Sciences, 2012, 54(OCT.): 1-58.
[3] Barrows D. Videogrammetric Model Deformation Measurement Technique for Wind Tunnel Applications[C]// Aiaa Aerospace Sciences Meeting & Exhibit. 2007.
[4] Nvidia C. Nvidia CUDA Programming Guide[J]. Nvidia Corporation, 2011.120(18):8
[5] Shane Cook. CUDA Programming: A Developer's Guide to Parallel Computing with GPUs[M]. Elsevier, MK, 2012.
[6] Cheng, John, McKercher, Ty. Professional CUDA C Programming[J]. 2014.
[7] Gonzalez, Rafael C, Woods, Richard E. Digital Image Processing (3rd Edition) [M]. Prentice-Hall, Inc. 2007.
[8] Xin G, Yue H U, Wei D U, et al. Application of Decreasing Noise of Gray Image by Corrosion and Expansion Algorithm[J]. Journal of Beijing Institute of Graphic Communication, 2014.
[9] Liu X, Xiang J, Yang B. A Labeling Algorithm Based on 8-connected Boundary Tracking[J]. Computer Applications in Engineering Education, 2001, 23(2):125-126.
[10] Fu-Sheng W, Guo-Qing Q I. Boundary Tracking Algorithm of Objects in Binary Image[J]. Journal of Dalian Maritime University, 2006.
[11] Shi J, Guo C. The Extraction of Circle Contour Based on Improved Boundary Tracing Algorithm[C]// International Conference on Intelligent Human-machine Systems & Cybernetics. IEEE, 2012.
[12] W. Xian, M. Qinwei, M. Shaopeng, et al. A Marker Locating Method Based on Gray Centroid Algorithm and its Application to Displacement and Strain Measurement[C]// 2011 Fourth International Conference on Intelligent Computation Technology and Automation. IEEE, 2011
[13] Toledo L, Pea A J, Sandra Catalán, et al. Tasking in Accelerators: Performance Evaluation[C]// 2019 20th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT). IEEE, 2020.