2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

A GPU Based Parallel Genetic Algorithm for the Orientation Optimization Problem in 3D Printing*

Zhishuai Li[1,2], Gang Xiong[1,3], Xipeng Zhang[1], Zhen Shen[†,1,4], Can Luo[1], Xiuqin Shang[1], Xisong Dong[1,4], Gui-Bin Bian[1], Xiao Wang[1,4] and Fei-Yue Wang[1]

Abstract— **The choice of model orientation is a very important issue in Additive Manufacturing (AM). In this paper, the model orientation problem is formulated as a multi-objective optimization problem, aiming at minimizing the building time, the surface quality, and the supporting area. Then we convert the problem into a single-objective optimization in the linear-weighted way. After that, the Genetic Algorithm (GA) is used to solve the optimization problem and the process of GA is parallelized and implemented on GPU. Experimental results show that when dealing with complex models in AM, compared with CPU only implementation, the GPU based GA can speed up the process by about 50 times, which helps to significantly reduce the optimization time and ensure the quality of solutions. The GPU based parallel methods we proposed can help to reduce the execution time and improve the efficiency greatly, making the processes more efficient.**

Index Terms— **Orientation Optimization; GPU; parallel computing; genetic algorithm; Additive Manufacturing**

I. Introduction

In recent years, with the rise of intelligent manufacturing, 3D printing, which is almost the same with the Additive Manufacturing (AM), has attracted widespread attention from academia and industry [1]–[3]. Unlike traditional turning, milling, stamping, and die-making processes, the 3D printing uses a 3D model and builds products by stacking materials layer by layer.

The choice of model orientation is a very important issue in 3D printing. The orientation determines the height of printed model which is an important indicator affecting the printing speed. At the same time, It influences the structure and quantity of the support, and the deformation and surface roughness for printing model. Specific parameters and constraints of the layered manufacturing process must also be considered when selecting the orientation. At present, there are two major kinds of methods for studying the problem of model orientation while printing 3D models.

Heuristic method: Lan *et al.* optimized the surface quality of the model in stereolithography printing technology by selecting the minimum or maximum vertical surface area of the print direction from several preselected directions [4]. Masood *et al.* used the method of measuring volumetric error of the model to determine the orientation [5]–[7].

Optimization method: Pandey *et al.* used multi-objective genetic algorithm to solve the orientation optimization of the model, taking into account the mutual constraints of construction time while printing and average surface roughness [8]. Research on model orientation problems based on optimization algorithms is gradually emerging and has achieved effective refinements [9], [10].

To determine the best orientation of a model for majority of approaches is a very difficult and time consuming task. Therefore, how to optimize the target indicators in model orientation and use a faster method to reduce calculation burden is a meaningful study.

Intelligent optimization algorithms have been applied in many engineering practice fields and have proved to be practical and usable optimization methods for solving problems in complex systems in recent years [11]–[14]. Different from Pandey *et al.* [8], we convert the problem into a single-objective optimization problem in a linear-weighted way. The requirement for personalized configuration of optimization indicators can be satisfied, which is closer to the way human think and decide. Furthermore, we use a parallel genetic algorithm to solve the optimization problem of model orientation.

As the calculation tool of this paper, the Graphic Processing Unit (GPU) is used to solve the problem of high computational burden when optimizing, quantifying, and evaluating model orientation. GPUs have played an important role in scientific computing such as biopharmaceuticals, chemical analysis, traffic simulation, and fluid mechanics [15]–[19]. By designing appropriate parallel algorithms used on GPU, the calculation process can be significantly accelerated and the problem of heavy calculation burden in the model orientation evaluation process can be effectively solved.

The remaining parts of this paper are organized as follows. In Section II, the problem formulation is presented, which aims at minimizing the building time, surface quality and supporting area. In Section III we further propose a problem solving method based on the genetic algorithm for the model orientation problem. In Section IV, the process of the genetic

*This work was supported in part by the National Natural Science Foundation of China under Grants 61773382, 61872365, 61533019 and 61702519; Beijing Natural Science Foundation (No. 4182065); Dongguan's Innovation Talents Project (Gang Xiong); the 2018-2019 joint project between Beijing Ten Dimensions Technology Co. Ltd. and Institute of Automation, Chinese Academy of Sciences; Chinese Guangdong's S&T project (2017B090912001); 2017 Special Cooperative Project of Hubei Province and Chinese Academy of Sciences.

[1]All the authors are with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China.

[2]Zhishuai Li is also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, 100049, China.

[3]Gang Xiong is also with the Cloud Computing Center, Chinese Academy of Sciences, Dongguan, 523808, China.

[4]Zhen Shen, Xisong Dong and Xiao Wang are also with the Qingdao Academy of Intelligent Industries, Qingdao, 266113, China.

[†]Corresponding Author. E-mail: zhen.shen@ia.ac.cn

algorithm is parallelized and implemented on the GPU to accelerate the problem solving and the evaluation in model orientation. In Section V, we present in detail experimental verification process by using NVIDIA® GPU and CUDA 9.0, and give the experimental results and related analysis. In Section VI, we summarize this paper.

II. Model orientation problem modeling

A. The description of model orientation problem

Due to the properties of 3D printed layered manufacturing, the rotation of the model around the z-axis has no effect on the forming process. Therefore, considering the orientation of the model, it is only necessary to consider the rotation angle θ_x of the model around x-axis and θ_y around y-axis.

In three-dimensional Cartesian coordinate system, the rotation matrix of the model is rotated by θ_x degrees around the x-axis as shown in (1).

$$\mathcal{R}_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \tag{1}$$

θ_x is in the same direction as the right-handed spiral. Similarly, the rotation matrix of the model is rotated by θ_y degrees around the y-axis as shown in formula (2).

$$\mathcal{R}_y(\theta_y) = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \tag{2}$$

Furthermore, we can get the model to rotate θ_x around x-axis and then θ_y around y-axis. The rotation matrix $\mathcal{M}$ for the model is (3).

$$\begin{aligned} \mathcal{M} &= \mathcal{R}_y(\theta_y)\mathcal{R}_x(\theta_x) \\ &= \begin{bmatrix} \cos\theta_y & \sin\theta_x\sin\theta_y & \sin\theta_y\cos\theta_x \\ 0 & \cos\theta_x & -\sin\theta_x \\ -\sin\theta_y & \cos\theta_y\sin\theta_x & \cos\theta_y\cos\theta_x \end{bmatrix} = \begin{bmatrix} \boldsymbol{m_1} \\ \boldsymbol{m_2} \\ \boldsymbol{m_3} \end{bmatrix} \end{aligned} \tag{3}$$

In this paper, we assume that the initial orientation $\boldsymbol{d}_0$ as the positive direction of the z-axis, i.e. $\boldsymbol{d}_0 = [0\ 0\ 1]^{\mathrm{T}}$, and the orientation of the model after rotation is $\boldsymbol{d}$.

$$\boldsymbol{d} = \mathcal{M}\boldsymbol{d}_0 = [\sin\theta_y\cos\theta_x \quad -\sin\theta_x \quad \cos\theta_y\cos\theta_x]^{\mathrm{T}} \tag{4}$$

The problem of model orientation optimization was systematically studied in [8], [20]. Here, we study the Stereo Lithography (STL) model orientation to the optimization problem combined with [9] and select three key factors, including model building time, surface quality and supporting area as optimization indicators, which affect 3D printing efficiency and finished product quality.

B. The relationship between model orientation and building time

The orientation of the model will determine the height of the model while printing, which in turn affects the building time. If the triangular surface of the STL model is used to obtain the height of the model, a large number of redundant calculations will occur, reducing the efficiency of the optimization algorithm. Here, we model the STL model as a set of vertices.

$$\begin{gathered} T = \{\boldsymbol{v}_j, j = 1, 2, ..., M\} \\ \boldsymbol{v}_j = \{[x\ y\ z]^T | x, y, z \in \mathbb{R}\} \end{gathered}$$

M represents the total number of vertices of the model. $\mathbb{R}$ is the set of real number.

Our work is based on the 3D printing technology represented by Digital Light Processing (DLP), in which the printing time is proportional to the height of the model, so the relationship between the building time and the orientation of the model is shown in (5).

$$f_1(\theta_x, \theta_y) = \frac{1}{\max\{\boldsymbol{m_3} \cdot \boldsymbol{v}_j\} - \min\{\boldsymbol{m_3} \cdot \boldsymbol{v}_i\}} \tag{5}$$

Where $\boldsymbol{v}_j$ and $\boldsymbol{v}_i$ mean the coordinates of vertices i, j and we find the maximum and minimum z values under the rotation by $\mathcal{M}$, respectively.

C. The relationship between model orientation and surface quality

The step effect is the main factor affecting the surface quality of the model. It causes a volumetric error between the target model and printed model, which in turn affects the surface quality of the printed model. But the volumetric error can be reduced by adjusting the orientation of the model, and the balance between the printing time and the surface quality of the model can be obtained. As shown in Fig. 1, for the printing of a simple three-dimensional object-triangular prism, the volumetric error caused by different orientations is also different, which causes the surface quality of the molded part to be different.

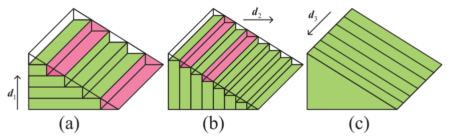

Fig. 1: Various directions lead to different surface quality

The influence of the step effect is a widely studied problem in 3D printing [8], [9], [21], [22]. We adopt a widely used step effect modeling method and quantify the relationship between step effect and the orientation, which uses the volumetric error between the STL model and printed model to quantify the influence of the step effect. It can be approximated as the volumetric error between all triangular facets of STL model and the surface of printed model, as shown in (6).

$$V = \sum_{i=1}^{n} V_i \tag{6}$$

Where V is the volumetric error between the STL model and the printed model, V_i is the volumetric error between the triangular facets t_i and the printed model, and n is the total number of triangular facets.

The volumetric error between the triangular facets and printed model can be regarded as the sum of several small volumetric errors ΔV, and each ΔV can be approximated as a triangular prism, so there is (7).

$$V_i = \sum \Delta V_i \tag{7}$$

$$\Delta V_i = \frac{1}{2} \times \Delta S_i \times h \tag{8}$$

Where ΔS is the area where triangular facets is sandwiched between the layer heights u, and h is the height corresponding to the bottom surface ΔS of the triangular prism. (9) and (10) can be available from the two-dimensional simplified diagram in Fig. 2. $\boldsymbol{n}_i$ is the normal vector of facet t_i.

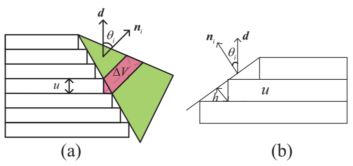

Fig. 2: Volumetric error in triangular facets and its two-dimensional simplified diagram

$$h = u \times |\cos\theta_i| \tag{9}$$

$$\cos\theta_i = \frac{\boldsymbol{d} \cdot \boldsymbol{n}_i}{|\boldsymbol{d}| \times |\boldsymbol{n}_i|} \tag{10}$$

Substituting (8) and (9) into (7), we can obtain (11).

$$\begin{aligned} V_i &= \sum \frac{1}{2} \times u \times |\cos\theta_i| \times \Delta S_i \\ &= \frac{1}{2} \times u \times |\cos\theta_i| \times \sum \Delta S_i \\ &= \frac{1}{2} \times u \times |\cos\theta_i| \times S_i \end{aligned} \tag{11}$$

Where S_i is the area of the triangular facets t_i, and the Cartesian coordinates of the three vertices in triangular facets t_i are $A(x_1, y_1, x_1), B(x_2, y_2, z_2), C(x_3, y_3, z_3)$, then S_i can be calculated by (12).

$$\begin{aligned} S_i &= \frac{1}{2} \left| \overrightarrow{AB} \times \overrightarrow{AC} \right| \\ &= \frac{1}{2} \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix} \end{aligned} \tag{12}$$

From (6), (10) and (11), we can get the volumetric error V caused by the step effect of the STL model under the orientation $\boldsymbol{d}$ as (13).

$$V = \frac{1}{2} \times u \times \sum_{i=1}^{n} \frac{|\boldsymbol{d} \cdot \boldsymbol{n}_i|}{|\boldsymbol{d}| \times |\boldsymbol{n}_i|} \times S_i \tag{13}$$

When both $\boldsymbol{d}$ and $\boldsymbol{n}_i$ are unit vectors, (13) can be further simplified to (14).

$$V = \frac{1}{2} \times u \times \sum_{i=1}^{n} |\boldsymbol{d} \cdot \boldsymbol{n}_i| \times S_i \tag{14}$$

Then the relationship between surface quality and orientation for the model can be obtained as (15).

$$f_2(\theta_x, \theta_y) = \sum_{i=1}^{n} |\boldsymbol{d} \cdot \boldsymbol{n}_i| \times S_i \times \delta_{1,i} \tag{15}$$

Where

$$\delta_{1,i} = \begin{cases} 0, & \arccos(\boldsymbol{d} \cdot \boldsymbol{n}_i) = 0, \pi. \\ 1, & else. \end{cases} \tag{16}$$

In (16), $\delta_{1,i}$ is a threshold function of the volumetric error, indicating that volumetric error is 0 when the normal vector of facet $\boldsymbol{n}_i$ is the same or opposite to the orientation.

D. The relationship between model orientation and supporting area

The influence of model orientation on designing support is enormous, and different model placements will result in completely different support structures and shaping effects. In STL model, supporting areas can be judged according to the orientation of the triangular facets. As (17) shows.

$$\delta_{2,i} = \begin{cases} 1, & \arccos(\boldsymbol{d} \cdot \boldsymbol{n}_i) > \alpha. \\ 0, & \arccos(\boldsymbol{d} \cdot \boldsymbol{n}_i) \leq \alpha. \end{cases} \tag{17}$$

Here α is the critical angle in the support generation, indicating that when the angle between the normal vector of facet and the printing direction is greater than α, it is necessary to add support to the triangular facet.

The contacting area between the triangular facets t_i and the support in the model can be measured by the projected area of the triangular facets to the working platform, that is:

$$S_{i\perp} = S_i \times |\boldsymbol{d} \cdot \boldsymbol{n}_i| \times \delta_{2,i} \tag{18}$$

Further, it can be obtained that the total contacting area between the model and support is the sum of the contacting areas supported by all the triangular facets, as (19) shows.

$$S_T = \sum_{i=1}^{n} S_i \times |\boldsymbol{d} \cdot \boldsymbol{n}_i| \times \delta_{2,i} \tag{19}$$

The goal of optimization is to find an orientation $\boldsymbol{d}$ that minimizes the amount of support required to print the finished model. So the indicator between model orientation and supporting area is (20).

$$f_3(\theta_x, \theta_y) = \sum_{i=1}^{n} S_i \times |\boldsymbol{d} \cdot \boldsymbol{n}_i| \times \delta_{2,i} \tag{20}$$

III. Solving Model Orientation Problem by using Genetic Algorithm

Using GA to find the best orientation was proposed in [8], considering about surface roughness and building time. Different from them, we concern supporting area as well. Furthermore, we choose to convert the problem from a multi-objective optimization problem into a single-objective optimization problem and accelerate the algorithm by using GPU. It is necessary to re-describe the problem as (21).

$$\begin{aligned} &\min \ \textstyle\sum_{i=1}^{3} w_i s_i \\ &s.t. \ \theta_x, \theta_y \in [0, 2\pi) \end{aligned} \tag{21}$$

Where w_i is the weight of the optimization indicator s_i respectively, and there is (22).

$$\sum_{i=1}^{3} w_i = 1,\ w_i \geq 0 \tag{22}$$

While $s_i, i = 1,2,3$ are the normalized forms of building time f_1, surface quality f_2 and supporting area f_3, which are obtained by previous section, and the normalized formula is:

$$s_i(\theta_x,\theta_y) = \frac{f_i(\theta_x,\theta_y) - \min f_i(\theta_x,\theta_y)}{\max f_i(\theta_x,\theta_y) - \min f_i(\theta_x,\theta_y)},\ i = 1,2,3 \tag{23}$$

Since we need to obtain the maximum value for each optimization indicator, the fitness function of each optimization indicator is as (24).

$$F_i(\theta_x,\theta_y) = 1 - s_i(\theta_x,\theta_y),\ i = 1,2,3 \tag{24}$$

So the comprehensive fitness function $F(\theta_x,\theta_y)$ can be writen as (25).

$$\begin{aligned} F(\theta_x,\theta_y) &= \sum_{i=1}^{3} w_i \times F_i(\theta_x,\theta_y) \\ &= 1 - \sum_{i=1}^{3} w_i \times s_i(\theta_x,\theta_y),\ i = 1,2,3 \end{aligned} \tag{25}$$

Similar to [18], [23], the specific steps of the GA used in our work are as Algorithm 1 shows, which are in favor of parallelization on GPU.

Algorithm 1 The Pseudo-code of **GA**

```
Input: Vertices and triangular facets of the STL model, the population of model orientation schemes $P$, fitness function $F$, maximum number of iterations $N$, crossover probability $p_c$ and mutation probability $p_m$.
output: The best orientation in the last population $\theta_x,\ \theta_y$.
1: Generate initial population of the orientation schemes $P$;
2: Calculate the fitness of all orientation schemes in $P$ by using the evaluation indicators calculation formula;
3: while "NOT yet converged" do
4: For the orientation scheme $I_i$ ($i$ is the index of the individual $I_i$ in $P$) in population $P$, we randomly select another orientation $I_j$ in $P$, where $i \neq j$. The crossover operator is applied to $I_i$ and $I_j$ on the basis of satisfying $p_c$, and the generated child individual $I_i'$ is added to the child generation population $Q$;
5: For each orientation scheme $I_i'$ in $Q$, the mutation operator is applied to $I_i'$ based on the probability $p_m$;
6: Calculate the fitness of all orientation schemes in $Q$;
7: Compare the fitness function $F$ of each orientation scheme $I_i$ in $P$ and $I_i'$ in $Q$. If there is $F(I_i') > F(I_i)$, replace $I_i$ with $I_i'$ to write the $i$ position in $P$, otherwise retain $I_i$;
8: end while
9: return The orientation scheme $(\theta_x,\theta_y)$ with the highest fitness function in current population $P$.
```

GA can usually find a satisfactory solution in a reasonable time. However, as the models become more complex, the burden of computation and solution time are rapidly increasing. Therefore, it is very important to parallel the GA to improve its efficiency.

IV. Parallel Implementation of the GA on GPU

As the parallel implementation of the genetic algorithm needs to be synchronized in each iteration, the main computational node (CPU) is required to control the iterative process of the GA and the convergence condition judgment. In our work, the master-slave fine-grained hybrid model is used to realize the parallelization of the GA. The CPU is responsible for optimizing the module, mainly controlling the execution flow of the GA and performing the operation of selecting cross-compilation. The GPU is responsible for evaluating the module and calculating in parallel by taking the individual as the computing unit. Many evolutionary algorithms were implemented in parallel with the emergence and development of GPU [24], [25]. There is also a growing interest in GPU-Based parallel GA [18], [23]. It has been used in such areas as traffic signal control [26], dynamic subarea division [27]. We apply it to the parallel optimization for model orientation problem. The principle of parallel computing experiment in our work is shown in Fig. 3.

The evaluation of the building time while printing is based on the vertices of STL model as the basic unit of operation, while the evaluation of the surface quality and supporting area are based on the triangular facets of the STL model (Section II), so the two parts of the calculation need to be placed in different kernel functions. It can significantly reduce the computing burden of CPU terminal and improve the realization efficiency of the algorithm by using GPU to evaluate the model orientation schemes. The GPU can effectively shorten the running time of the scheme and improve the calculation efficiency of the evaluation module by

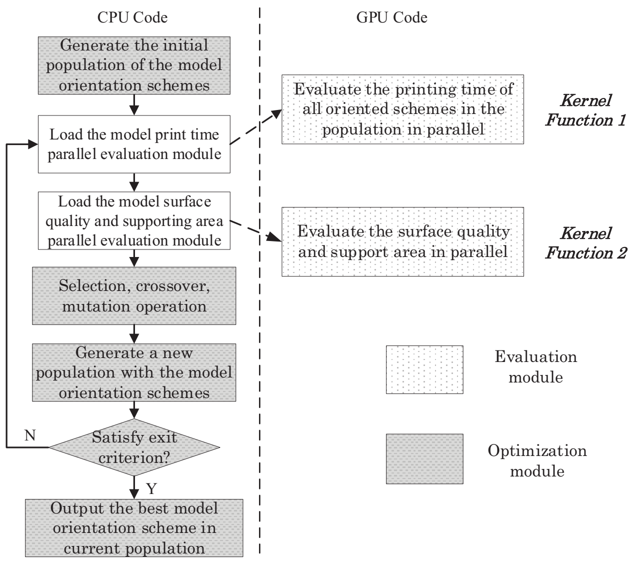
CPU Code
GPU Code
Generate the initial population of the model orientation schemes
Load the model print time parallel evaluation module
Evaluate the printing time of all oriented schemes in the population in parallel
Kernel Function 1
Load the model surface quality and supporting area parallel evaluation module
Evaluate the surface quality and support area in parallel
Kernel Function 2
Selection, crossover, mutation operation
Generate a new population with the model orientation schemes
Satisfy exit criterion?
N
Y
Output the best model orientation scheme in current population
Evaluation module
Optimization module

Fig. 3: Flow chart of GPU based parallel GA

effectively decomposing the evaluation tasks and designing a reasonable parallelization method.

It can be seen from (5)(15)(20) that if the operations of product and point multiplication result are regarded as a whole, the calculation of the printing time optimization indicator involves taking the maximum and minimum operations for a group of elements, while calculating the surface quality and supporting area optimization indicators is the cumulative sum of a set of elements. This computational process is called "reduction" operation, which refers to the process of reducing the number in a set to one number by using a binary operation with two inputs and one output.

The GPU version of the reduction operation plays an important role in many GPU scientific computing fields [28]–[30]. The calculation of indicators in our optimization problem is the general case of the reduction operation. However, considering there are usually hundreds of thousands or even millions elements (vertices or facets in the STL model) involved in the operation, it may be necessary to perform multiple reduction operations to obtain the final optimization indicators value because of the limitation of the amount of threads in a block. After several reduction operations, the atomic operation plan and the parallel "reduction" plan must be weighed to select the most efficient solution because of the decreasing of participating reduction elements.

V. Experiment and result analysis

A. Hardware configuration and programming environments

This experiment uses a tower-type high-performance GPU desktop server as a parallel computing experimental platform. The server includes a six-core *Intel Xeon E5-2620* processor running at 2.0 GHz, 128G DDR3 ECC memory, and a NVIDIA$_{®}$ *TITAN Xp* GPU connected via PCIe 16x bus. In terms of software, the server is equipped with the *Windows Server 2012 R2* operating system and the NVIDIA graphics driver with version number 375.66. The server uses *Visual Studio 2013 professional* for CPU code compilation and NVIDIA *CUDA 9.0* for GPU code compilation.

B. Experiment design

Before the experiment, we first introduce the test models used in this experiment. We have selected 5 models that

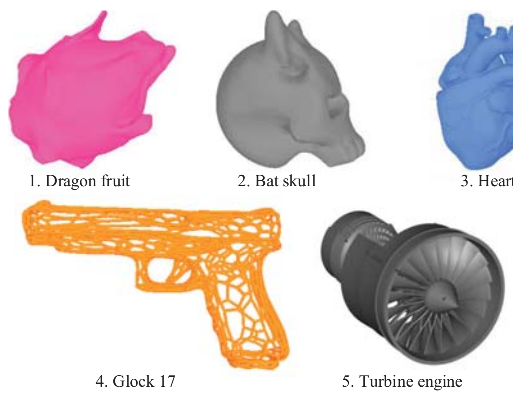

Fig. 4: Models used in our experiment

TABLE I: Parameters of models used in our experiment

Number	Model name	Vertex number	Facet number
1	Dragon fruit	44 k	88 k
2	Bat skull	92 k	185 k
3	Heart	108 k	217 k
4	Glock 17	1044 k	2090 k
5	Turbine engine	1527 k	3059 k

are difficult to directly determine the orientation as shown in Fig. 4. The specific information of the model is shown in TABLE I. These models are available for free from the Pinshape and Print Tiger online 3D printing communities [31], [32]. In order to understand the difference between the best orientation of different models, we "align" these models by determining the enveloping cuboid under the initial orientation of the models and moving the model to the place where the center of these enveloping cuboid overlapped with the origin of coordinates, the model vertex coordinate translation formula is,

$$\boldsymbol{v}_i' = \boldsymbol{v}_i - [\frac{x_{min}+x_{max}}{2} \frac{y_{min}+y_{max}}{2} \frac{z_{min}+z_{max}}{2}] \quad (26)$$

Where $\boldsymbol{v}_i'$ is the vertex coordinates after the model transformation, $\boldsymbol{v}_i$ is the vertex coordinates before the model translation, and x_{min}, x_{max}, y_{min}, y_{max}, z_{min} and z_{max} are coordinate minimum and maximum values respectively on the x, y and z axis before the translation of models.

The GA used in our experiment has a population size of 100 individuals, and the number of iteration calculations which is used as the termination criterion of GA is 200. Due to the evolutionary strategy of parent-child competition in our experiment, when the child is exactly the same as the parent, repeated evaluation of the feasible solution will occur, resulting in redundant calculation. Therefore, in order to improve the computational efficiency, we set the crossover probability p_c=1 to ensure that the offspring individuals are as different as the parent individuals, and the mutation probability is 0.1 in this experiment. The optimization variables θ_x and θ_y are encoded and decoded by binary code. Each optimization variable is represented by a 10-bit binary code, so the minimum of angle error between θ_x in the feasible solution is about 0.351, and θ_y is also the same.

In thread configuration of parallel implementation of GA on GPU, the size of block is set to $BlockSize = 1,024$ in our experiment, and two reductions are used to calculate the optimization indicators. Specifically, under this block size setting, if each thread performs the specification of one element, the two reductions can collectively deal with $BlockSize \times BlockSize$ elements. Assume that the number of elements participating in the optimization indicators calculation is N. When there is N<$BlockSize \times BlockSize$, the calculation of the optimization indicators can be calculated by the first N threads participating in reduction, and each thread can reduce one element. When there is

TABLE II: The comparison of orientation angle and fitness between serial and parallel operation results

Model number	$\theta_x(/^\circ)$		$\theta_y(/^\circ)$		F_1		F_2		F_3		F	
	Serial	Parallel	Serial	Parallel	Serial	Parallel	Serial	Parallel	Serial	Parallel	Serial	Parallel
1	140.6	135.7	24.3	30.4	0.99	0.94	0.96	0.98	0.96	0.98	0.97	0.97
2	10.5	10.2	0	359.6	0.71	0.73	0.97	0.97	0.97	0.97	0.88	0.89
3	46.4	44.3	25.3	26.7	0.85	0.86	0.92	0.89	0.92	0.88	0.90	0.88
4	1.4	1.4	270.0	271.4	0.12	0.12	1.0	0.99	1.0	0.99	0.71	0.70
5	270.0	272.5	86.5	86.3	1.0	0.99	0.99	0.99	0.99	0.99	0.99	0.99

$N > BlockSize \times BlockSize$, each thread calculates the reduction of k elements, and the reduction of $k+1$ elements is calculated by the former $N-(BlockSize \times BlockSize \times k)$ threads. Where k is defined by (27).

$$k = \lfloor \frac{N}{BlockSize \times BlockSize} \rfloor \tag{27}$$

Where $\lfloor \cdot \rfloor$ represents a floor function. In the calculation of building time optimization indicator, N is the number of vertices in printed model. In the calculation of the model surface quality and the supporting area optimization indicator, N is the number of facets in printed model.

We normalize each optimization indicator according to (23), and set the weights of the building time, the model surface quality and the supporting area as $w_1 = 0.33, w_2 = 0.33, w_3 = 0.34$.

C. Result analysis

The variation curve of comprehensive fitness $F(\theta_x, \theta_y)$ calculated by (25) in the iterative process of each model genetic algorithm are shown in Fig. 5, and the optimization results are shown in TABLE II.

It can be seen from Fig. 5 that the GA set the weights of in a few iterations, and has certain performance advantages in solving the problem. Taking advantage of the parallel acceleration optimization on GPU, the parallel GA method for the model orientation problem is performed. We have timed the evaluation of orientation in a single iteration. TABLE III shows the consumed computational time of five different models and speedup ratio. The results in table are obtained from the average of 20 independent executions.

TABLE III: Comparison between Serial and Parallel implementation

Model Number	Serial implementation(/s)	Parallel implementation(/s)	Speedup ratio
1	2.147	0.050	42.9
2	4.084	0.094	43.4
3	5.283	0.112	47.1
4	50.413	0.876	**57.5**
5	70.021	1.108	**63.2**

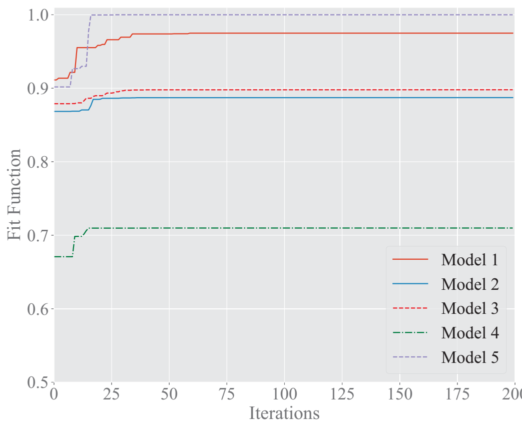

Fig. 5: The change of comprehensive fitness for each model

Compared with the CPU implementation, the GPU based parallel method can achieve a speedup ratio of about 50 times on complex models, which effectively reduces the solution time of the model orientation optimization problem and improves the solution efficiency. It also shows that as the model becomes more and more complex, the speedup ratio increases, which further validates the effectiveness of the GPU parallel implementation method of the proposed GA in our orientation optimization problem.

VI. Conclusion

In this paper, we propose to use GPU based parallel genetic algorithms to solve the multi-objective optimization problem of the 3D model orientation. Based on the mathematical relationship between model orientation and optimization indicators such as the building time, the surface quality and the supporting area, we present in detail how to formulate the problem and convert it to a single-objective optimization problem. The problem solving method of GA is implemented on the GPU in parallel. The experimental results show that when dealing with complex models, compared with CPU only implementation, the GPU based parallel GA can speed up the process by about 50 times. By GPU based methods, we can have a better solution within the same time.

References

[1] G. Xiong and X. Shang, "Intelligent manufacturing for personalized product: Upgrade from mass customization to social manufacturing," *Automation Panorama*, vol. 33, no. z1, 2016.

[2] J. Karjalainen and G. Xiong, "Social manufacturing and business model innovation," in *IEEE International Conference on Service Operations and Logistics, and Informatics*, 2016, pp. 18–23.

[3] F.-Y. Wang, D. Zeng, Z. Shen, S. Li, Y. Zhao, H. Gao, *et al.*, "Cloud computing-based additive manufacturing resource scheduling system and corresponding methods," China Patent CN103 414 792A, 2013.

[4] P. Lan, S. Chou, L. Chen, and D. Gemmill, "Determining fabrication orientations for rapid prototyping with stereolithography apparatus," *Computer-Aided Design*, vol. 29, no. 1, pp. 53–62, 1997.

[5] W. Rattanawong, S. Masood, and P. Iovenitti, "A volumetric approach to part-build orientations in rapid prototyping," *Journal of Materials Processing Technology*, vol. 119, no. 1-3, pp. 348–353, 2001.

[6] S. Masood, W. Rattanawong, and P. Iovenitti, "Part build orientations based on volumetric error in fused deposition modelling," *The International Journal of Advanced Manufacturing Technology*, vol. 16, no. 3, pp. 162–168, 2000.

[7] S. Masood, W. Rattanawong, and P. lovenitti, "A generic algorithm for a best part orientation system for complex parts in rapid prototyping," *Journal of materials processing technology*, vol. 139, no. 1-3, pp. 110–116, 2003.

[8] P. M. Pandey, K. Thrimurthulu, and N. V. Reddy*, "Optimal part deposition orientation in FDM by using a multicriteria genetic algorithm," *International Journal of Production Research*, vol. 42, no. 19, pp. 4069–4089, 2004.

[9] J. Zhao, L. He, W. Liu, and H. Bian, "Optimization of part-building orientation for rapid prototyping manufacturing," *Journal of Computer-Aided Design & Computer Graphics*, vol. 18, no. 3, pp. 456–463, 2006.

[10] H. S. Byun and K. H. Lee, "Determination of the optimal build direction for different rapid prototyping processes using multi-criterion decision making," *Robotics and Computer-Integrated Manufacturing*, vol. 22, no. 1, pp. 69–80, 2006.

[11] J. Y. Jung, G. Blau, J. F. Pekny, G. V. Reklaitis, and D. Eversdyk, "A simulation based optimization approach to supply chain management under demand uncertainty," *Computers & Chemical Engineering*, vol. 28, no. 10, pp. 2087–2106, 2004.

[12] F. Yu, H. Liu, and J. Dai, "Grey particle swarm algorithm for multi-objective optimization problems," *Computer Applications*, vol. 26, no. 12, pp. 2950–2952, 2006.

[13] L. Wang, H. Wu, F. Tang, D. Zheng, and Y. Jin, "Hybrid quantum genetic algorithms and performance analysis," *Control and Decision*, vol. 20, no. 2, pp. 156–160, 2005.

[14] W. Zhu, L. Jia, and X. Wu, "The control of the urban main road traffic flows based on multi-objective optimization," *Journal of Shandong University*, vol. 34, no. 3, pp. 72–76, 2004.

[15] T. Narumi, R. Sakamaki, S. Kameoka, and K. Yasuoka, "Overheads in accelerating molecular dynamics simulations with GPUs," in *Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies*. Otago, New Zealand: IEEE, 2008, pp. 143–150.

[16] J. Tölke, "Implementation of a Lattice Boltzmann kernel using the Compute Unified Device Architecture developed by nVIDIA," *Computing and Visualization in Science*, vol. 13, no. 1, p. 29, 2010.

[17] A. Benso, S. Di Carlo, G. Politano, and A. Savino, "GPU acceleration for statistical gene classification," in *Automation Quality and Testing Robotics (AQTR), 2010 IEEE International Conference on*, vol. 2, IEEE. Cluj-Napoca, Romania: IEEE, 2010, pp. 1–6.

[18] S. Tsutsui and N. Fujimoto, "Solving quadratic assignment problems by genetic algorithms with GPU computation: a case study," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, ACM. Montreal, Canada: ACM, 2009, pp. 2523–2530.

[19] Z. Shen, K. Wang, and F. Zhu, "Agent-based traffic simulation and traffic signal timing optimization with gpu," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE. Washington, DC, USA: IEEE, 2011, pp. 145–150.

[20] D. T. Pham, S. S. Dimov, and R. S. Gault, "Part orientation in stereolithography," *International Journal of Advanced Manufacturing Technology*, vol. 15, no. 9, pp. 674–682, 1999.

[21] Y. Gong, C. Chen, M. Xia, and E. Song, "Step effect analysis of FDM 3D printing model surface," *Manufacturing Technology and Machine Tools*, no. 4, pp. 27–30, 2016.

[22] G. Li, "Effect of fdm rapid prototyping process parameter on step effect," *Mechanical engineering & Automation*, no. 6, pp. 131–132, 2017.

[23] R. Arora, R. Tulshyan, and K. Deb, "Parallelization of binary and real-coded genetic algorithms on gpu using cuda," in *Evolutionary Computation*, 2010.

[24] G. Wilson and W. Banzhaf, "Deployment of cpu and gpu-based genetic programming on heterogeneous devices," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. New York, NY, USA: ACM, 2009, pp. 2531–2538.

[25] M. L. Wong and T.-T. Wong, "Implementation of parallel genetic algorithms on graphics processing units," *Studies in Computational Intelligence*, vol. 187, pp. 197–216, 2009.

[26] K. Wang and Z. Shen, "A gpu-based parallel genetic algorithm for generating daily activity plans," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1474–1480, 2012.

[27] Z. Shen, K. Wang, F.-Y. Wang, and C. L. P. Chen, "Gpu based genetic algorithms for the dynamic sub-area division problem of the transportation system," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 5115–5120, 2014.

[28] S. Tomov, R. Nath, and J. Dongarra, "Accelerating the reduction to upper Hessenberg, tridiagonal, and bidiagonal forms through hybrid GPU-based computing," *Parallel Computing*, vol. 36, no. 12, pp. 645–654, 2010.

[29] Y.-H. Kim and S.-K. Lee, "Fast GPU implementation for the solution of tridiagonal matrix systems," *Journal of KIISE: Computer Systems and Theory*, vol. 32, no. 11_12, pp. 692–704, 2005.

[30] G. Franceschetti, R. Guida, A. Iodice, D. Riccio, and G. Ruello, "Efficient simulation of hybrid strip-map/spotlight SAR raw signals from extended scenes," *IEEE Transactions on Geoscience & Remote Sensing*, vol. 42, no. 11, pp. 2385–2396, 2004.

[31] P. Tiger. (2018) 3D printing community and marketplace. [Online]. Available: http://www.dayinhu.com/

[32] Pinshape. (2013) 3D printing community and marketplace. [Online]. Available: https://pinshape.com/