基于 AdaBoost. MH 的 Reyes 渲染架构时间预估算法

孟庆利[1] 吕 琳[1] 靳 颖[1] 孟祥旭[1] 孟 雷[2]
(山东大学计算机科学与技术学院 济南 250101)[1] (南洋理工大学计算机工程学院 新加坡)[2]

摘 要 在大规模真实感渲染系统中,需要对渲染任务进行分解和调度,将其优化后分配给不同的可用计算资源,实现快速集群渲染。为了实现渲染任务的有效分解和调度,提高并行效率,高精度的时间预估算法是不可欠缺的。通过深入研究使用 RenderMan 规范的渲染器常用的 Reyes 渲染架构中对渲染时间产生影响的各种因素,分析提取出影响渲染时间的7大要素特征,提出了基于 AdaBoost. MH 的渲染时间预估算法。通过在基于 Reyes 渲染架构的渲染引擎中的实验与测试表明,训练集和测试集的准确率分别达到79%和78%,为渲染任务的并行调度奠定了基础,同时也为渲染费用预估提供了依据。

关键词 时间预估,AdaBoost. MH 算法,Reyes 渲染架构,集群渲染
中图法分类号 TP391 **文献标识码** A

Time Prediction for Reyes Rendering Architecture Based on AdaBoost. MH Algorithm

MENG Qing-li[1] LV Lin[1] JIN Ying[1] MENG Xiang-xu[1] MENG Lei[2]
(Department of Computer Science and Technology, Shandong University, Jinan 250101, China)[1]
(School of Computer Engineering, Nanyang Technological University, Singapore)[2]

Abstract A high performance computer system, e. g. a computer cluster, built for large-scale photorealistic rendering, i. e. render farm, is a basic infrastructure for producing CG animations and movie special effects. In a render farm, one of the key issues is the strategy of scheduling and dispatching rendering jobs, which greatly affects the computing efficiency. Time prediction for a render job plays an important and essential role in the job scheduling and dispatching stage. However, there is no feasible algorithm and even little research work on this problem. We focused on the Reyes rendering architecture. We first analyzed the factedors that affect the rendering time and extracted the seven key features as the feature vector based on the analysis. Then we proposed a time prediction framework based on AdaBoost. MH algorithm, in which we transformed the rendering time into intervals and combined them with the feature vector to obtain the samples. Experimental results show the effectiveness of the algorithm, and the accuracy of training set and test set is 79% and 78%.

Keywords Time prediction, AdaBoost. MH algorithm, Reyes rendering architecture, Render farm

1 引言

随着多媒体技术和 CG 技术的发展,渲染引擎在电影动画、模拟仿真、游戏特效等方面具有越来越广泛的应用。同时,渲染的计算量也日益增加,单个计算机难以应对其巨大的计算量,因此集群渲染应运而生。集群渲染[1,2]平台主要采用 PRMan、3Delight、Pixie 等渲染引擎提供渲染服务,这些渲染软件多使用 Reyes[3,4](Renders Everything You Ever Saw)渲染算法,可实现快速高质量渲染。

尽管 Reyes 渲染较其他算法渲染速度快,但是对于大规模真实感渲染任务,需要集群渲染来实现,渲染任务的并行化不可或缺,渲染时间预估则是渲染任务的分解与并行调度中关键的一步,此外,用户的渲染费用预估同样需要渲染任务的时间预估提供依据。Reyes 渲染框架下的时间预估是根据大量的历史数据预测新场景需要花费的渲染时间,这方面的研究工作较少。Antonios Litke[5]等人使用人工神经网络算法建立网格基础设施上任务的负载预估模型,并使用 3D 渲染应用作为实验对象。通过从 RIB 文件中提取出 Render Width、Render Height、Cast Shadow 等8个参数作为人工神经网络算法的输入,输出浮点操作、内存、磁盘、网络传输量等7个资源信息,根据预估资源信息进行下一步资源管理和调度。不同的渲染器使用不同的渲染算法,并对应不同的影响因素,以不同的方式影响渲染时间[6],文献[5]在对渲染任务进行预估时,没有对渲染算法进行区分。Nikolaos Doula-

到稿日期:2013-05-20 返修日期:2013-08-16 本文受863重大项目子课题(2012AA01A306),国家自然科学基金(61202147),山东省自然科学基金(ZR2012FQ026)资助。

孟庆利(1989—),男,硕士生,主要研究方向为人机交互与虚拟现实等,E-mail:meng-qingli@163.com;**吕 琳**(1981—),女,博士,副教授,CCF会员,主要研究方向为计算机图形学、计算几何等;**靳 颖**(1988—),男,硕士生,主要研究方向为人机交互与虚拟现实等;**孟祥旭**(1962—),男,博士,教授,CCF会员,主要研究方向为人机交互与虚拟现实等;**孟 雷**(1987—),男,博士生,主要研究方向为机器学习等。

mis[6,7]等人提出使用模糊神经网络模型对 ray tracing、radiosity 和 Monte Carlo irrandiance 3 种渲染算法通过选择分辨率、灯光类型和数量、与渲染算法相关的参数等一般特性以及几何复杂度、表面纹理、材质等几何特性进行负载预估。Reyes 渲染算法也是目前较为流行的渲染算法之一，文献[6]没有对该渲染算法进行分析和研究。国内外对 Reyes 渲染架构的时间预估研究较为少见，这也是本文研究的出发点之一。时间预估是回归问题，神经网络[8]作为解决回归问题的典型代表，具有较多缺陷：对网络的配置和训练缺乏严格的理论指导；收敛速度慢；易收敛到局部极小值；易出现过适应现象。而 AdaBoost 为特征选择和非线性分类提供了一种简单而有效的阶段性学习策略，本文尝试使用较为成熟的 AdaBoost. MH 算法进行时间预估。

本文对使用 RenderMan 规范的渲染器常用的 Reyes 渲染架构中对渲染时间产生影响的各种因素进行了深入研究，分析提取出影响渲染时间的 7 大要素特征：PixelSamples、像素数量、灯光数量、阴影图数量、材质数量、细分面片数量、线程数量，提出了基于 AdaBoost. MH 的渲染时间预估算法。归纳本文的贡献，如下：

1）详细分析了 Reyes 渲染架构中对渲染时间产生影响的各种因素。

2）为 AdaBoost. MH 算法在时间预估方面找到新的应用。

3）提出 Reyes 渲染架构下的时间预估算法，为集群渲染的并行调度奠定了基础，同时为用户费用预估提供了依据。

本文第 2 节介绍相关知识；第 3 节进行特征提取、类别处理，以获取样本；第 4 节使用 AdaBoost. MH 算法进行学习分类；最后得到第 5 节的时间预估算法，并在第 6 节进行实验验证。

2 相关知识

2.1 大规模渲染并行调度流程

对于大规模真实感渲染系统，任务的并行化不可或缺。在大规模渲染并行调度中，首先输入大量渲染任务，这些渲染任务是以 RIB[9]（RenderMan Interface Bytestream）文件形式存在的，每一帧为一个渲染任务，并对应 RIB 相关文件。解析模块将输入的 RIB 文件进行解析，提取出不同的有效特征向量，输入到对应的时间预估算法和其他资源预估算法，各预估算法通过预估计算得到每一个渲染任务的渲染时间和渲染需要的其他资源。根据这些预估资源，调度模块以每一帧的渲染任务为调度单位，将渲染任务分配给不同的可用资源，最后在节点上进行并行计算。大规模渲染并行调度流程如图 1 所示。

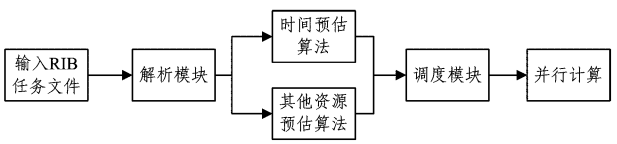


图 1 大规模渲染并行调度流程

2.2 Reyes 渲染架构

Reyes 渲染算法是目前最常用的渲染算法之一。输入是一个 RIB 文件，包括相关的着色器和贴图文件，输出是一幅渲染后的图像，主要分为 Bound、Split、Dice、Shade、Bust、Hide 几个步骤。首先将 RIB 文件解析成由灯光、几何、摄像机等组成的场景，计算每个几何元素的包围盒，根据视线范围进行裁剪。然后将几何元素划分成若干几何子元素，划分标准通过 ShadingRate[9]和 GridSize[9]两个参数设定。将几何元素细分成网格后，对每个细分面片的所有顶点进行着色，并将细分网格分解，再一次进行裁剪。最后通过参数 PixelSamples[9]对每个像素采样，并进行滤波计算，最终得到每个像素的颜色，输出渲染图像。

2.3 AdaBoost. MH 算法

时间预估是根据大量的历史数据预测渲染新场景需要花费的时间。为了实现时间预估的智能化，机器学习[10]是一个明智的选择。AdaBoost（Adaptive Boosting）算法[11,12]是 Boosting[13]算法的一个典型特例，能够提高任意给定算法的性能。该算法通过多次迭代，将每次迭代产生的弱分类器集合起来，构成一个更强的最终分类器（强分类器）。与大多数学习算法相比，AdaBoost 算法更不容易产生过拟合现象[14]。本文采用 Schapire 和 Singer 提出的 AdaBoost. MH[15,16]算法进行多类单标签分类，该算法将多类分类问题转换为二类分类问题，并使用简单决策树[10]作为弱分类器，经过若干迭代，最终得到能够预测分类的强分类器。

3 抽取样本

为了得到更加精确的时间预估算法，样本的抽取十分重要。通过分析 Reyes 渲染算法中对渲染时间有影响的各种因素，归纳出需要提取的特征。为了得到样本的类别，将渲染时间等分成 n 段，每一段作为一个类别。然后将特征和类别组合，得到 AdaBoost. MH 算法需要的样本。

3.1 特征提取

根据 Reyes 渲染算法流程，主要考虑表 1 中的因素。

表 1 影响因素

因素	理论说明
PixelSamples	影响采样的大小，采样越高，计算量越大，渲染时间越长
ShadingRate	影响细分面片的面积，取值越小，细分面片越多，渲染时间越长
分辨率	影响图像像素个数，像素个数越多，计算量越大，渲染时间越长
灯光	影响着色的时间，灯光数量越多，渲染时间越长
阴影	影响生成阴影和着色的时间，阴影数越多，渲染时间越长
几何	影响细分面片数量，几何越复杂，细分面片越多，渲染时间越长
材质	影响着色的时间，材质数量越多，渲染时间越长
视角	影响细分面片数量，视角越近，细分面片越多，渲染时间越长

选择基本场景，如图 2 所示。本文使用控制变量法对表 1 中的因素依次进行分析并总结，其中渲染时间单位为秒。

图 2 基本场景

1）PixelSamples

该参数将一个像素在水平方向和垂直方向分成若干子像素，用于采样，将参数取值为 1～32。通过渲染场景，得到渲染时间随 PixelSamples 变化的二维图表，如图 3 所示。可以

看出，渲染时间随着 PixelSamples 的变大而变大，可以将 PixelSamples 作为一个特征。

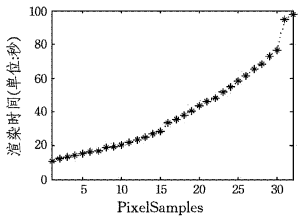


图 3　渲染时间随 PixelSamples 变化的情况

2)ShadingRate

ShadingRate 控制细分面片的面积上限，将参数取值为 0.1～5.0。通过渲染场景，得到渲染时间随 ShadingRate 变化的二维图表，如图 4 所示。可以看出，渲染时间随着 ShadingRate的变大而减小，且 ShadingRate 取值越大，渲染时间下降越慢，ShadingRate 的取值直接影响细分面片的个数，不作为特征使用。

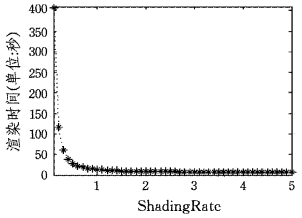


图 4　渲染时间随 ShadingRate 变化的情况

3)分辨率

分辨率决定最后渲染的图像所包含的像素个数，使用目前常用的 43 种分辨率进行实验分析，得到渲染时间随分辨率变化的二维图表，如图 5 所示，其中横坐标使用像素个数表示。可以看出，渲染时间随着像素个数的变大而变大，分辨率直接影响计算量，可以作为一个特征。

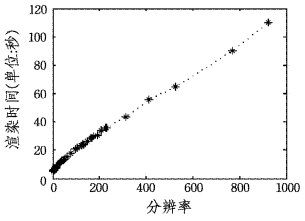
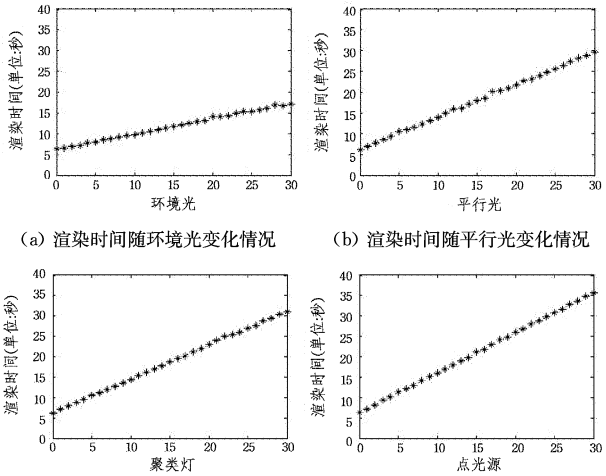


图 5　渲染时间随分辨率变化的情况

4)灯光

Reyes 渲染算法中，灯光会影响着色时间，本文使用 4 种标准灯光类型，分别为环境光、平行光、聚光灯、点光源。通过分别将每种灯光类型个数取值为 0～30，得到渲染时间随灯光个数变化的二维图表，如图 6 所示。可以看出，渲染时间随着灯光个数的变大而变大，且成线性关系。其中，图(a)斜率为0.354，图(b)斜率为 0.759，图(c)斜率为 0.802，图(d)斜率为0.950。通过建立多个场景进行分析，发现除环境光外，其他 3 种灯光对渲染时间的影响相差不大，且环境光对渲染时间的影响平均值近似为其他 3 种灯光的 1/2。所以，将灯光数量作为一个特征，并使用式(1)进行计算，其中 *nlight* 表示灯光总数，*alight* 表示环境光数量，*dlight* 表示平行光数量，*slight* 表示聚光灯数量，*plight* 表示点光源数量。

$$nlight = alight * 0.5 + dlight + slight + plight \quad (1)$$



(a) 渲染时间随环境光变化情况　(b) 渲染时间随平行光变化情况

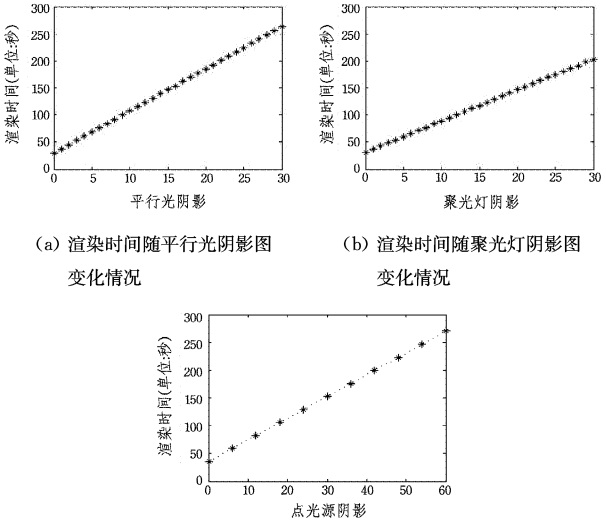
(c) 渲染时间随聚光灯变化情况　(d) 渲染时间随点光源变化情况

图 6　渲染时间随灯光变化的情况

5)阴影

阴影的产生方式有两种[16]，即阴影图和光线跟踪。本文使用阴影图作为实验对象。环境光不产生阴影，平行光和聚光灯具有特定的方向，产生阴影时，每个光源生成一个阴影图，而点光源是向三维空间的所有方向进行照射，本文所使用的渲染器将点光源剖分成上下、左右、前后 6 个方向，每个点光源生成 6 个阴影图。取每个灯光类型阴影图个数为 0～30，得到渲染时间随阴影变化的二维图表，如图 7 所示。可以看出，渲染时间随着阴影图个数的变大而变大，且成线性关系。其中，图(a)斜率为 7.506，图(b)斜率为 5.543，图(c)斜率为3.772。通过建立多个场景进行分析，发现除点光源阴影图外，其他两种灯光的阴影图对渲染时间的影响相差不大，且点光源阴影对渲染时间的影响平均值近似为其他两种灯光阴影图的 1/2。所以，将阴影图数量作为一个特征，并使用式(2)进行计算，其中 *nlightshadow* 表示灯光阴影图总数，*dlightshadow* 表示平行光阴影图数量，*slightshadow* 表示聚光灯阴影图数量，*plightshadow* 表示点光源阴影图数量。

$$nlightshadow = plightshadow * 0.5 + dlightshadow + slightshadow \quad (2)$$



(a) 渲染时间随平行光阴影图变化情况　(b) 渲染时间随聚光灯阴影图变化情况

(c)渲染时间随点光源阴影图变化情况

图 7　渲染时间随阴影图变化的情况

6)几何与材质

几何复杂度影响细分面片的个数，通过在场景中增加几何元素的个数表示几何复杂度，同时材质依附于几何元素，几何元素数量增加时，材质的数量也相应增加。由于不同类型材质的着色时间不同，且相差很大，为保证时间预估算法的精度，本文只考虑漫反射材质，这些材质着色时间相近，所以只需考虑数量，忽略其类型。本文将场景复制为原来的1～20倍，得到渲染时间随几何复杂度变化的二维图表，如图8所示。可以看出，渲染时间随着几何个数的增加而变大，由于几何复杂度影响细分面片的个数，不将其作为特征使用。另一方面，几何复杂度会影响材质数量，并且材质数量与细分面片没有关系，故将材质数量作为一个特征。

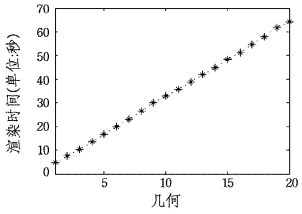


图8　渲染时间随几何变化的情况

7)视角

视角的远近会影响场景物体到投影平面的大小，进而影响细分面片的个数。通过对同一场景设置10个由远到近的视角，得到渲染时间随视角变化的二维图表，如图9所示。可以看出，渲染时间随着视角在一定范围内不断靠近会不断变大，视角影响细分面片个数，不将其作为特征使用。

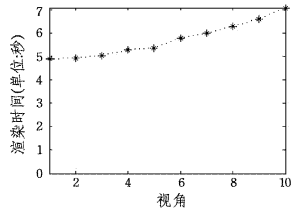


图9　渲染时间随视角变化的情况

上述分析中，ShadingRate、几何复杂度以及视角均会影响细分面片个数，细分面片个数直接影响渲染时间，故将细分面片个数作为一个特征。细分面片个数不能从RIB文件中直接得到，需要对RIB文件进行解析。灯光、阴影以及材质都会影响着色时间，由于单个灯光和阴影对渲染时间的影响相对于单个材质对渲染时间的影响相差甚大，故将其分别作为一个特征处理。

同时，在渲染命令中可以指定使用的线程数量，线程数量越多，渲染时间越短。通过对同一个场景指定1～8个线程进行渲染，得到渲染时间随线程数量变化的二维图表，如图10所示。可以将线程作为一个特征。

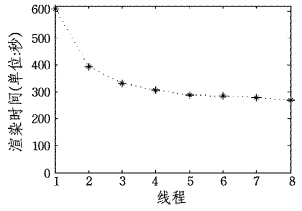


图10　渲染时间随线程数量变化的情况

总之，通过分析影响渲染时间的各种因素，本文提取出PixelSamples、像素数量、灯光数量、阴影图数量、材质数量、细分面片数量、线程数量等七维特征。

3.2　类别处理

使用具有自主产权的渲染引擎对场景进行渲染并得到渲染时间后，为了得到完整的样本，以进一步应用机器学习算法进行学习，需要对场景进行类别划分。本文将所有场景的渲染时间进行排序，根据最大值max和最小值min将时间等分成n段，每一段的长度为Δt，并作为一个类别，则

$$\Delta t=(\max-\min)/n \tag{3}$$

通过使用AdaBoost.MH算法的强分类器进行预测分类后，再将类别还原到时间段。因此，时间预估算法最后得到的结果是一个时间范围。

4　AdaBoost.MH算法的分类计算

本文的分类属于多类别单标签分类，AdaBoost.MH是一种直接将K类问题转化为K个两类问题的算法。该算法由于转化为两类问题，弱分类器的精度要求可以较容易地得到满足，多类分类效果改善明显[13]。已有研究表明[15]，AdaBoost.MH在现有的多类分类算法中准确率是较高的，因此本文采用该算法进行时间预估。

4.1　多类转换二类

AdaBoost.MH算法将样本集扩展J倍，将样本类别作为一个新的特征加入到原来的特征向量中，转换成为二类分类问题，再应用Real AdaBoost算法进行分类，得到每个原始样本属于各个类别的可能值，其中最大可能值对应的类别确定该样本最终的分类。如算法1[15]所示。

算法1　AdaBoost.MH算法

AdaBoost.MH [Schapire and Singer (1998)]

1. 将N个观测样本扩展到N×J对$((x_i,1),y_{i1}),((x_i,2),y_{i2}),\cdots,((x_i,J),y_{iJ})$，$i=1,\cdots,N$，其中$y_{ij}$表示样本i属于第j类的值域$\{-1,1\}$。
2. 对扩展的数据集执行Real AdaBoost算法，产生一个函数$F:\chi\times(1,\cdots,J)\mapsto R$；$F(x,j)=\sum_m f_m(x,j)$。
3. 输出强分类器：$\operatorname{argmax}_j F(x,j)$。

4.2　二类问题的分类

本文使用简单决策树作为学习过程中的弱分类器。设输入的n个样本为：$\{(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)\}$，其中$x_i$是输入的训练样本，$y_i\in\{-1,+1\}$分别表示负样本和正样本。学习过程中，首先初始化样本权重，且每个样本权重相同。然后经过M次迭代，在每次迭代中，使用简单决策树对每个特征计算一个阈值，得到相应的加权错误率，选择错误率最低的弱分类器作为本次迭代产生的最佳弱分类器，该弱分类器能够返回样本属于每个类别的可能值。然后根据该最佳弱分类器更新权重，增加分类错误的样本权重，降低分类正确的样本权重，进行下一次迭代，迭代完成后得到M个弱分类器。最后输出由M个弱分类器组合成的强分类器。具体过程如算法2[15]所示。

算法2　Real AdaBoost算法

Real AdaBoost

1. 初始化样本权重$\omega_i=1/N$，$i=1,2,\cdots,N$。
2. 对$m=1,2,\cdots,M$

(1)在样本权重ω_i分布下，由弱分类器得到一个类别可能值

$$p_m(x)=\hat{P}_\omega(y=1|x)\in[0,1]$$

(2)设置$f_m(x)\leftarrow\frac{1}{2}\log p_m(x)/(1-p_m(x))\in R$。

(3)设置 $\omega_i \leftarrow \omega_i \exp[-y_i f_m(x_i)]$，$i=1,2,\cdots,N$，然后归一化使 $\sum_i \omega_i = 1$。

3. 输出强分类器：$\text{sign}[\sum_{m=1}^{M} f_m(x)]$。

5 时间预估算法

通过上述分析和描述，可以得到时间预估算法的具体过程。首先将输入的 RIB 文件进行解析，提取出 PixelSamples、像素数量、灯光数量、阴影图数量、材质数量、细分面片数量等六维特征，并根据渲染命令提取出线程数量作为第七维特征。然后将渲染器的渲染时间划分为类别，与提取出的特征组合成样本。为了最终时间预估算法的健壮性和准确性，需要建立涵盖所有情况的样本。从大量的样本中抽取出各类别数目相同的训练集和测试集，将训练集应用于 AdaBoost. MH 算法，经过若干次迭代，得到强分类器，再使用该强分类器对测试集进行分类，将类别转换成时间范围后，与对应的真实渲染时间进行比较，得到时间预估准确率。具体流程如图 11 所示。

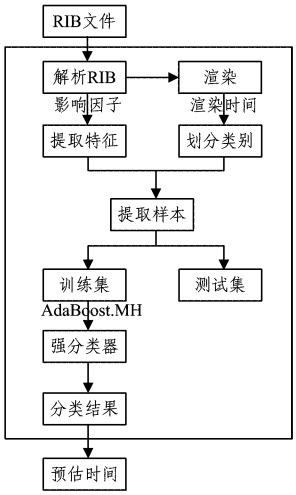


图 11 时间预估算法流程图

6 实验与分析

为了对本文提出的时间预估算法进行验证，需要建立多个场景模拟大量历史数据。通过使用建模软件 Autodesk Maya 2011，以图 2 所示为基本场景，各影响因素相互组合，建立 2400 个三维场景。根据各参数常用值，设定取值范围，如表 2 所列。其中，PixelSamples 最常用值为 3，ShadingRate 最常用值为 1.0，故选取其左右临近值；分辨率选取最常用的 4 种分辨率；灯光和阴影使用 4 种基本灯光均匀取值；几何和视角均影响细分面片的个数，各取值 2 个。

表 2 各因素取值说明

因素	取值范围	取值个数
PixelSamples	1、2、3、4、5	5
ShadingRate	0.6、1.0、1.6	3
分辨率	640 * 480、800 * 600、1024 * 768、1280 * 960	4
灯光	环境光、平行光、聚光灯、点光源依次增加，每次增加 5 个	
阴影	分别设置 5 个平行光和聚光灯产生阴影，设置 2 个点光源产生阴影	10
几何	每次将基本场景复制一次，总共复制 2 次	
材质	随几何变化	2
视角	对于每种场景设置两个不同远近的视角	2

通过使用具有自主版权的使能工具，将 Maya 场景导出到 RIB 文件。通过使用配置为 2.4GHz CPU，12G RAM 的高性能计算机，应用 64 位 Pixie 渲染器在 Reyes 渲染架构下解析并渲染 RIB 文件，在渲染时可以指定使用线程的数量，在本实验中设置线程数量分别为 1、2、4、8。渲染完成后，得到由所有场景的场景名、渲染时间、特征向量组成的信息文件。

根据信息文件，可以得到实验需要的数据，以评估时间预估算法的性能。具体实验环境为：3.4GHz CPU 和 8G RAM；操作系统为 Windows 7 64 位；开发环境为基于 C++的 Visual Studio 2008。将该信息文件中的渲染时间根据式(3)转换成类别，然后与特征向量组成样本，并保存到样本文件。样本文件与信息文件是一一对应的。

通过观察样本文件发现，每个类别的样本数目相差较大，且样本数目随着类别的增大而不断减少。为了获取更适合 AdaBoost. MH 算法的样本，需要每一类的样本数目相同。在本文实验中，式(3)中的变量 max 为 437.718 秒，min 为 1.967 秒，设定 n 为 35，则 Δt 为 12.45 秒。划分后，得到拥有 35 个类别的样本文件，然后从每个类别均匀抽取出 100 个样本，样本数不够 100 的类别不进行统计。最终得到 1000 个样本，总共 10 类。将抽取的样本划分为具有 900 个样本的训练集和 100 个样本的测试集，并更新相应的信息文件。部分样本如表 3 所列。

表 3 部分样本数据

类别	PixelSamples	像素	灯光	阴影图	细分面片	材质	线程
0	1	7040	2.5	0	418632	11	4
0	1	76800	7.5	5	219594	20	8
1	2	76800	7.5	10	803044	47	2
1	3	30720	12.5	10	1064102	56	4
2	3	48000	12.5	13	1701640	128	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

	0	1	2	3	4	5	6	7	8	9
0	84	6	0	0	0	0	0	0	0	0
1	12	51	22	4	0	0	0	1	0	0
2	0	24	33	24	7	1	0	0	0	1
3	0	14	9	47	11	7	1	0	0	1
4	0	2	1	24	43	11	3	3	2	1
5	0	3	2	12	21	25	17	6	3	1
6	0	0	1	9	11	17	18	21	6	7
7	0	0	0	3	2	11	9	44	9	12
8	0	0	0	2	2	6	7	25	21	27
9	0	0	0	0	0	7	3	10	9	61

(a) 训练集分类矩阵

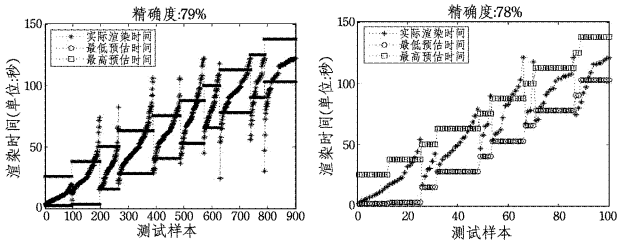
	0	1	2	3	4	5	6	7	8	9
0	10	0	0	0	0	0	0	0	0	0
1	2	6	2	0	0	0	0	0	0	0
2	0	4	3	3	0	0	0	0	0	0
3	0	2	1	6	1	0	0	0	0	0
4	0	1	0	4	1	4	0	0	0	0
5	0	0	0	3	2	1	1	2	1	0
6	0	0	0	1	0	3	2	2	1	1
7	0	0	0	0	1	3	0	4	0	2
8	0	0	0	0	0	1	0	7	0	2
9	0	0	0	0	0	1	1	1	0	7

(b) 测试集分类矩阵

图 12 训练集和测试集分类矩阵

使用 MultiBoost webpage[18] 提供的 AdaBoost. MH 算法，将训练集输入，经过 100 次迭代，得到强分类器，然后分别对训练集和测试集进行测试，准确率分别为 47.44%和 40%。经分析，类别之间的连续性是造成准确率不高的主要原因。分类矩阵如图 12 所示。

根据分类矩阵，训练集和测试集中分别有 63%和 76%的分类错误样本分布在其真实类别的附近，这是由于将渲染时间转换为类别的过程中，各类别之间相邻处的样本渲染时间较为相近，特征相差不大。为了解决该问题，提高分类准确率，将类别转换为时间范围的过程中，设置一个阈值 θ，将每个类别的时间范围向两侧扩大 θ 倍。同时，θ 不宜太大，预估时间范围过大将影响精细度，本文取 θ 为 0.9。将类别转换为时间范围后，根据信息文件得到每个场景的真实渲染时间，计算准确率，训练集和测试集的准确率分别达到 79%和 78%，分类效果如图 13 所示。其中，训练集和测试集最小时间分别为 3.152 秒和 2.858 秒，最大时间为 122.086 秒和 121.397 秒，每类时间段长为 34.86 秒。预测分类后，将信息文件中的场景名与预估时间范围对应，得到最后的预估结果，如表 4 所列。



(a) 训练集分类效果　　(b) 测试集分类效果

图 13　训练集和测试集分类效果

表 4　部分预估结果

场景名	最低预估时间(s)	最高预估时间(s)	实际渲染时间(s)
467	28.112	62.972	49.181
1052	28.112	62.972	56.397
1093	53.012	87.872	85.86
2359	53.012	87.872	91.399
1843	77.912	112.772	69.45
⋮	⋮	⋮	⋮

实验结果表明，使用本文方法能够提取出 Reyes 渲染算法中的有效特征，经过 AdaBoost. MH 算法的分类计算，能够对测试场景进行较为准确的时间预估，验证了该算法的可行性。

结束语　本文针对大规模集成渲染系统中的并行计算问题，提出基于 Reyes 渲染架构的时间预估算法，主要通过分析对渲染时间产生影响的各种因素，提取出有效的多维特征，并将渲染时间进行处理得到场景所属的类别，得到样本后，应用 AdaBoost. MH 算法进行训练得到支持多类单标签的强分类器，输入测试样本，得到样本的预测分类，进而获得样本的渲染时间预估范围。实验结果表明，使用本文提出的预估算法，能够获得较为准确的结果，为渲染任务的分配和调度奠定了基础，同时为用户的渲染费用估计提供了依据。

目前在该算法中只考虑了材质数量还未考虑其类型；细分面片数量的获取需要对 RIB 文件进行深度解析，花费时间较长。在下一步工作中，我们将针对这些问题进行更多的实验，测试更多的学习方法，并考虑 Reyes 之外的其他渲染架构，以进一步提高渲染时间预估算法的准确度和健壮性。

参 考 文 献

[1] Reinhard E, Jansen F W. Scheduling Issues in Parallel Rendering [C]// Proceedings of the First Annual Conference of the Advanced School for Computing and Imaging, May 1995: 268-277

[2] Chong A, Sourin A, Levinski K, et al. Grid-based computer animation rendering[C]// Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia, 2006. Kuala Lumpur, Malaysia: ACM 2006: 39-47

[3] Cook R, Carpenter L, and E. Catmull. The Reyes Image Rendering Architecture [J]. ACM SIGGRAPH Computer Graphics, ACM, 1987, 21(4): 95-102

[4] Apodaca A A, Gritz L. Advanced RenderMan creating CGI for motion pictures[M]. Francisco: Morgan Kaufmann Publishers, 2000: 137-137

[5] Litke A, Tserpes K, Varvarigou T. Computational workload prediction for Grid oriented industrial applications[C]// IEEE Internationale Symposium on The case of 3D-image rendering: Cluster Computing and the Grid, 2005, CCGrid2005. IEEE, 2005, 2: 962-969

[6] Doulamis N, Doulamis A, Panagakis A, et al. A combined fuzzy-neural network model for non-linear prediction of 3-D rendering workload in Grid computing [J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2004, 34(2): 1235-1247

[7] Doulamis N, Doulamis A, Panagakis A, et al. Workload Prediction of Rendering Algorithms in GRID Computing[C]// European Multigrid Conference. Hohenwart, Germany, 2002: 7-10

[8] 张自敏，樊艳英，陈冠萍. 改进的 BP 神经网络在地方 GDP 预测中的应用[J]. 计算机科学，2012，39(Z11)：108-110，127

[9] Pixar. The Renderman Interface[S]. Version 3.2.1 specification, San Rafael, California: Pixar, November, 2005: 4-48

[10] Mitchell T M. 机器学习[M]. 曾华军，张银奎，等译. 北京：机械工业出版社，2003：1-10

[11] Freund Y, Schapire R E. Adecision-theoretic generalization of on-line learning and an application to boosting[C]// Computational Learning Theory. Berlin: Springer-Verlag, 1995: 23-37

[12] Freund Y, Schapire R E. Experiments with a new boosting algorithm [C] // Machine Learning-International Workshop then Conference. San Francisco: Morgan Kauffman, 1996: 148-156

[13] Schapire R E. A brief introduction to Boosting [C]// International Joint Conference on Artificial Intelligence. San Franscisco: Morgan Kaufmann Publishers Inc, 1999: 1401-1406

[14] Wikipedia. AdaBoost[OL]. http://en.wikipedia.org/wiki/AdaBoost, 2013-04-15

[15] Schapire R E, Singer Y. Improved Boosting Algorithms Using Confidence-rated Predictions [J]. Machine Learning, 1999, 37(3): 297-336

[16] JFriedman J, Hastie T, Tibshirani R. Additive logistic regression: a statistical view of boosting [J]. Annals of Statistics, 2000, 28: 337-407

[17] Cortes R, Raghavachary S. The RenderMan Shading Language Guide[M]. Thomson Course Technology, 2007: 237-241

[18] Benbouzid D, Busa-Fekete R, Casagrande N, et al. MultiBoost: a multi-purpose boosting package[J]. Journal of Machine Learning Research, 2012, 13: 549-553