Nikolaos DOULAMIS, Anastasios DOULAMIS, Konstantinos DOLKAS,
Athanasios PANAGAKIS, Theodora VARVARIGOU and Emmanuel VARVARIGOS

National Technical University of Athens,
Department of Electrical and Computer Engineering
9, Heroon Polytechiou Str. Zografou 15773, Athens, Greece
ndoulam@cs.ntua.gr

NON-LINEAR PREDICTION OF RENDERING WORKLOAD FOR GRID INFRASTRUCTURE[1]

Abstract.
Grid Computing clusters a wide variety of geographically distributed resources. As a result it can be considered as a promising platform for solving large scale intensive problems. For this reason, it can be considered as one of the hottest issues in the computer society. A computational intensive application which can be gained from such a Grid infrastructure is rendering, a process dealing with creating realistic computer-generated image and with many applications ranging from simulation to design and entertainment. To implement, however, a rendering process in a Grid infrastructure is to perform prediction of its computational complexity. This is addressed, in this paper, by using several neural network modules, each of which is appropriate for a given rendering process. For this reason, initially, a feature vector is constructed to describe with high efficiency the parameters affected the complexity of a rendering algorithm. The feature vector is estimated by parsing a file on a RIB format. Then, prediction is performed using a neural network model. Prediction of three types of rendering algorithms is examined; the Ray Tracing, the Radiosity and the Monte Carlo irradiance analysis.

keywords : Grid computing, 3D rendering, neural networks

1 INTRODUCTION

Creating realistic computer-generated images is a very useful task for many fields and applications, such as simulation, design, research, education, entertainment and advertisement. Examples includes, training of air-planes pilots, designing of 3D objects, such

[1] This research work was supported by the European Union under the program of Information Societies Technology (IST) with grand No. IST-2001-33240, Grid Resources for Industrial Applications (GRIA).

as automobiles, or buildings or using molecular modeling in biological systems. Another interesting application is the entertainment worlds both in traditional animated cartoons and in realistic images for logos [1].

However, a fundamental difficulty in achieving total visual realism of synthetic images is the complexity of the real world. In a real environment, there are many surface textures, shadows, reflections and slight irregularities in the surrounding objects. Think of patterns on wrinkled cloth, the texture of skin, tousled hair, scuff marks on the floor and chipped paint on the wall. All these combine to create a "real" visual experience. The computational costs of simulating these effects are high; creating such pictures can take many minutes or even hours on powerful computers [2].

For this reason, computational Grids can be used to implement more efficiently a rendering algorithm [3]. This is due to the fact that Grid computing clusters wide variety of geographically distributed computational resources, including supercomputers, PC's, PDA's and workstations, and presents them as a single unified integrated resource. Grid computing has gained popularity in the last decade due to the rapid growth of the Internet as a medium for global communication and the development of fast hardware devices and sophisticated software. For this reason, it is now considered as one of the hottest issues in the computer society [4]. As a result, the rendering problems can be solved more feasibly and in a reasonably time and cost using a Grid infrastructure than a single resource supercomputing scheme.

In order to implement, however, a rendering algorithm to a grid infrastructure, two main issues should be addressed. The first refers to the parallelization of the rendering process, while the second to the way of allocating different tasks to the available resources. Parallelization of a rendering process can be performed in frame domain. In this case, each frame of the generated video sequence is handled independently from the previous frames and thus can be assigned to a different resource for processing. The second issue is solved by applying an appropriate scheduling scheme such as the Earliest Deadline First (EDF) [5]. However, scheduling algorithms requires prediction of the task workload. Thus one of the main issues that should be addressed before implementing a rendering algorithm to a Grid infrastructure is to perform prediction of its computational complexity.

Workload forecasting of a rendering process is an arduous task due to the fact that many parameters are involved in the process which affect in different way the computational cost. Although several predictors have been proposed in the literature, each of them is appropriate for a specific application. This is due to the fact an appropriate model should be developed and the inherent parameters, which affects the final outcome should be identified. According to the authors' knowledge no work has been proposed in the literature for workload prediction of a rendering process, which is an important issue for implementing such applications in Grid infrastructure.

In this paper several neural network modules each of which corresponds to a particular environment (i.e., rendering algorithm) is proposed to predict the computational complexity of rendering algorithms in creating real life synthetic world. In particular, initially a feature vector is constructed which includes appropriate parameters of the rendering process. Parameters have been obtained by analyzing the model used to create the rendered images. These parameters are extracted by parsing a RIB file format, which provides a general structure of describing a synthetic world. RIB format includes information about the object geometric primitives (such as cylinder, cone and sphere), object transformation, object material, number of light sources, rendering algorithm parameter so that any detail used for creating rendered images is specified. The extracted feature vector of the RIB file is used as input to a neural network architecture, which predicts the rendering workload. The use of the neural network architecture is due to the fact that the rendering parameters affect the

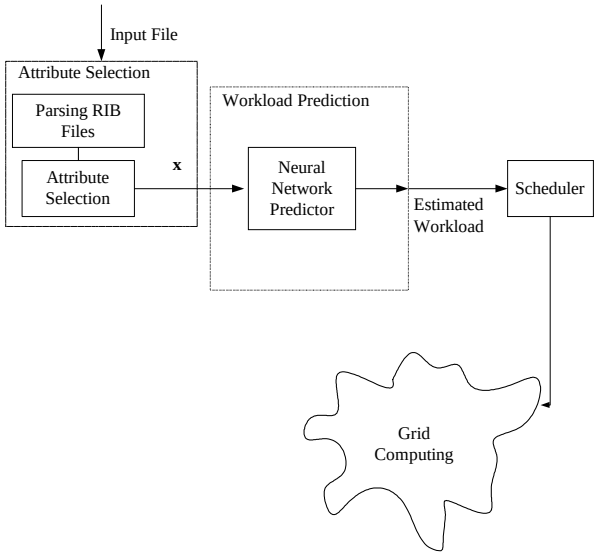


Fig. 1. System Overview.

computational cost in a non-linear and complex way. In our experiments three different rendering algorithms are investigated, the Ray Tracing, the Radiosity and the Monte Carlo irrandiance analysis, each of which are modeled by a different neural network architecture, since each rendering algorithm affects the computational complexity in a different way.

2 SYSTEM OVERVIEW

An overview of the proposed architecture used to predict the computational complexity of a rendering process is depicted in Fig. 1. As can be seen, the architecture consists of three main parts; the attribute extraction module, the workload prediction module and the finally the scheduler.

Attribute Selection Module: This module receives as input a file, which describes the 3D geometrical model (i.e., the synthetic environment), the algorithm as well as the respective parameters used for performing the rendering. The input file is encoded on the RIB format, which provides a general framework for describing the structure of a synthetic world and uses the minimum information required of interacting the rendering environment. In the output of the module, a feature vector of appropriate attributes is constructed to predict the rendering workload. The module consists of two subsystems. The first is responsible for parsing the input file (i.e., to identify the parameters that affect the rendering process and the algorithm used). In this way, a set of candidate attribute is constructed. The second is responsible for selecting the most significant attributes among all candidates.

Workload Prediction module: This module predicts the workload of a rendering process by taking into consideration the feature vector estimated from the previous module. For the workload prediction, a feed forward neural network is used, which relates the extracted parameters to the computational complexity of the rendering process.

Scheduler: The third module of the proposed architecture is responsible for scheduling a rendering task to the Grid infrastructure, by exploiting information provided by the workload prediction module. In this paper, we concentrate on the first two modules, while for the scheduler; conventional algorithms are used, such as the Earliest Deadline First (EDF) algorithm [5].

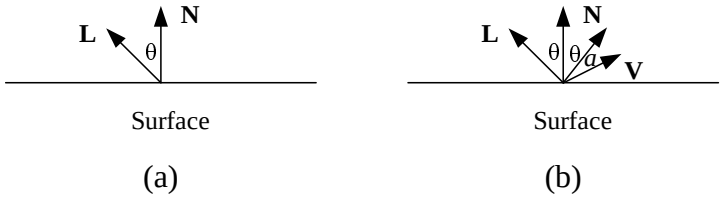


Fig. 2. (a) Lambertian Reflection and (b) Specular reflection.

3 RENDERING

Since the workload prediction of a rendering process depends on the specific algorithm used for creating the realistic word from the 3D geometrical primitives (models) and their respective parameters, it is necessary initially to briefly describe the Rendering algorithms used. Due to the complexity of the real word, several rendering techniques have been proposed in the literature, each of which is characterized by different properties and parameters. Some typical methods adopted are the Ray Tracing, the Radiosity and the Monte Carlo Irradiance analysis [1].

3.1 Ray Tracing

Although Ray Tracing has been initially proposed as an algorithm for determining the visibility of objects surfaces, it has been easily extended to rendering computer- generated images [1], [2]. In its simplest form, the luminosity of a pixel is defined using an illumination model at the closest intersection of eye ray with an object. For this reason, we initially describe the illumination models adopted and then we concentrate on the ray tracing algorithm.

A) Illumination Models

The illumination models define how a pixel or a surface is shaded (illuminated) based on the position orientation and surface material characteristics as well as the light sources illuminated them. A general illumination model is given by the following relation [1]

$$I = I_a k_a + I_p \cdot k_d \cdot \cos\theta + I_p \cdot k_s \cdot \cos^n a \qquad (1)$$

The first tem of equation (1), i.e., $I = I_a k_a$, corresponds to the ambient light. The ambient light impeaches equally on all surfaces from all directions The I_a factor is the ambient light intensity and k_a the *ambient reflection coefficient*, which is a material property [1].

The second term, i.e., $I_p \cdot k_d \cdot \cos\theta$, refers to the *Lambertian Reflection* also known as diffuse reflection. In this illumination model the brightness depends only on the angle θ between the normal vector of the surface **N** and the direction **L** to the li1ght source. This is illustrated in Fig. 2(a) [1].

The I_p is the point light source intensity and k_d the diffuse-reflection coefficient ranging from 0 to 1 and depending on the surface material.

The third term of (1), i.e., the $I_p \cdot k_s \cdot \cos^n a$ models the specular reflection, implemented by the Phong illumination model [6]. The variable *a* is the angle between the view point and the

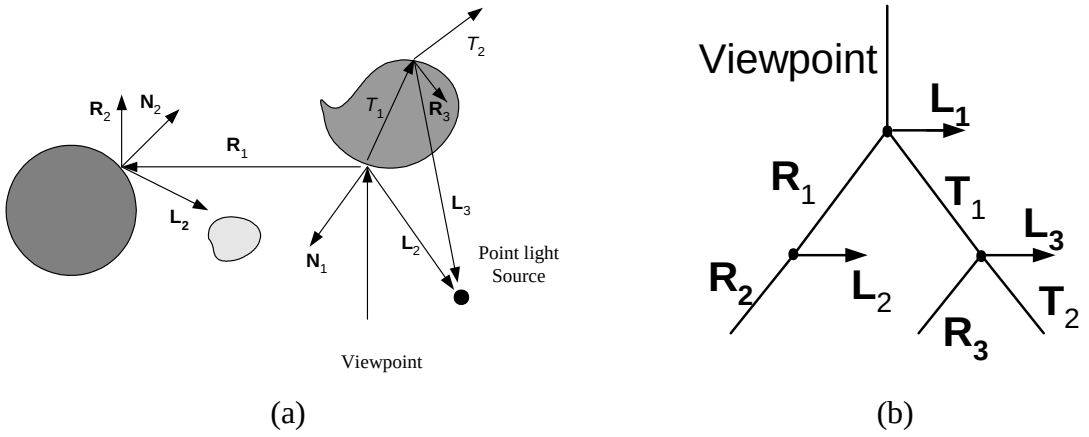


Fig. 3. (a) Rays recursively spawn other rays. (b)The ray tree constructed.

direction of reflection (see Fig. 2(b)). This rapid fall off is approximated by the factor $\cos^n a$, where *n* corresponds to the material specular reflection exponent. For a perfect reflector, *n* is infinite.

B) Recursive Ray Tracing

Recursive ray tracing takes into consideration the reflection and refraction of all objects in the scene and recursively estimates the luminosity intensity of a pixel [2]. In particular, reflection and refraction rays are considered along with the primary rays come from the eye. Shadows can be also included in the algorithm by firing an additional ray from the point of the intersection to each of the light sources.

Each of these reflection and refraction rays may, in term, recursively spawn shadow reflection and refraction rays. This is explained in Fig. 3(a), where rays recursively spawn another rays. In this way the rays form a ray tree such as that of Fig. 3(b). A branch is determined if the reflective and refractive rays fail to intersect an object, if some user's specified maximum depth is reached or if the system runs out of storage. The tree is evaluated from bottom to up and each node's intensity is computed as a function of each children's intensity [7], [8].

From the aforementioned analysis, it is clear that the basic parameters which affect a ray tracing scheme are the following; the number of light sources since each ray is related to a light source, and the depth of the tree (see Fig. 3(b)) used in the recursive implementation. Beyond a predetermined limit of reflections/ refractions the rays no further spawn into other rays. In addition, the computational complexity is also affected from the type of the material used, which determines the type of the illumination model.

3.2 Radiosity

Although ray tracing does an excellent job of modeling specular reflection, it still makes use of a directionless ambient lighting term to account for all other global lighting contributions. Approaches based on thermal engineering models for the emission or reflection of radiation eliminate the need for the ambient lighting term by providing a more accurate

treatment of interobject reflections. In particular, all energy emitted or reflected by every surface is accounting for by its reflection from or absorption by other surfaces. The rate at which energy leaves a surface called its radiosity and is the sum of rates at which the surface emits energy and reflects or transmits it from that surface to other surfaces [9], [10].

Imagine breaking up the environment into a finite number of discrete patches, each of which is assumed to be finite size, emitted and reflected light uniformly over each entire area. Then, if we consider each patch to be diffuse emitter and reflector we have that

$$B_i = E_i + \rho_i \sum_{1 \le j \le n} B_j F_{j-i} \frac{A_j}{A_i} \tag{2}$$

where B_i, B_j are the radiosities of the ith and jth patch. E_i is the rate at which light is emitted from the ith patch, ρ_i is the patch reflectivity and F_{j-i} is the form or configuration factor, which specifies the fraction of energy leaving the entirety of patch *j* that arrives at the entirety of patch *i* taking into account the shape and relative orientation of both patches and presence of any obstructing patches. Finally, A_i and A_j are the respective patch areas.

Based on the previous equation, we can estimate the interaction of light among all the patches in the environment, by solving a linear system of equations.

The finer the patch parametrization is, the better the results are at the expense of the increased computational time for n^2, where *n* is the number of patch. Other parameters which affect the computational load are the type of patches, that is if it is an emitter or transceiver surface, and the method used for estimating the Form factors.

4 MONTE CARLO IRRADIANCE APPROACH

Finite element radiosity has some big drawbacks, almost all of which are related to the fact that it has to pre-mesh the entire scene. First, it gets inaccuracies for some specific surfaces. Furthermore, it can use lots of time and memory when you have many geometric primitives, especially if your objects are made out of lots of little polygons or subdivision meshes.

The newer Monte Carlo irradiance approach has a different set of tradeoffs. Rather than enmeshing the scene and solving the light transport up front, the Monte Carlo approach is "pay as you go". As it is rendering, when it needs information about indirect illumination, it will do a bunch of extra ray tracing to figure out the irradiance. It will save those irradiance values, and try to reuse them for nearby points [11].

The Monte Carlo approach works just fine with almost all surfaces and trim curves. It does not unpack procedural primitives until they are really needed. It takes longer than radiosity for small scenes, but it scales better and should be cheaper for large scenes.

4.1 Other Factors Affecting Rendering Performance

A) Texture Mapping

Texture mapping is the technique to map an image, either synthesized or digitized on a surface. The image is called a texture map and each individual elements are often called *texels*. A simple approach starts by mapping the four corners of a pixel onto the surface. For bicubic patch, this mapping naturally defines a set of points in the surface coordinates. Then,

Table 1. An example of a RIB file format.

```
Projection "perspective" "fov" 40
Format 200 150 1

Option "render" "integer max raylevel" [4]
   WorldBegin

      Cylinder 1 -1 1 360
      Surface "shiny" "Kd" [0.2]

   WorldEnd
```

the pixels corner points in the surface coordinate space are mapped into the texture coordinate space [12], [13].

B) Shading

The simpler shading model for a polygon is constant shading, also known as faceted shading. This approach applies an illumination model once to determine a single intensity value that it is then used to shade the entire polygon.

Interpolating shading eliminates intensity discontinuities appear by constant shading. In particular, in this technique we assume that the normal for each mesh vertex is known. Alternatively, if the vertex normal is not stored and cannot be detected directive from the actual surface, it can be estimated by averaging the surface normal over all polygonal facets sharing the same vertex. Then, the vertex intensity is estimated using the vertex normal and appropriate illumination model. Finally, each polygon is shading by linear interpolation of vertex intensities along each edge and then between edges along each scan line [14].

5 ATTRIBUTE PARSING

One of the most important issue, affecting the prediction accuracy of the workload of a rendering process is the attributes used for creating the realistic synthetic images. These attributes are evaluated through an input file encoded by the RIB format which provides a general framework for describing a synthetic environment. For this reason, in subsection 5.1, we discuss the general concepts of the RIB format, while in subsection 5.2 we mention the most significant attributes and their respective organization.

5.1 The RIB format

The purpose of the RIB format is to provide the general structure of a synthetic world and to use the minimum information required of interacting the rendering environment. The RIB format provides the possibility of describing shape, geometric primitives (e.g., a cone, a sphere), object transformations (e.g., rotation, translation) surface material characteristics (and thus the respective illumination model), light sources (e.g., the number and the type), parameters of the rendering process (e.g, max number of patches in the radiosity method) and finally other rendering characteristics such as the type of texture mapping, the shadow

algorithm and so on. Using this information, we can describe any synthetic world and extract the basic parameters, which affect the prediction workload of a rendering algorithm.

Table 1 presents an example of a RIB file in which the synthetic world comprises only one cylinder. The surface cylinder is characterized by diffuse reflection. As a result, ambient and specular reflection are not represented (first and third term of (1)) in this particular example. The statement "*option*" includes rendering parameters which affect the whole synthetic world. For example, in Table 1, we define the depth level (see Fig. 3(b)) of the ray tracing algorithm. Similar to the "*option*" statement is the "*attribute*" statement with the difference that the latter refers to specific pieces of geometry instead of the entire rendered frame. Finally, the "format" determines the image resolution.

5.2 Attribute selection

As can be seen from the above, the RIB format provides a general framework for extracting and evaluating the basic attributes, which affect the workload of a rendering process. Table 2 presents some basic attributes, which can be estimated by parsing RIB format statements. In this table the parameters have been grouped into four different categories. The first refers to attributes which are independent from the rendering algorithm used, while the other three corresponds to each of aforementioned described rendering techniques (see section 2).

6 WORKLOAD PREDICTION

Since the extracted attributes affect the computational load in a non-linear and complex way, in our case, the rendering workload is modeled using a continuous non-linear function

$$y = g_c(\mathbf{x}),\ c \in \{\Pi_1, \cdots, \Pi_M\} \tag{3}$$

where vector **x** includes the rendering attributes (some of them presented in Table 2) and y is the respective computational cost. Index c of function $g_c(\cdot)$ corresponds to a particular type of rendering algorithm Π_i. This is due to the fact that different input-output relations are expected for different types of rendering methods since each of them uses different methodology for creating realistic images from the synthetic 3D models. In our experiments, M=3 different types of rendering algorithms have been evaluated; the ray tracing, the radiosity method and the Monte Carlo irrandiance since, as mentioned above, these are the most common used techniques for creating a synthetic world.

The main difficulty of applying equation (3) is that function $g_c(\cdot)$ is actually unknown. Modeling of function $g_c(\cdot)$ is performed through a feedforward neural network architecture, since it can approximate any non-linear function within any degree of accuracy ([12], pp. 208-213, 249). In our case, M feedforward neural networks are implemented, each of which corresponds to a specific environment, i.e. to a specific rendering algorithm. In this case function $g_c(\cdot)$ is approximated by

$$\hat{g}_c(\mathbf{x}) = \sum_l v_l \cdot u_l \tag{4a}$$

with

$$\mathbf{u} = [u_1, \cdots u_q]^T = \mathbf{f}(\mathbf{W}^T \cdot \mathbf{x}) \tag{4b}$$

Table 2. Some attributes used as inputs to the neural network predictor architecture.

Common Parameters	Ray Tracing	Radiosity	Monte Carlo Irrandiance
Frame Resolution	Maximum number of recursive rays	Number of radiosity steps	Maximum error metric
Number of Samples per pixel	Minimum shadow bias	Minimum number of samples per patch (Form Factors)	Number of rays used to estimate irradiance
Number of Light Sources	Object visibility (reflection, refraction)	Patch size	Maximum pixel distance (Forces recomputation)
Surface complexity (Number of meshes)	Surface shadow (none, opaque, shade)	Type of Radiosity calculations on surfaces (surface receives and/or shoots energy)	
Surface material (specular, diffuse reflection)			
Type of Texture mapping used			

where we assume one hidden layer of q neuron with activation function $f(\cdot)$ and a linear output neuron. Weights v_l connect the hidden neurons with the output ones, while u_l are the output of the hidden neurons. Matrix $\mathbf{W}$ includes the network weights of the hidden neurons, while $\mathbf{x}$ corresponds to the attribute (feature) input vector. Finally, $\hat{g}_c(\mathbf{x})$ indicates the approximate of function $g_c(\cdot)$ by the neural network architecture.

In order to estimate the network weights v_l and $\mathbf{W}$, a training set of N samples is used, which contains pairs of the form $(\mathbf{x}_i, y_i)$. Vector $\mathbf{x}_i$ includes the parameters of a specific rendering environment, while y_i corresponds to the respective workload obtained by these parameters. The training algorithm minimizes the mean squared value of the error for all samples in the training set

$$C = \frac{1}{2}\sum_{i=1}^{N}\{y_i - g(\mathbf{x}_i)\}^2 \tag{5}$$

A second order method has been used, in our case, for training the network based on a modification of the Marquardt-Levenberg algorithm. This method has been selected due to its efficiency and fast convergence, since it was designed to approach second order training speed without having to compute the Hessian matrix. To further increase the generalization performance of the network, the cross validation method has also been applied. Particularly, the available data are divided into two subsets; the one used for training and the one used for evaluating the network performance (validation set). The error on the validation set is monitored during the training process and when it increases for a number of iterations, the training is stopped.

Table 3. Average prediction accuracy over all experiments conducted for the three types of rendering algorithms.

	Ray Tracing	**Radiosity**	**Monte Carlo**
Average	8.668	10.31	8.10
Max	18.91	21.88	20.00

7 EXPERIMENTAL RESULTS

In this section, we evaluate the performance of the proposed neural network architecture for predicting the workload of rendering algorithms by applying several real life experiments. All experiments have been obtained by parsing RIB file format corresponding to synthetic images. The rendering engine that we have used is the BMRT2.6 for the Linux operation platform. BMRT supports RIB format to describe a synthetic world.

The three types of the rendering algorithms, described in section 2, have been used, i.e., the ray tracing, the radiosity and the Monte Carlo irrandiance analysis. For each rendering algorithms the attributes of Table 2 have been used along with the common rendering parameters, such as the image resolution, number of pixel samples, material type and so on.

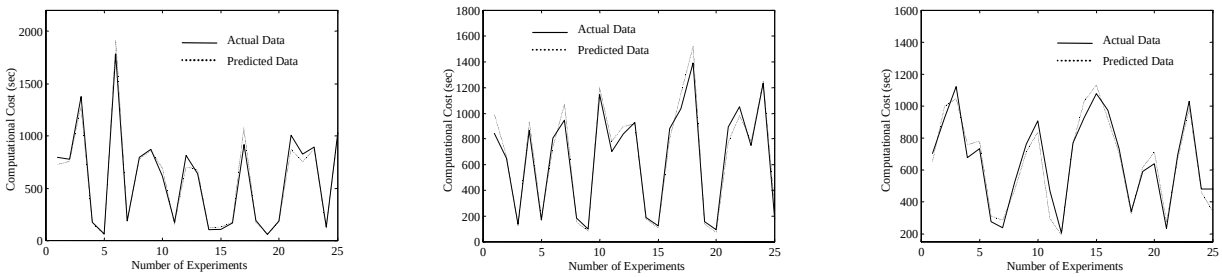
Fig. 4(a) presents the computational complexity for 25 different experiments obtained using the Ray Tracing rendering algorithm. In this figure, we have also added the predicted computational load as obtained by the respective neural network architecture. The respective results provided for the other two rendering algorithms are presented in Figs. 4(b,c). In all cases the computational cost has been measured on a PC AMD Athlon 1.60GHZ of physical memory 128MB of Linux Suse 2.4 operation system. In these Figures the computational cost has been normalized with respect to a reference RIB file. In our implementation, the images of Fig. 6 have been used as reference image. In all case a feedforward neural network has been used with one hidden layer, one output neuron of linear activation function and 15 hidden neurons. As can be seen good prediction accuracy is accomplished in all cases.

An alternative way to illustrate the prediction performance is to use the fractile diagrams or the Q-Q (Quantiles-Quantiles) plots. According to this method the actual cost is plotted versus the predicted ones. Therefore, perfect prediction lies on a line of 45^0 slope. The advantage of this method is that it shows all prediction differences with the same accuracy, regardless of the value of actual computational cost. It can be seen in Fig. 5 that the Q-Q plots are close to the line of perfect fit, meaning that the proposed model is good predictor of rendering computational complexity.

Fig. 6 presents the rendered images obtained using different parameters of the Monte Carlo irradiance method (see Figs. 6 (a,b)) and the Ray Tracing scheme (Fig. 6(c)) to show the affect of different rendering algorithm and parameters on the visual content of the final rendered image.

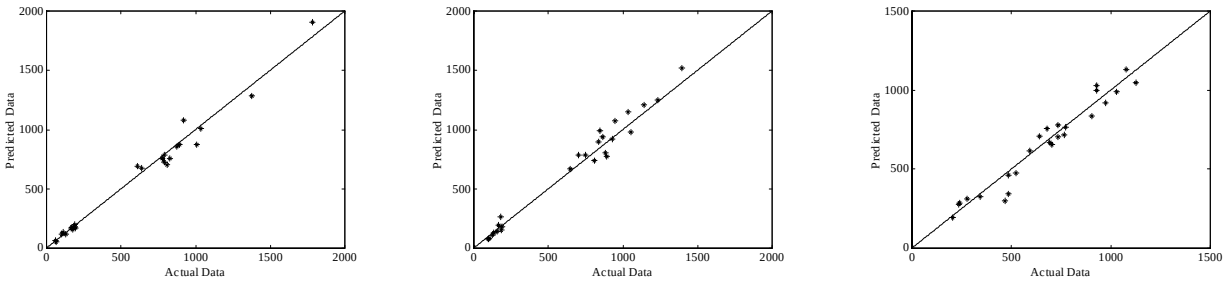
An objective measure for evaluating the prediction accuracy is to compute the relative prediction error with respect to the actual data E,

$$E = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{y_i}\times 100 \qquad (6)$$



(a) (b) (c)

Fig. 4. The actual and the predicted computational cost of various experiments for three different types of rendering algorithms. (a) Ray Tracing algorithm, (b) Radiosity algorithm and (c) the Monte Carlo irrandiance.



(a) (b) (c)

Fig. 5. The Q-Q (Quantiles-Quantiles) plots for (a) Ray Tracing algorithm, (b) Radiosity algorithm and (c) the Monte Carlo irrandiance.

(a) (b) (c)

Fig. 6. A synthetic world using three different rendering options. (a,b) The Monte Carlo irradiance algorithm. (c) The ray tracing algorithm.

where N is the total number of experiments and y_i, $\hat{y}_i$ the actual and the predicted computational cost at the ith experiment. Table 3 presents the relative prediction error for the three types of rendering algorithms. In this Table, we have also present the maximum error ($p = \infty$ - norm).

8 CONCLUSIONS

In this paper, we apply a neural network architecture for predicting the workload of a rendering process. Three different types of rendering algorithms have been investigated, the Ray Tracing, the Radiosity and the Monte Carlo irrandiance. For each rendering type a different neural network is constructed. The attributes used for the computational prediction are extracted by parsing a RIB file format, which provides an efficient and easy way to

describe a synthetic world. Simulation results illustrates a good prediction accuracy and therefore the proposed scheme can be used as input to a scheduling process, which assigns the rendering tasks to the Grid infrastructure.

REFERENCES

[1] J. D. Foley, A. van Dam, S. K. Feiner and J. F. Hughes, *Computer Graphics: Principles and Practice.* Second Edition in C. Addison Wesley, July 1997.

[2] A. Watt and M. Watt, *Advanced Animation and Rendering Techniques: theoru and Practice.* Addison Welsey, New york, 1992.

[3] Foster, I., and Kesselman, C. (editors), *The Grid:Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, USA, 1999.

[4] Leinberger W., Kumar V., Information Power Grid: The new frontier in parallel computing? *IEEE Concur.*, October-December 1999, 75-84.

[5] M. S. Fineberg and O. Serlin, "Multiprogramming for Hybrid Computation," *Proc. of IFIPS Fall Joint Computer Conference*, Washington DC, 1967.

[6] B.-T. Phong, " Illumination for Computer Generated Pictures," *CACM*, Vol. 18, pp. 311-317, June, 1975.

[7] A. Appel, "Some Techniques for Shading Machine Rendering of Solids," *SJCC*, pp. 37-45, 1968.

[8] T. Whitted, "An Improved Illumination Model for Shaded Display," *CACM*, Vol. 23, No. 6, pp. 343-349, June 1983.

[9] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile, "Modeling the Interaction of Light Between Diffuse Surfaces," *SIGGRAPH*, pp. 213-222, 1984.

[10] J. T. Kajiya, "The Rendering Equation," *SIGGRAPH*, pp. 143-150, 1986.

[11] T. Theoharis and A. Boehm, *Computer Graphics: Principles and Algorithms*, Papadamis Press, Athens, 1999.

[12] S. Haykin. *Neural Networks: A Comprehensive Foundation.* New York: Macmillan, 1994.

[13] E. Catmull, *A subdivision Algorithms for Computer Display of Curved Images.* PhD. Thesis, Computer Science Dep. University of Utah, Salt Lake City, Utah, Dec. 1974.

[14] J. F. Blinn and M. E. Newell, ""Texture and Reflection in Computer Generated Images," *CACM*, Vol. 19, No. 10, pp. 542-547, Oct. 1976.

[1] H. Couraud, "Continuous Shading of Curved Surfaces," *IEEE Trans. on Computers*, Vol. C-20, No. 6, pp. 623-629..