A Statistics Based Prediction Method for Rendering Application

Qian Li(✉), Weiguo Wu, Long Xu, Jianhang Huang, and Mingxia Feng

Department of Computer Science and Technology,
Xi'an Jiaotong University, Xi'an 710049, China
qian.l@stu.xjtu.edu.cn, wgwu@mail.xjtu.edu.cn, longxu20145@gmail.com,
huangjhsx@gmail.com, b83316772@163.com

Abstract. As an interesting commercial application, rendering plays an important role in the field of animation and movie production. Generally, render farm is used to rendering mass images concurrently according to the independence among frames. How to scheduling and manage various rendering jobs efficiently is a significant issue for render farm. Therefore, the prediction of rendering time for frames is relevant for scheduling, which offers the reference and basis for scheduling method. In this paper a statistics based prediction method is addressed. Initially, appropriate parameters which affect the rendering time are extracted and analyzed according to parsing blend formatted files which offers a general description for synthetic scene. Then, the sample data are gathered by open source software Blender and J48 classification algorithm is used for predicting rendering time. The experimental results show that the proposed method improve the prediction accuracy about 60 % and 75.74 % for training set and test set, which provides reasonable basis for scheduling jobs efficiently and saving rendering cost.

Keywords: Rendering · Prediction · Statistics · Classification · Render farm

1 Introduction

High performance computing (HPC) provides a chance for scientists to use large scale computing resources to settle problems in intensive computation application [15]. Gradually, researchers concentrate on how to manage what application could be applied on HPC platforms to maximize the performance. As an intensive computation and massive data access application, rendering with the feature of independence among frames is suitable for parallel processing in HPC environment. Rendering is the process of creating an image from a model by means of computer programs [2,16], which is used widely on many field, for example, animation design, development of games, simulation [3], etc.

With the rapid development of animation industry, rendering is widely used in movie production and directly determines the visual effect of animation works.

G.R. Gao et al. (Eds.): NPC 2016, LNCS 9966, pp. 73–84, 2016.
DOI: 10.1007/978-3-319-47099-3_6

For an animation production, rendering a photo-realistic image with high resolution of 4K to 6K can be exceptionally costly, often taking days, or even weeks. Generally, render farm, which is a cluster of interconnected computers used for rendering images in parallel [4], is used by companies to rendering mass images concurrently, which has been proven to be quite effective way to cutting down hours of render time into minutes.

When using the render farm, users just submit their rendering jobs which includes the requirement of number of rendering resources, types of rendering software, budget and deadline etc. through web portal or client. The render farm with a pool of finite rendering resources receives the rendering jobs, chooses appropriate resources according to the demands of rendering users and scheduling rendering jobs according to some policies. Generally speaking, the research spot is focus on optimizing and design of scheduling policy [9,19,26]. But many issues are based the on the premise of obtaining the application running time in advance or have not a-priori knowledge on execution time prediction [17], which will negatively impacts the efficiency of scheduling policy, especially for the static policy from the view of system. And from the view of users, without prediction, users have no idea how long they will be waiting for finish rendering jobs, users cannot obtain better experience on rendering service and choose an economical way to employ rendering resources. Therefore, how to predict running time of applications and improve prediction accuracy is significant topic for render farm.

In this paper, we propose a statistics based prediction method to provide a guidance for scheduling policy in render farm. The proposed method extracts and analyzes firstly the appropriate parameters which affect the rendering time by the way of parsing blend formatted files. Then, the sample data are gathered by open source software Blender and we use J48 classification algorithm to train and predict rendering time. And the test result indicate that our method improves the prediction accuracy about 60 % and 75.74 % for training set and test set respectively.

The rest of this paper is organized as follows: related works are introduced in Sect. 2. In Sect. 3, we describe the system design and proposed method. Section 4 presents the experimental results. And conclusion and future works are provided in Sect. 5.

2 Related Works

Rendering is one of the most important research topics in computer graphics, and it is closely related to the other technologies. Rendering plays an important role in the field of animation and movie production. The execution time prediction for rendering is necessary for scheduling and user's Qos. Obviously for this schedule to be effective and experience to be better, a good prediction of the execution time for rendering is required.

Many researchers have explored the use of various time models to predict execution time. According to auto-regressive model a time series based prediction method [21] is also proposed. For physically-based medical simulation,

a multirate output-estimation method using the ARMAX model [11] is presented to improve the computational speed and accuracy. By rigid model, Liu et al. [14] proposes a proxy position prediction method using the geometric patch. The patch is calculated from the real-time contact region prediction method. Hou and Sourina [8] present a new prediction method based on auto-regressive model for smooth haptic rendering [22] to overcome the low update rate of the physical simulation with complex or deformable models. Schnitzer et al. [17] predict the individual execution time of graphics commands using models for the main three commands relevant for rendering, namely, FLUSH, CLEAR, and DRAW.

And then, prediction ideology is also be used for various types of render. For remote image-based rendering (IBR) [12] in mobile virtual environments, Lazeml et al. [10] propose a prefetching scheme that predicts the client potential path based on the movement history using a linear prediction model. The server then sends one predicted image with every position update for the client, which reduces the server load and improves the system scalability. In Depth image-based rendering (DIBR), a combined hole-filling approach [25] with spatial and temporal prediction is presented.

Considering the workload in the process of rendering, Wimmer and Wonka [20] present an estimation approach based on sampling rendering times, and an improved mathematical heuristics based on a cost function. They propose two hardware extensions, i.e. a time-stamping function and a conditional branch mechanism, which make rendering time estimations more robust. Meantime, the proposed estimation method for rendering time can also be used to calculate efficient placement of impostors in complex scenes. Doulamis et al. [7] address the method of 3D rendering workload prediction based on a combined fuzzy classification and neural network model. Litke et al. [13] predict computational workload of jobs assigned for execution 3D image rendering on commercially exploited Grid. And Doulamis et al. [6] propose an efficient non-linear task workload prediction mechanism with a non-linear model incorporated with a fair scheduling algorithm for task allocation and resource management in Grid computing. The history-based motion prediction method [23] aims at high file hits and predicts the future motion according to the historical ones, which produces better results when interpolation is added. Son et al. [18] propose a new approach for predicting and identifying the load condition of agents to solve load balancing problem. However, these methods only works for static model but ignore the influence of the parameters used in render files.

3 Proposed Strategy

3.1 System Design

A workflow of prediction a rendering process is depicted in Fig. 1. The input scene files are encoded on BLEND format which is standard format support by Blender. Blender is the free and open source 3D creation suite. It supports

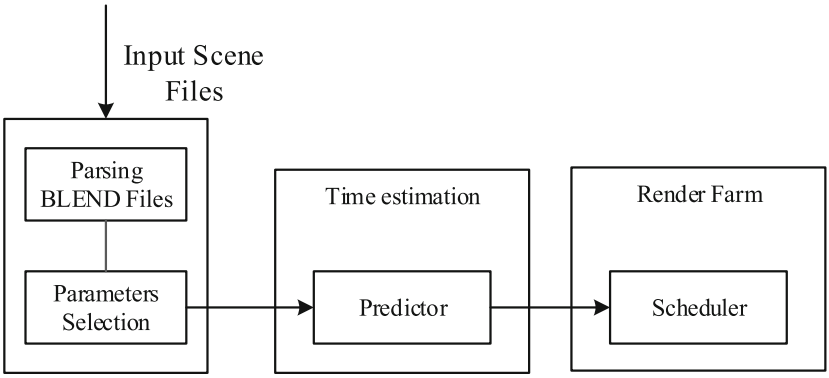

Fig. 1. Workflow of prediction

the entirety of the 3D pipeline-model, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation [1]. And the scene file offers a general framework for describing the structure of a synthetic world and uses the minimum information required of interacting the rendering environment. Also it describes the rendering algorithm as well as respective parameters used for performing the rendering. To identify the parameters that impact the rendering process and the algorithm used, the scene files are parsed. In this way, a set of candidate parameters is constructed, which is responsible for selecting the most significant parameters among all candidates, i.e., a feature vector of appropriate attributes is constructed to predict the rendering time.

The predictor predicts the time of a rendering process by taking into consideration the feature vector. The J48 classification algorithm is used for training and predicting rendering time. And then, scheduler will scheduling rendering jobs according to exploiting information provided by the previous prediction result. In the paper, we mainly focus the process of prediction.

3.2 Parsing of Parameters

A BLEND file usually contains many parameters which describe the rendering attributes, and some parameters may have less impact on rendering time. We use Blender's API for Python scripting to parse parameters and the scene image is shown in Fig. 2. And Table 1 shows the initial parsing result.

We use the method of control variable to select the parameters and change the corresponding value of parameter to get rid of the parameters with less impact on the rendering time. The rendering time is an average value obtained by 20 times rendering. As shown in Table 1, the change of parameters value has less impact on the rendering time. Except that, the vertex, edge and face of object is fixed in a scene and the impact of these factors is included in that of no. of Mesh. And then three parameters are also rejected. Therefore, at the processing of prediction we don't consider these parameters. Then, we will introduce the parameters which mainly impact on the rendering time.

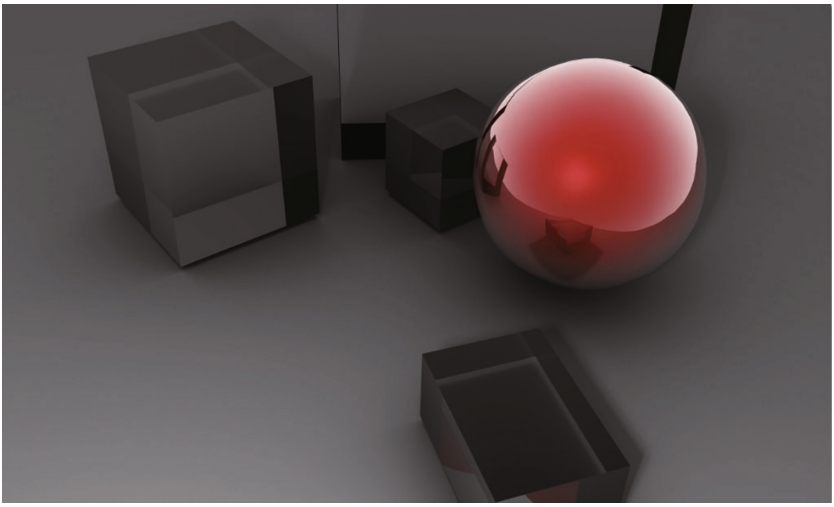

Fig. 2. The scene image for prediction

Table 1. The parameters with less impact on rendering time

Parameter	Variation range	Rendering time
Intensity of illuminant	[1.01–11]	[55.0401–59.9534]
No. of camera	[1–10]	[55.0572–55.6572]
Absorption types of ambient light	[MULTIPLY, ADD]	[50.3848–55.6572]
Absorption intensity of ambient light	[1–10]	[50.3848–55.6572]
Types of diffuse reflection	5 types	[50.1567–55.6572]
Types of highlight	5 types	[50.1688–55.6572]
Types of anti-aliasing filter	7 types	[55.6572–58.8323]

– Shadow [5], Blender provides two types of shadow, Ray-Tracing Shadow and Buffer Shadow. Table 2 shows the relationship between rendering time and types of Buffer Shadow. When an image is rendered, the rendering time with different types of Buffer Shadow is different.

– Light type, there are five types of light, point light, sun light, spot, hemi light and area light. Hemi light has no sampling value so it is rejected. We use formula (1) to indicate the influence of light types on rendering time. And the coefficient is empirical value.

$$\begin{aligned} Light\_s = & \tfrac{1}{7} * Point\_s + \tfrac{1}{7} * Sun\_s \\ & + \tfrac{1}{7} * Spot\_s + Aera\_s \end{aligned} \tag{1}$$

Table 2. The rendering time with different types of shadows

Types	Rendering time
Regular	49.7928
Irregular	50.6198
Halfway	52.3219
Deep	50.3958

– Number of light, which is an important factor on rendering time. In a scene, there includes various types of light corresponding to different number of light. We use formula (2) to indicate the influence on rendering time. And the coefficient is empirical value.

$$\begin{aligned} NLight\_s = & \tfrac{1}{5} * NPoint\_s + \tfrac{1}{5} * NSun\_s \\ & + \tfrac{1}{10} * NSpot\_s + \tfrac{1}{10} * NHemi\_s \\ & + Aera\_s \end{aligned} \tag{2}$$

– Ambient light, there are two types, raytrace and approximate. The approximate is rarely used and we use raytrace as the main factor. The raytrace usually includes three types, CONSTANT_QMC, ADAPTIVE_QMC, CONSTANT_JITTERED. We use formula (3) to indicate the influence on rendering time. And the coefficient is empirical value.

$$\begin{aligned} Ambientlight = & 5 * CONSTANT\_QMC \\ & + 3.5 * ADAPTIVE\_QMC \\ & + 1.4 * CONSTANT\_JITTERED \end{aligned} \tag{3}$$

– Pixel value, which is the product of resolution ratio, the more pixels the more elaborate image, and the rendering time is longer. In the paper, we use 17 types of resolution ratio.

– Tile size, the more tile size, the longer rendering time. Usually, the tile size is approximate number of resolution ratio.

– Number of object, the object has many attributes and in the paper we just consider the influence of number of object. The more objects the longer rendering time.

– Anti-aliasing, there are 4 types of Anti-aliasing sample values, 5, 8, 11, 16. And the value 0 means that the anti-aliasing is not used. The impact is shown in Table 3.

– Raytrace method, there include 6 types, AUTO, OCTREE, BLIBVH, VBVH, SIMD_SVBVH and SIMD_QBVH. The impact with different types is shown in Table 4.

Table 3. The rendering time with different types of Anti-aliasing sample values

Value	Rendering time
0	14.5008
5	55.6572
8	85.4189
11	116.9777
16	171.1598

Table 4. The rendering time with different types of Anti-aliasing sample values

Type	Rendering time
AUTO	55.6572
OCTREE	162.5122
BLIBVH	138.0688
VBVH	78.2354
SIMD_SVBVH	56.6742
SIMD_QBVH	59.0663

3.3 Strategy Description

In this section, we will introduce the prediction strategy according to the parameters parsed in above section. We use Weka as tool and J48 classification algorithm [24]. The J48 has the following advantages.

- Easy understand for rules. Each branch of the decision tree in J48 corresponds to a classification rule, and the total rules set is easy to understand.
- The fast running. The time complexity of J48 algorithm is $O(n^3)$, n is sample size.
- The higher accuracy.

4 Experimental Results

At the beginning, we need dispose the original data. The inactive data is to be rejected. And then, the rendering time set need to be discretized, which is helpful to classification and prediction. We sort the rendering time set with descending order, and each 5 second as a subsection.

The disposed data is input into Weka and Fig. 3 shows the visualization result. The rendering time is automatically discretized into 20 classes.

We use the J48 algorithm provided by Weka and the accuracy rate of training set is 49.14 % after 10 times of cross validation. And the accuracy rate of test set is 66.67 %. The classification matrix is shown in Figs. 4 and 5.

Fig. 3. The visualization result of discretized data

```
7  2  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  a = 2T
3  4  1  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  b = 3T
0  4  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  c = 4T
0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  d = 5T
0  0  1  1  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  e = 6T
0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  f = 7T
1  0  0  0  1  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  g = 8T
0  0  0  0  1  0  0  0  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  h = 9T
0  0  0  0  1  0  0  0 10  5  7  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  i = 10T
0  0  0  0  0  0  0  0  3 37 16  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  j = 11T
0  0  0  0  0  0  0  0  1  5 51  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  k = 12T
0  0  0  0  0  0  0  0  0 11  5  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  l = 13T
0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  m = 14T
0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  n = 15T
0  0  0  0  0  0  0  0  0  2  0  0  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  o = 16T
0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  p = 17T
0  0  0  0  0  0  0  0  0  1  0  0  1  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0 |  q = 18T
0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  r = 19T
0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  s = 20T
0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  t = 21T
0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  u = 22T
0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  v = 23T
0  0  0  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0 |  w = 24T
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0 |  x = 26T
0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  y = 28T
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0 |  z = 29T
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0 | aa = 31T
0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | ab = 33T
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0 | ac = 34T
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | ad = 35T
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0 | ae = 42T
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0 | af = 44T
```

Fig. 4. The classification matrix of training set

10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	a = 2T
3	6	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	b = 3T
0	0	4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	c = 4T
0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	d = 5T
1	0	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	e = 6T
0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	f = 7T
1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	g = 8T
0	0	0	0	0	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	h = 9T
0	0	0	0	0	1	0	0	17	2	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	i = 10T
0	0	0	0	0	0	0	0	3	41	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	j = 11T
0	0	0	0	0	0	0	0	1	4	53	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	k = 12T
0	0	0	0	0	0	0	0	0	0	4	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	l = 13T
0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	m = 14T
0	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	n = 15T
0	0	0	0	0	0	0	0	0	0	2	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	o = 16T
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p = 17T
0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	q = 18T
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	r = 19T
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	s = 20T
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	t = 21T
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	u = 22T
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	v = 23T
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	w = 24T
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	x = 26T
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	y = 28T
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	z = 29T
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	aa = 31T
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ab = 33T
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	ac = 34T
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ad = 35T
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	ae = 42T
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	af = 44T

Fig. 5. The classification matrix of test set

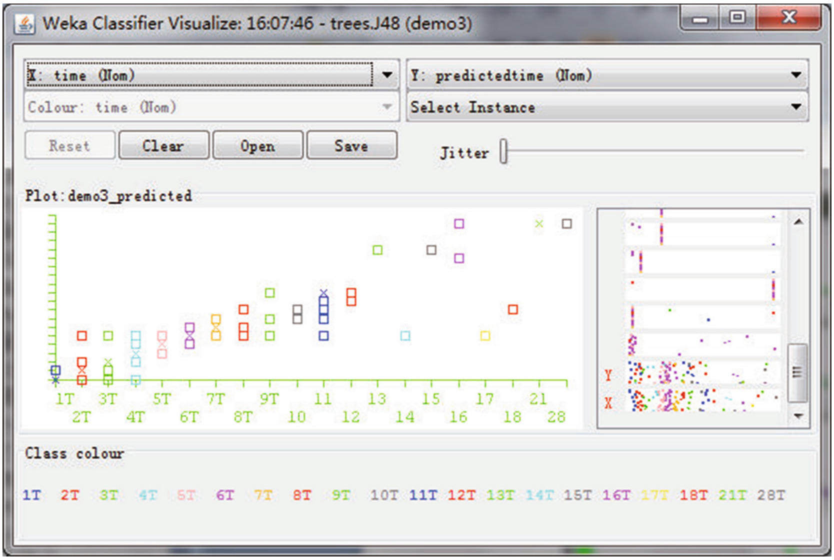

Fig. 6. The comparison of rendering time and prediction result.

The low accuracy rate is due to the less change range for each parameter. The useful data in each classification is less and unrepresentative. Therefore, we can suitably reduce the accuracy of the classification for rendering time and use 10 second as a subsection and discretize rendering time set again. And the accuracy rate of training set and test set is 60 % and 75.74 % respectively. The prediction result is shown in Fig. 6. The comparison result shows that the J48 classification algorithm can obtain the accurate prediction result for rendering time according to the main parameters parsed from scene files.

5 Conclusion and Future Work

In this paper, we present a statistics based prediction method for rendering application. The scene files with BLEND format are parsed to extract and analyze the parameters which affect the rendering time. And we use Blender to gather the sample data and J48 classification algorithm to predict rendering time. The experimental results show that the proposed method improve the prediction accuracy about 60 % and 75.74 % for training set and test set respectively, which provides reasonable guide for scheduling jobs efficiently and saving rendering cost.

Acknowledgments. This work is supported by Natural Science Foundation of China (61202041, 91330117), National High-Tech R&D (863) Program of China (2012AA01A306, 2014AA01A302) and the Key Technologies Program for Social Development of Shaanxi Province (2016SF-428). Computational resources have been made available on Xi'an High Performance Computing Center.

References

1. http://www.blender.org/
2. Baharon, M.R., Shi, Q., Llewellyn-Jones, D., Merabti, M.: Secure rendering process in cloud computing. In: 2013 Eleventh Annual International Conference on Privacy, Security and Trust (PST), pp. 82–87. IEEE (2013)
3. Chapman, W., Ranka, S., Sahni, S., Schmalz, M., Moore, L., Elton, B.: A framework for rendering high resolution synthetic aperture radar images on heterogeneous architectures. In: 2014 IEEE Symposium on Computers and Communication (ISCC), pp. 1–6. IEEE (2014)
4. Chong, A., Sourin, A., Levinski, K.: Grid-based computer animation rendering. In: Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia, pp. 39–47. ACM (2006)
5. Doulamis, N., Doulamis, A., Dolkas, K., Panagakis, A., Varvarigou, T., Varvarigos, E.: Non-linear prediction of rendering workload for grid infrastructure. Mach. Graph. Vis. **13**(1/2), 123–136 (2004)
6. Doulamis, N., Doulamis, A., Litke, A., Panagakis, A., Varvarigou, T., Varvarigos, E.: Adjusted fair scheduling and non-linear workload prediction for QoS guarantees in grid computing. Comput. Commun. **30**(3), 499–515 (2007)

7. Doulamis, N.D., Doulamis, A.D., Panagakis, A., Dolkas, K., Varvarigou, T., Varvarigos, E., et al.: A combined fuzzy-neural network model for non-linear prediction of 3-D rendering workload in grid computing. IEEE Trans. Syst., Man, Cybern., Part B: Cybern. **34**(2), 1235–1247 (2004)
8. Hou, X., Sourina, O.: A prediction method using interpolation for smooth six-DOF haptic rendering in multirate simulation. In: 2013 International Conference on Cyberworlds (CW), pp. 294–301. IEEE (2013)
9. Khan, A.R., Saleem, M., Banday, S.A.: Improving the performance of hierarchical scheduling for rendering. In: 2014 Innovative Applications of Computational Intelligence on Power, Energy and Controls with Their Impact on Humanity (CIPECH), pp. 457–460. IEEE (2014)
10. Lazem, S., Elteir, M., Abdel-Hamid, A., Gracanin, D.: Prediction-based prefetching for remote rendering streaming in mobile virtual environments. In: 2007 IEEE International Symposium on Signal Processing and Information Technology, pp. 760–765. IEEE (2007)
11. Lee, K., Lee, D.Y.: Real-time haptic rendering using multi-rate output-estimation with armax model. In: International Conference on Control, Automation and Systems, ICCAS 2007, pp. 1821–1826. IEEE (2007)
12. Leung, W.H., Chen, T.: Compression with mosaic prediction for image-based rendering applications. In: 2000 IEEE International Conference on Multimedia and Expo, ICME 2000, vol. 3, pp. 1649–1652. IEEE (2000)
13. Litke, A., Tserpes, K., Varvarigou, T.: Computational workload prediction for grid oriented industrial applications: the case of 3D-image rendering. In: IEEE International Symposium on Cluster Computing and the Grid, CCGrid 2005, vol. 2, pp. 962–969. IEEE (2005)
14. Liu, Y., Chou, W., Yan, S.: Proxy position prediction based continuous local patch for smooth haptic rendering. J. Comput. Inf. Sci. Eng. **12**(3), 031004-1–031004-9 (2012)
15. Mochocki, B.C., Lahiri, K., Cadambi, S., Hu, X.S.: Signature-based workload estimation for mobile 3D graphics. In: Proceedings of the 43rd Annual Design Automation Conference, pp. 592–597. ACM (2006)
16. Monte, C.F.P., Piccoli, F., Luciano, C., Rizzi, S., Bianchini, G., Scutari, P.C.: Estimation of volume rendering efficiency with gpu in a parallel distributed environment. Procedia Comput. Sci. **18**, 1402–1411 (2013)
17. Schnitzer, S., Gansel, S., Durr, F., Rothermel, K.: Concepts for execution time prediction of 3D GPU rendering. In: 2014 9th IEEE International Symposium on Industrial Embedded Systems (SIES), pp. 160–169. IEEE (2014)
18. Son, B.H., Lee, S.W., Youn, H.Y.: Prediction-based dynamic load balancing using agent migration for multi-agent system. In: 2010 12th IEEE International Conference on High Performance Computing and Communications (HPCC), pp. 485–490. IEEE (2010)
19. Tamm, G., Kruger, J.: Hybrid rendering with scheduling under uncertainty. IEEE Trans. Vis. Comput. Graph. **20**(5), 767–780 (2014)
20. Wimmer, M., Wonka, P.: Rendering time estimation for real-time rendering. In: Rendering Techniques, pp. 118–129 (2003)
21. Wu, J., Song, A., Li, J.: A time series based solution for the difference rate sampling between haptic rendering and visual display. In: IEEE International Conference on Robotics and Biomimetics, ROBIO 2006, pp. 595–600. IEEE (2006)
22. Xiao-peng, H., Cheng-jun, C., Yi-qi, Z.: Contact elements prediction based haptic rendering method for collaborative virtual assembly system. In: WRI Global Congress on Intelligent Systems, GCIS 2009, vol. 4, pp. 259–263. IEEE (2009)

23. Xiaohong, Z., Shengzhong, F., Jianping, F.: A history-based motion prediction method for mouse-based navigation in 3D digital city. In: Seventh International Conference on Grid and Cooperative Computing, GCC 2008, pp. 686–690. IEEE (2008)
24. Yadav, A.K., Chandel, S.: Solar energy potential assessment of western Himalayan Indian state of Himachal Pradesh using J48 algorithm of WEKA in ANN based prediction model. Renew. Energy **75**, 675–693 (2015)
25. Yu, W., Wang, W., He, G., Goto, S.: Combined hole-filling with spatial and temporal prediction. In: 2013 20th IEEE International Conference on Image Processing (ICIP), pp. 3196–3200. IEEE (2013)
26. Yu, X., Zhang, X., Liu, L., Hwang, J.N., Wan, W.: Dynamic scheduling and real-time rendering for large-scale 3D scenes. J. Sig. Process. Syst. **75**(1), 15–21 (2014)