

A Data-Mining Approach to 3D Realistic Render Setup Assistance

C. Gonzalez-Morcillo,* L. M. Lopez-Lopez, J. J. Castro-Schez

Escuela Superior de Informatica
University of Castilla-La Mancha, Spain

B. Moser

Software Competence Center GmbH
Hagenberg, Austria

Abstract

Realistic rendering is the process of generating a 2D image from an abstract description of a 3D scene, aiming at achieving the quality of a photo. The quality of the generated image depends on the accuracy with which the employed render method simulates the behaviour of the light particles through the scene. According to the current practice, it is up to the user to choose optimal settings of input parameters for these methods in terms of time-efficiency, as well as image quality. This is an iterative *trial and error* process, even for expert users. This paper describes a novel approach based on techniques from the field of data mining and genetic computing to assist the user in the selection of render parameters. Experimental results are presented which show the benefits of this approach.

Introduction

Nowadays, almost every widely used rendering method tries to approximate the rendering equation defined by Kajiya (Kajiya 1986), in order to simulate the physical behaviour of the light particles through the scene.

Among all the available rendering methods, there is one group which is worth mentioning, that is *global illumination* algorithms (Goral *et al.* 1984) (Veach & Guibas 1997). These methods simulate the behaviour of light particles more accurately by considering the effects of both the *direct* and *indirect* illumination. This approach leads to results with a greater degree of photo realism and it is specially suited for difficult lighting configurations, such as indoor scenes, where the light comes from small-sized sources. However, these algorithms are computationally very expensive, so rendering of just one image may take from several minutes to hours, depending on the complexity of the 3D model and the established lighting parameters (*lighting design*). From the user's point of view, lighting design involves specifying values for the lighting variables that are taken as input by the rendering engine (e.g. number of samples per light or depth limit in ray tracing).

This paper introduces a novel, fully automatic approach (called *YAReW* standing for “*Yeah! A REndeR Wizard*”) to provide users with configurations of lighting values geared towards achieving comprehensible resulting images while reducing the required rendering time. This is managed by means of machine learning and artificial intelligence methods which search for structural patterns in a knowledge base. Thus, our approach tries to find that combination of lighting parameters values that maximises the relationship between the resulting image quality (I_Q) and the required render time (R_T).

The remainder of this paper is organised as follows. The following section overviews the state of the art and the current methods for lighting design. The next section describes the key points of *YAReW*. In the next section, empirical results are shown that have been obtained from different configurations. The final section offers a careful discussion and concluding remarks.

Related Work

Typically, two approaches are considered towards lighting design in realistic image synthesis. These are *direct design* and *inverse design*. According to the former, users must go through a trial-and-error process by which they must iteratively specify the values for the required lighting parameters, render the scene, evaluate the results, make modifications on the lighting design and so forth. Obviously, this approach is quite tedious and time-consuming. On the other hand, according to the inverse design paradigm, the user specifies the properties, objectives or constraints the resulting image must satisfy and the values for the lighting parameters are calculated automatically. Although this approach is certainly less tedious, it still requires users to know and express *a priori* the appearance of the results they expect to achieve. One of the main drawbacks of these is that they still require significant knowledge or expertise from the user (Shackel & Lischinski 2001). Besides, most existing systems search in a very limited space of the lighting design space.

The methods proposed by (Poulin, Ratib, & Jacques 1997) and (Poulin & Fournier 1992) infer the location of light sources from a specification of shadows and highlights provided by the user. In (Kawai, Painter, & Cohen 1993), the direction of light sources and reflectance of surfaces are estimated. This method does not change the location of

*Also research at Software Competence Center GmbH Hagenberg, Austria. This work has been funded by the Junta de Comunidades de Castilla-La Mancha under Research Projects “E-Pactos” (Ref. PAC-06-0141) and “Sarasvati” (Ref. PBC06- 0064-4504). Copyright © 2009, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

the light sources and is mostly suitable for diffuse scenes. (Costa, de Sousa, & Ferreira 1999) introduces a methodology in which the user defines and locates *fictitious luminaries* specifying the desired radiance distribution. Specifying the design goals and constraints appears to be a difficult task even for experienced users. In (Marks, Andalman, & Beardsley 1997), given a set of lighting parameters, the space of solution images is optimally dispersed in terms of perceptual quality. Then, the user browses the possible results and combines them in order to compose the desired solution. In (Shackel & Lischinski 2001), given a geometric model of a 3D object or scene, the material properties of the surfaces and the desired viewing parameters, the values of various lighting parameters are calculated by optimising a perception-based image quality objective function. Several works have been done in inverse lighting for interactive rendering, but the problem and the render algorithms are different to the methods used in realistic rendering. (Zupko & El-Nasr 2008) present a good survey of these approaches.

In these works, users are required to know how to handle different parameters or constraints which are presented to them by the system, in order to obtain a suitable lighting design for the model to render. The degree of difficulty when using such systems depends on the chosen approach, but always implies obstacles to users. *YAReW* tries to overcome those drawbacks by using a knowledge base where data about previously performed render processes drives our system towards discovering configurations of lighting parameters that are suitable for a particular 3D scene. *YAReW* works as a black-box from the user's point of view, as no previous knowledge of *lighting design* is required.

YAReW Approach

The problem of suggesting configurations of lighting values may be approached as a decision support problem (Bohanec 2003). Decision support approaches may be classified in *model-driven* and *data-driven*. Approaches based on the former require a model of the problem (e.g. rule-based expert systems), whereas approaches based on the latter consist in generic algorithms of which behaviour depends on specific data (e.g. decision trees or clustering techniques), which makes them more flexible. Data-driven methods need to perform a knowledge acquisition process, which may turn to be computationally expensive.

Our objective is to discover combinations of lighting values that are convenient to the user in achieving comprehensible image results while reducing the required render time. This may be managed by means of *supervised learning* techniques which need a set of well-known data instances in order to induce hypotheses which can generalise to describe unknown data. *YAReW* is a data-driven approach based on supervised learning techniques. Two basic components are needed (see Figure 1):

- A **knowledge base** containing a collection of known data instances (empirical data).
- A **learning kernel** responsible for offering decision support based on hypotheses induced from the data in the knowledge base.

Knowledge Base

The Knowledge Base is composed of a set of *key* 3D models. Each key model is stored along with some extra data:

- **Two direct illumination maps:** A direct illumination map is a black and white picture that reflects the direct impacts of light particles on the scene (see Figure 2.b and 2.c).
- **A time domain dataset:** A set of different render instances in which the goal attribute is R_T .
- **A quality domain dataset:** A set of different render instances in which the goal attribute is I_Q .

Both, the time and quality domain datasets share the same set of input attributes. The only difference between them is the goal attribute. The knowledge base contents generation is an iterative process composed of the following stages.

1. Domain Attributes Selection Each render instance is described through a set of attribute-value pairs. Such attributes must be those render parameters which have the strongest influence on both the render time and resulting image quality. Examples of such attributes may be the depth in Ray Tracing methods *RayDepth* or the number of samples per pixel taken in Shadow Quality *ShQual* (see Table 1)¹.

Since our interest is focused on estimating R_T and I_Q , the goal attributes in our domain are the *render time* and the *result quality*. Therefore, we handle two domains in which the only difference between them is the goal attribute (see Figure 1).

2. Bootstrapping This stage generates a collection of render instances for each key scene, which will later be used to create a training database for the learning methods. In this process, each render instance is generated by altering each value of the domain attributes with a randomly chosen value within the legal ranges of each specific attribute. Therefore, each key scene is submitted to the bootstrapping process, thus generating a collection of render instances for which the only unknown attribute value is that of the goal attributes: R_T and I_Q .

3. Rendering of instances In this stage, the collection of different render instances obtained in the previous stage are applied to the key scenes. Such key scenes are rendered several times, each one establishing one different configuration. The R_T and the resulting 2D image for each instance are obtained in this stage.

4. Image quality labelling Whereas the render time is an objective measure and it just need to be annotated, the image quality is a subjective one. A mechanism to automatically measure it is then needed. Our criteria to label the quality of an image is to label with better quality those images which has uniform distributions of hue, saturation and

¹The reader is referred to <http://www.yafray.org> for a reference to the render options and parameters available in the render engine used in the current implementation of *YAReW*. Anyway, *YAReW* is isolated from the render engine, and the set of attributes might be another.

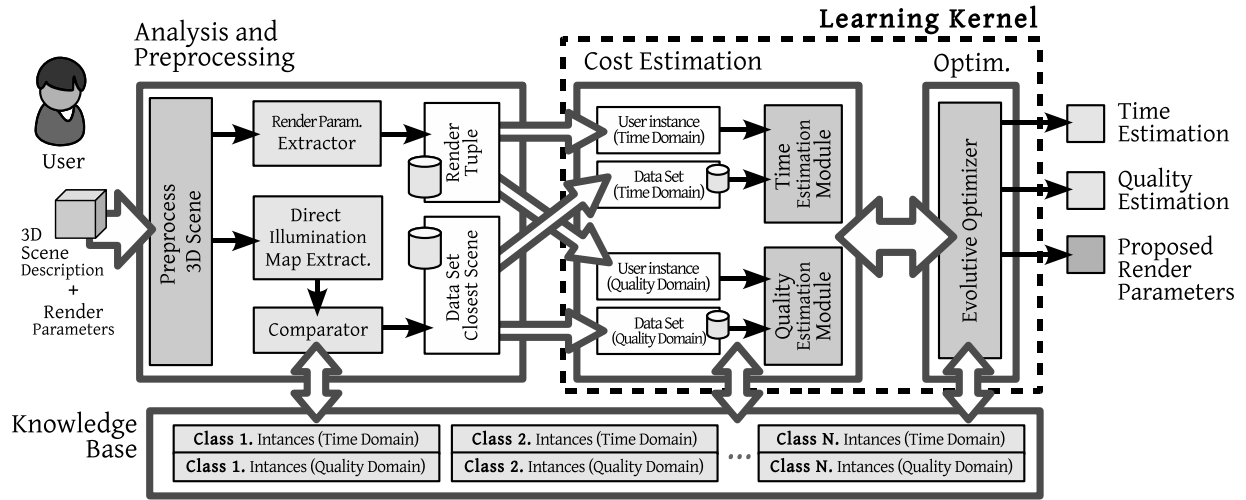


Figure 1: Representation of the main modules of the YAReW architecture.

brightness values and do not have illumination artifacts (in form of noise, see the corners of the room in Figure 2.e).

Let I be a set of n images and I_{ref} be an image which has been selected by an expert as the *best* image of the whole set ($I_{ref} \in I$) and it is labelled accordingly with the *best* quality. Any given image I_i , $1 \leq i \leq n$, in the set is labelled with a quality inversely proportional to the *distance* between I_i and I_{ref} . The employed *distance* measure is the *Root Mean Square Deviation* (RMSD). This measure returns a value which is an approximation to the degree of colour difference between two images. As the RMSD is a colour-based distance measure, it may cause problems in some situations. For example, a simple change in the brightness of an image may cause a big value of RMSD, but the quality of the image is just the same (the “noise” is the same). To help in solving this problem, the values of hue, saturation and brightness of the image being compared are adjusted before actually comparing it with the reference image. Such adjustment consists in distributing the histogram of the image being compared in a way such to get an histogram distribution similar to that of the reference image I_{ref} . Thus, when the comparison between both images is performed, they have similar histograms and consequently they have similar values of hue, saturation and brightness. This method is called **histogram matching** (Gonzalez & Woods 2002). Figure 2.(i) shows the comparison between Figure 2.(d) after applying histogram matching (left) with respect to I_{ref} (right).

5. Data Extraction and Codification At this point, we have achieved a set of render instances for which we already know the value of the goal attributes R_T and I_Q . In this stage, every render instance comprising the base of knowledge is codified as a row in a tabular file format. Each instance is described through a set of attributes which correspond to the columns of the table, being the goal attribute the last column of it.

6. Model Evaluation This stage aims at providing a quality measure of the data model just created. Such quality is

measured in terms of the degree in which the model allows the employed learning methods to induce hypotheses that describe well the underlying data and thus to be able to provide estimations about new, unknown data instances as precisely as possible.

As mentioned, we handle two datasets; one on the time domain and another on the quality domain. The R_T is measured in seconds and hence it is a continuous measure (a learning method suitable for estimation of numeric attributes must be used). On the other hand, the I_Q is labelled with a nominal label, hence it is a discrete measure (a learning method suitable for classification is needed).

The model evaluation is carried out by using the *k-fold cross validation* method, being $k = 10$. In each iteration, 90% of the data instances are used to train the learning algorithm and the remaining 10% is used to test its performance. The time estimator performance is measured using the *mean squared error* (MSE), whereas the number of wrongly classified instances is used in the case of the image quality classifier.

When this stage finishes, performance of another iteration of the content generation process may be considered to refine some aspects of the resulting model.

Wizard System Architecture

When the user inputs a new scene, it is first analysed by the components in the **Analysis Module** (see Figure 1). The goal of this first level is to extract all relevant data from the user’s scene and to make up a data instance of which goal attributes will later be estimated. This first level is also responsible for finding which scene in the knowledge base is most similar to the user input scene. Once the process has passed through the user’s scene analysis level, the next level, called **learning kernel**, estimates the render time and the image quality which are to be achieved by the render configuration and the scene provided by the user. Besides, this module also searches for a more suitable render configuration.

Analysis Module Inputs to this level are the user 3D model and a user established configuration of render option values. The main goal of this module is to extract all the relevant data from it to create a render instance configuration. Is easy to support in this level any graphic development software by means of different front-end modules to extract data.

Moreover, in this level the system tries to find that scene in the knowledge base which is most similar to the user scene. To do this, we have used an artifact called **direct illumination map**. We define a direct illumination map as a 2D image that represents the distribution of light particles which impact on the scene and directly come from the light sources (see Figure 2 a, b and c). These artifacts provides us with an estimation of the illumination model of the given scene, as a consequence of the direct action of the light sources over the scene. The need for having two direct illumination maps for every scene is to be able to compute a relationship between the amount of illumination in the scene which comes from indoor and outdoor located light sources.

The extraction of direct illumination maps from a scene is a non-trivial and computationally complex process. It involves a ray tracing procedure which simulates the distribution of light particles through the scene.

The first stage in this process is the **extraction of geometry data** from the 3D model. The relevant information comprises the following elements:

- **Set of polygons:** Each one of them is described through its set of points and its normal vector.
- **Set of light sources:** Each one is described through its location point, the type of light source, its name and its location with respect to the scene (indoors or outdoors).
- **Virtual camera:** It is the point of view of a virtual observer. Actually, it is not needed to perform the ray tracing process, but it is useful to optimise that process. The light impacts outside the view volume are ignored.

The next stage is indeed the direct illumination ray tracing over the reconstructed geometry of the given model. Actually, this process is a simplified ray tracing in which rays are traced from the light sources to some random direction, according to the specific type of light source. The result of this stage is a list of hit points. A hit point is produced when a ray is casted from a light source and it impacts on the surface of any object defined on the scene. Once this stage is finished, two different lists of hit points are obtained: one with those hit points caused by indoor-located light sources and another with those hit points caused by outdoor-located light sources. These hit points are represented in 2D images (maps) with a fast 3D rasterization. The illumination maps are used to select the scene in the knowledge base which is most similar to the current one. Each scene in the knowledge base is stored along with its own direct illumination maps. The employed measure is the *root mean square deviation* (RMSD). The scene in the knowledge base which produces a lower difference value when compared with the user's input scene is selected, hence its associated training datasets are afterwards used by the decision support engine.

Learning Kernel There are mainly two objectives in this level: to provide the user with R_T and I_Q estimations on the input scene, and to offer decision support to the user in form of render option tuples which maximise the relationship I_Q/R_T (this relationship is the *utility function* to be maximised).

There are three key components in this level; the R_T *Estimation Module* and the I_Q *Estimation Module* handle the processes of estimating these parameters and the *Evolutive Optimizer*, which is responsible for searching for possible combinations of render option values that maximise the utility function (see *Learning Kernel* component in Figure 1).

Quality Estimation Module Conversely to the render time case, the image quality is a discrete measure, labelled with categorical values. Hence, learning methods suitable to estimate categorical values or *classes* must be used. Currently the learning methods supported by the system to classify the image quality are Quinlan's C4.5 (Quinlan, J.R. 1993) and ID3 (Quinlan 1986), to induce classification trees and k -nearest neighbours, supporting *Euclidean* and *Manhattan* distance measures.

Time Estimation Module Since the render time, as conceived in this particular case, is a continuous measure, learning methods to be used need to be suitable for estimating numerical values. Specifically, the regressors used in the current version of the system are a variation of Quinlan's C4.5, where the splitting measure used to induce the tree is the mean squared error (MSE), and k -nearest neighbours method, employing *Euclidean* or *Manhattan* distance measures.

Evolutive Optimizer The problem of suggesting *good* render values may be seen as a problem of searching the *best* solution from space of possible solutions. Each possible render values configuration corresponds to a particular solution to this problem. The *fitness* of any given solution is given by the defined utility function, hence the search process must go through every possible solution trying to maximise the value of the utility function and return that solution which maximises it. This component uses a genetic algorithm, starting the search process with a population of possible solutions, each one returning a specific value of the utility function.

The genetic search method in use proceeds in several iterations that performs the usual cross between two randomly selected individuals, the mutation operation (with small probability value) and the calculation of the fitness value using the utility function. The length of the population keeps constant during the search process. The algorithm finishes either when a given number of iterations have been executed or when the value of the utility function of the best individual in the population is greater than some threshold.

Experimental Results

These results have been generated using the current implementation of *YAReW*, available at <http://code.google.com/yarew09/> under the GNU GPL Free Software License. These results are based on a computer with an Intel Dualcore 2 Ghz processor and 3GB RAM run-

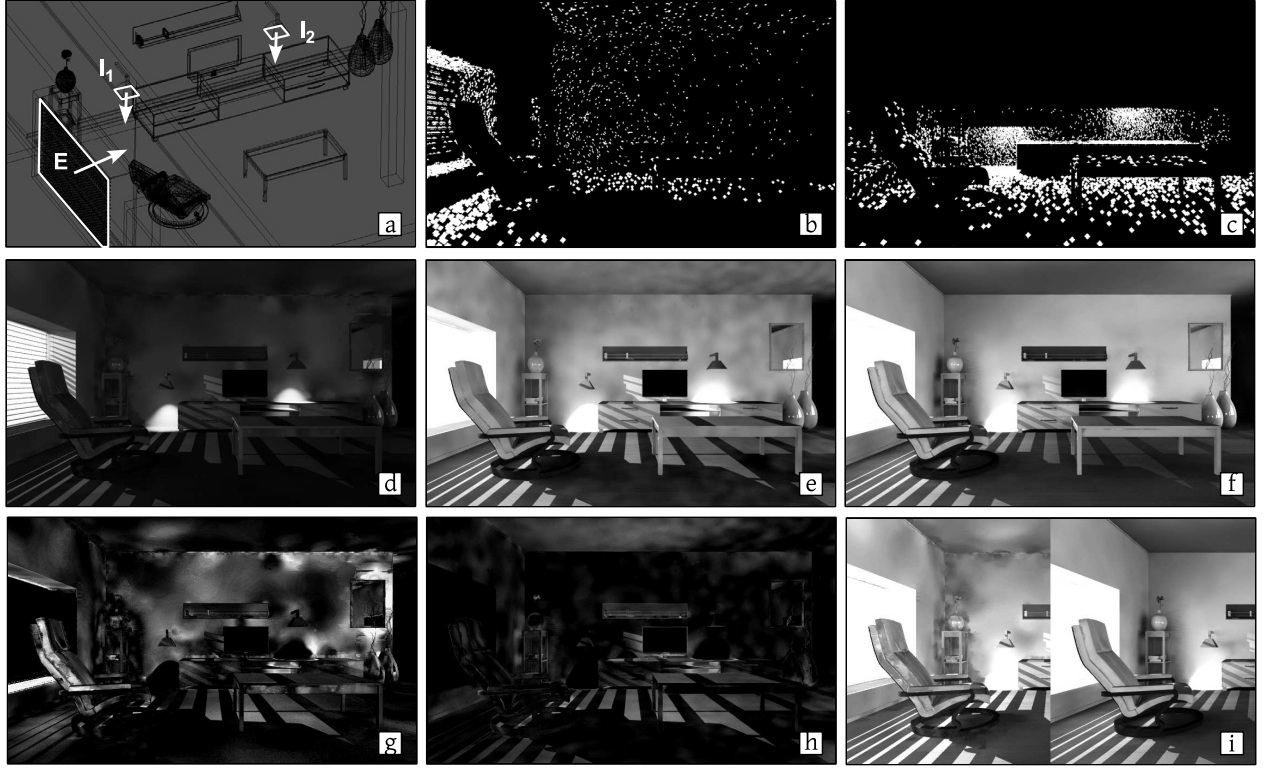


Figure 2: Case of Study: (a) Position of the external light source E and the two internal light sources I_1 , I_2 in the scene. (b) Direct illumination map of the external light source. (c) Direct illumination map of the internal light source. (d) Result with *low* optimization level. (e) Result with *medium* optimization level. (f) Result with *high* optimization level. (g) Difference between *low* (c) and *high* (f) optimization level (7.1%). (h) Difference between *medium* (d) and *high* (f) optimization level (4.8%). (i) Visual comparison between (d) and I_{ref} after histogram matching.

OptLevel	RayDepth	OSA	GIQual	Depth	PCount	PRad	PMix	ShQual	Prec	ShRef	EPower	GIPower
Low	36	8	Low	1	273863	33.7	454	0.31	39	0.56	0.23	13.54
Medium	12	5	High	1	100927	1.9	193	0.18	44	0.75	0.58	11.13
High	12	8	High	6	498776	1.9	384	0.06	52	0.75	0.58	11.13

Table 1: Configurations of render parameters suggested by different optimization levels.

ning Debian OS and Yafray 0.0.9 (<http://www.yafray.org>) (Pathtracing - PhotonMapping) as the chosen render engine. The indoor scene to be rendered contains 88482 faces and three light sources; two internal lamps and one external near a window (see Figure 2.a).

The scene was optimised using three optimisation levels: *Low*, *Medium* and *High*. As shown, this value configures the number of iterations in the genetic optimisation algorithm. Because of the random nature of algorithm (Michalewicz, Z. 1996), the results obtained with the same input scene might be different in different executions. In Figure 2.d, 2.e and 2.f, three concrete images with *Low*, *Medium* and *High* are shown. The corresponding configurations obtained for these images are represented in Table 1. In different optimisation levels, some fragments of the proposed configurations are maintained (such as *EPower* and *GIPower* between Medium and High). This is due to the good fitness value of this con-

figuration for this class of scenes.

The difference between the best result (Fig. 2.f) and the other two (Fig. 2.d and 2.e) is calculated by RMSD in Fig. 2.g and 2.h, respectively. In these figures, the white colour represents the bigger difference and the dark zones represents the smaller difference. Note that the lighter zones are situated in the most difficult places to be illuminated. As mentioned, this difference is calculated after the histogram matching is applied (see Figure 2.i). The result with Low optimisation level (Figure 2.d) shows a noisy image in the corners of the room (there is a difference of 7.1% between these images). The difference is smaller with the Medium optimization level (4.8%).

The prediction of the render time (T_{Est}) and the estimated render quality (Q_{Est}) are represented in Table 2. The time is represented in seconds. After 10 tests with 3 different scenes, the mean error in time estimation is around 27 sec-

Opt Level	T_{Est}	T_{Real}	Q_{Est}	Q_{Real}
Low	118	95	8	7
Medium	171	147	8	8
High	126	132	9	9

Table 2: Predicted and real values of the render time and quality of the result.

onds. Besides the coarse approximation, this time is a good initial estimation (other techniques such as *profiling* also has a big error) and could be improved using more information in the knowledge base (see Conclusion section). The prediction of I_Q estimates the value correctly in 87.5% of the cases.

As a final remark, note that optimization may result in different results by the nature of genetic approach. But usually, the results with High optimization level are very close. The mean differences between them are less than 1%.

Discussion and Conclusion

The geometrical configuration of some scenes, where the light comes from small holes (e.g. in indoor scenes) or small-sized light sources, makes difficult to find correct values for the render variables involved in lighting design, even for expert users. Inverse lighting does not avoid these difficulties because it is necessary to specify the constraints and other parameters presented by the system. Also, these methods have a strong dependency on the render method that is being used. *YAReW* has been developed in response to this challenge.

The data-driven model presented in this work, based on supervised learning techniques and genetic optimisation, offers several desirable features which together makes this approach unique and of highest practical value:

- It is independent on the knowledge of the user. *YAReW* only needs the input scene to provide users with a light design model.
- The system provides a configuration that maximises the I_Q/R_T ratio. A numerical estimation of these factors are also available. The output of *YAReW* is obtained in few seconds and it avoids the iterative *trial-and-error* process of direct lighting design.
- *YAReW* can be automatically adapted to other render methods (which involves a different set of render variables) and different machine learning methods (Castro, Castro-Schez, & Zurita 1999). The pipeline design of the architecture provides an easy adaptation for such extensions. Different render instances can be stored with different set of attributes (used in different domains).
- The system is developed using free software, running in the main platforms (Windows, Linux and MacOSX).

It is very promising to extend *YAReW* towards a better generation of the knowledge base. Its contents are critical for the performance of the system. One line of our current work analyses the geometric and internal characteristics of

the scenes to delimit the possible values of each parameter. Future work also concentrates on using an hierarchical approach, taking into account more parameters such as the properties of the materials, the geometry of the objects in the viewport, etc.

References

- Bohanec, M. 2003. *Data Mining and Decision Support: Integration and Collaboration*. Springer. chapter 3, 23–36.
- Castro, J.; Castro-Schez, J.; and Zurita, J. 1999. Learning maximal structure rules in fuzzy logic for knowledge acquisition in expert systems. *Fuzzy Sets and Systems* 101(3):331–342.
- Costa, A. C.; de Sousa, A. A.; and Ferreira, F. N. 1999. Lighting design: A goal based approach using optimization. In *Rendering Techniques*. Springer-Verlag. 317–328.
- Gonzalez, R., and Woods, R. 2002. *Digital Image Processing*. Prentice Hall, 2nd edition. ISBN-13: 978-0201180756.
- Goral, C. M.; Torrance, K. E.; Greenberg, D. P.; and Battaile, B. 1984. Modeling the Interaction of Light Between Diffuse Surfaces. *Computer Graphics* 18(3):213–222.
- Kajiya, J. 1986. The Rendering Equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, 143–150. ISSN: 0097-8930.
- Kawai, J. K.; Painter, J. S.; and Cohen, M. F. 1993. Radiotimization goal-based rendering. In *Computer Graphics Proceedings*, Annual Conference Series, 147–154.
- Marks, J.; Andalman, B.; and Beardsley, P. A. 1997. Design Galleries: A general approach to setting parameters for computer graphics and animation. In *SIGGRAPH'97 Proceedings*, 389–400. Addison Wesley.
- Michalewicz, Z. 1996. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 3rd edition. ISBN: 978-3-540-60676-5.
- Poulin, P., and Fournier, A. 1992. Lights from highlights and shadows. In D. Zeltzer., ed., *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, 31–38.
- Poulin, P.; Ratib, K.; and Jacques, M. 1997. Sketching shadows and highlights to position lights. In *Proceedings of Computer Graphics International 97*, 56–63.
- Quinlan, J.R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann. ISBN: 1-55860-238-0.
- Quinlan, J. R. 1986. Induction of Decision Trees. *Machine Learning* 1(1):81–106. ISSN: 0885-6125.
- Shacked, R., and Lischinski, D. 2001. Automatic Lighting Design using a Perceptual Quality Metric. In A. Chalmers, and T.-M. Rhyne., eds., *Computer Graphics Forum*, volume 20, 215–226. Eurographics Association.
- Veach, E., and Guibas, L. J. 1997. Metropolis Light Transport. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, 65–76.
- Zupko, J., and El-Nasr, M. 2008. A tool for adaptive lighting design. In *ACM Symposium on Video games*, 135–142.