A QUALITATIVE EXPERT KNOWLEDGE APPROACH TO RENDERING OPTIMIZATION

D. Vallejo-Fernandez, C. Gonzalez-Morcillo and L. Jimenez-Linares
Escuela Superior de Informatica, Universidad de Castilla-La Mancha, Paseo de la Universidad 4, Ciudad Real, Spain

Keywords: Rendering, Qualitative Expert Knowledge, Fuzzy Systems, Fuzzy Sets, Multi-Agent Systems.

Abstract: The rendering process allows the developer to obtain a raster 2D image from the definition of a 3D scene. This process is computationally intensive if the source scene has a certain complexity or high-quality images are required. Therefore, a lot of time is spent and many computational resources are needed.
In this paper, a novel approach called QUEKARO (standing for a QUalitative Expert Knowledge Approach to Rendering Optimization) is presented for adjusting some relevant parameters involved in the rendering process by using expert systems. This way, the developer can obtain optimized results which reduce the time spent in the rendering process and, in most cases, do not affect the final quality of the raster 2D image. These results will be exposed on the result section, in which different optimizations will be studied.
As we discuss on the final section of this paper, the use of expert systems in the rendering process involves a novel approach which reduces drastically the resources used and provides us with a high-scalable system. Using these arguments, we will justify the inclusion of expert systems in this area and will study future works.

1 INTRODUCTION

Fotorrealistic image synthesis can be interpreted as the process which pursues the creation of synthetic images which cannot be distinguished from the images obtained from the real world. This process is divided into several phases such as modelling, setting materials and textures, placing the virtual light sources, and, finally, rendering.

The latter lies basically in generating a 2D image from an abstract description, involving the geometry of the scene, the definition of lights, the camera positions, and the use of materials. Rendering is usually the most computationally intensive phase of the whole process and, therefore, it takes a long time to be done. In addition, if the scene to be rendered is complex or high-quality realistic images are required, such process involves a lot of time.

To overcome this problem, the rendering phase has experimented an important evolution. First, researchs tried very hard to solve basic problems, such as the detection of visible surfaces or the basic shading. Time passed and different rendering algorithms were developed, ranging from simple and fast to more complex and accurate, which simulate the light behaviour faithfully. Such methods are usually classified into two main categories: local and global illumination algorithms. Kajiya pointed out all rendering algorithms aim to model the light behaviour over various types of surfaces and try to solve the rendering equation (Kajiya, 1986), which forms the mathematical basis of all rendering algorithms.

As a result of the huge amount of time spent in the rendering phase, it is often considered a bottleneck in photorealistic projects. In fact, the generation of a single high-quality image may take several hours up to several days, even on fast computers. In addition, the user normally sets the rendering parameters so that the final rendering time increases without obtaining substantial quality improvements, that is, the user tends to optimize this process excessively.

This paper presents a novel approach called QUEKARO based on the use of fuzzy systems to optimize the rendering phase. This way, final results can be improved by entities which use qualitative expert knowledge over a distributed platform. With this design we have obtained a robust, scalable, and flexible system which allows the final user to optimize the rendering phase without lossing perceptual quality of the resulting image.

This paper is structured as follows. First, Section 2 overviews the state of the art related to rendering optimizations. Section 3 describes QUEKARO, discussing the underlying architecture and studying in depth the fuzzy system used in QUEKARO. Section 4 shows empirical results obtained with this approach by using different configurations. Finally, section 5 resumes main conclusions and suggests future works.

2 RELATED WORK

The huge amount of time required by the rendering process implies all related optimizations pursue the reduction of the final time spent. These optimizations can be classified into three general types (or into a mix of them) (Sundstedt et al., 2005):

1. Optimizations related to the use of powerful processing elements (hardware optimizations).
2. Optimizations related to the use of distributed systems (pararellel/distributed optimizations (Chalmers et al., 2002)).
3. Optimizations related to the use of multi-agent systems.

The first type of optimization is related to the use of powerful processing elements. This way, it is worth pointing out that some researchers use programmable GPUs (Graphics Processing Units) so that these powerful processors run several portions of code of the rendering algorithm. The main problem of this approach is the low-abstraction level used, that is, the algorithms must be specifically designed for each hardware architecture.

The second type of optimization is based on the 'divide and conquer' principle. If we adapt this basic principle for rendering, we can employ several computational units to reduce the final calculation time. There are many related approaches in this research line, such as the work made by Fernandez-Sorribes et al. (Fernandez-Sorribes et al., 2006), which uses a grid architecture for distributed rendering. However, most of them are not able to achieve effective load balancing and, therefore, the final time required depends on the most complex task.

The third type of optimization is based on the use of multi-agent systems. In this field, different proposals have been presented, such as the work exposed by Rangel-Kuoppa et al. (Kuoppa et al., 2003). However, this approach does not make use of the multi-agent technology because there are not interaction mechanisms among agents. Another related work is proposed by Schlechtweg et al. (Schlechtweg et al., 2005), which makes use of a multi-agent system for rendering artistic styles such as stippling and hatching.

2.1 Comparison to the Quekaro Approach

The approach described in this paper is based on a fuzzy system. This way, we can represent the qualitative expert knowledge by using a set of linguistic rules, which allows each computational unit to adjust a number of settings related to the rendering process. Throught these adjustments, final render times are reduced by solving the problem related to the user's tendency to use rendering parameters that are not optimum, because these adjustments increase the rendering time without obtaining important quality benefits.

QUEKARO has been satisfactorily used in MAgArRO, standing for a Multi-Agent Architecture for Rendering Optimization. This multi-agent system (designed under the principles of the set of FIPA standards (Foundation for Intelligent Physical Agents FIPA, 2002)) is composed of an indeterminate number of specialized agents, which compete for the purchase of work units which make up the complete work. A work unit can be interpreted as a part of a scene or as a frame in an animation.

The approach used in QUEKARO implies several important improvements regarding other research lines:

- It supplies us with a high-level of abstraction due to the use of a fuzzy system.
- The descriptive power of the fuzzy system used allows the expert to model the knowledge relating to rendering easily (Tanaka, 1998).
- The use of a fuzzy system combined with a multi-agent system permits each agent to employ different models of expert knowledge. This way, we could employ different specialized agents by using different sets of rules.
- It complements the use of expert knowledge in hardware or paralell rendering based alternatives.

3 PRESENTATION OF THE QUEKARO APPROACH

As we mentioned in previous sections, we have used a fuzzy system to optimize relevant parameters involved in the rendering process. The choice of sets of rules as the method for modelling the qualitative expert knowledge allows us to describe and extend this knowledge easily. This way, the system allows the user to incorporate different rendering methods, such as Raytracing or bidirectional Pathtracing, each one modelated by using a different set of rules. In this paper, we will study the set of rules made for the Pathtracing rendering method. Therefore, we will have several entities specialized in this method. The choice of this method is because it is one of the most used, owing to its good rendering time-quality ratio.

Next, we will study the different elements which compose QUEKARO. First, we must specify how the rendering parameters are adjusted by using QUEKARO. This is done by obtaining the values related to the output parameters of the fired rules (the consequents) and by using a method to adjust the rendering parameters correctly. Second, we must define how these rules are fired. This is done by matching the input parameters with the antecedents of the rules. Last, we must define the set of rules which governs the QUEKARO behaviour. All these elements will be studied carefully.

The output parameters, which repay with the consequents of the rules, are configured to decrease the final time required to finish the rendering process without obtaining important losses of the perceptible quality. The linguistic labels used are VS (very small), S (small), N (normal), B (big), and VB (very big) (Zadeh, 1975). Next, we will explain the meaning of these parameters:

- Recursion Level (Rl): this parameter is defined over the linguistic labels {VS, S, N, B, VB}. It refers to the global recursion level in Raytracing, that is, the number of light bounces.
- Light Samples (Ls): this parameter is defined over the linguistic labels {VS, S, N, B, VB}. It represents the number of samples per light on the scene.
- Interpolation Band Size (Ibs): this parameter is defined over the linguistic labels {VS, S, N, B, VB}. It defines the interpolation band size in pixels among adjacent work units, as shown in Figure 1. Moreover, this parameter has a relevant importance because it is used in the final composition process of the image.

We have used classic trapezoidal functions to define the membership of fuzzy sets related to the output variables, as we can see in Figure 2.

The output parameters defined (recursion level, light samples, and interpolation band size) have a strong dependency with the Pathtracing rendering

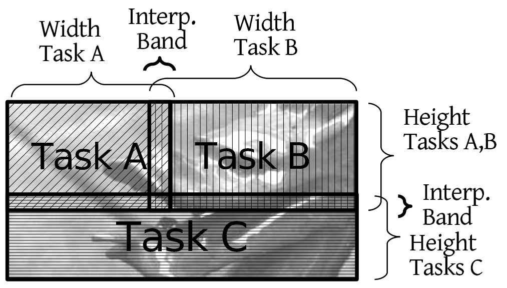

Figure 1: Definition of the interpolation band size between tasks.

method. However, the parameters related to the antecedents of the rules can be used with other rendering methods, depending on their characteristics. We will only need to change the descriptions of the rules. Most of these parameters are directly related to the characteristics of the scene:

- Complexity (C): this parameter is defined over the linguistic labels {VS, S, N, B, VB}. It represents the complexity-size ratio of the work unit. This complexity is calculated depending on parameters such as the materials used to define the elements which compose the model.
- Size (S): this parameter is defined over the linguistic labels {S, N, B}. It refers to the size of the work unit in pixels.
- Neighbour Difference (Nd): this parameter is defined over the linguistic labels {VS, S, N, B, VB}. It represents the complexity difference among the current work unit and its neighbour work units. To take this difference into account is important not to make quality differences among neighbour work units out.
- Optimization Level (Op): this parameter is defined over the linguistic labels {VS, S, N, B, VB}. Its value is selected by the user, and it allows him/her to apply different levels of optimization (more or less agressive than the user predefined optimization).

The definition of the fuzzy sets related to the input variables, with the exception of the optimization level, is made dynamically, that is, in run-time. The intervals of these sets are defined depending on the characteristics of the input scene and are made by linear distribution. For example, different intervals of the complexity variable will be created depending on the complexity of the scene. This way, we can focus on each problem independently. Another important question is related to the influence of the output variables in relation to the final rendering parameters. When we defuzzificate and obtain crisp values for the recursion level and the light samples, we do not use

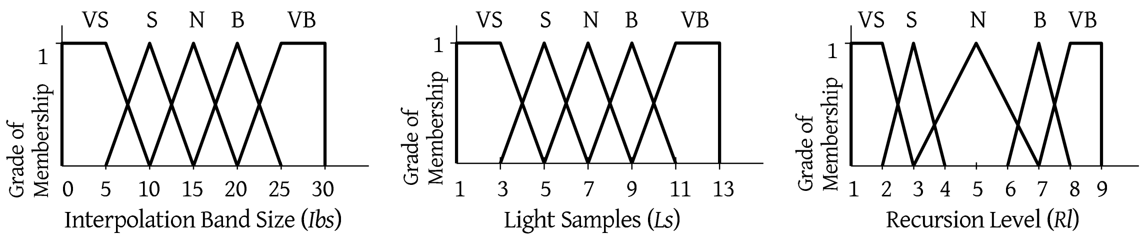

Figure 2: Definition of the output variables.

this values directly in the rendering. On the contrary, we weight these obtained values with the values predefined by the user. This way, we can apply local optimizations so that the final result does not show important losses of the perceptible quality. In fact, this is essence of QUEKARO, that is, to optimize the rendering process depending on the values defined by the user.

Once we have studied the different variables, the next step is to present the set of rules. We have used if-then rules due to their resemblance to human reasoning. This set, which shows an expert's knowledge in Pathtracing, has 28 rules, but we will study some of them, which represent the main types of rules used:

1. If C is {B, VB} and S is {B, N} and Op is VB then Ls is VS and Rl is VS.
2. If C is {S, VS} and S is {B, N} and Op is VB then Ls is S and Rl is S.
3. ...
4. If C is {B, VB} and Nd is B then Ls is VB.
5. If C is {B, VB} and Nd is N then Ls is B.
6. If Nd is VB then Ibs is VB.
7. If Nd is B then Ibs is B.
8. ...

The first rule takes into account parameters related to the complexity of the scene together with the optimization level defined by the user. As this optimization level is very aggressive (VB), the output parameters suffer a big reduction (VS). The fourth rule uses the neighbour difference parameter to determine the value of the light samples parameter. The rest of this type of rules are focused on solving the problem related to the interpolation among work units. The seventh rule is related to the second rule, and it adjusts the interpolation band size depending on the neighbour difference.

All the variables and all the rules have been described by using the XML metalanguage. This way, we can model QUEKARO easily by using the descriptive power of XML. In our next research development, we will only need to add some linguistic variables and some rules to use more rendering methods, as well as implementing the code which manages these parameters. Next, we have a piece of our description:

```
<?xml version='1.0' encoding='UTF-8'?>
<system name='QUEKARO>
 <linguisticvar type='in' name='Nd'>
  <fset a='37' b='37' c='45' d='53' label='VS'/>
  <fset a='45' b='53' c='53' d='60' label='S'/>
  ...
 </linguisticvar>
 ...
 <rule name='R1'>
  <antecedent name='C' label='B'/>
  <antecedent name='S' label='B'/>
  <antecedent name='Op' label='VB'/>
  <consequence name='Ls' label='VS'/>
  <consequence name='Rl' label='VS'/>
 </rule>
  ...
</system>
```

4 EXPERIMENTAL RESULTS

The results exposed here have been obtained by using QUEKARO with the MAgArRO system. All the implementations related to these systems are available for download under the GPL Free Software License (http://code.google.com/p/masyro06/). To test the system presented in this paper we have used 8 computers with the same hardware characteristics. Each of them has a Pentium Intel Centrino 1,6 GHz, 1 GB RAM, and the Debian GNU/Linux operating system. The rendering method used in all cases was the Pathtracing method, using the Yafray 0.0.9 render engine, an 8 oversampling level, a 5 recursion level in global configuration for Raytracing, an image resolution of 1000x600 pixels, and 1024 light samples. Making the rendering in one machine and using this configuration, a rendering time of 1 hour, 27 minutes, and 27 seconds (mm:ss format from now on) was required to obtain the 2D image shown in Figure 4.a.

The most important parameter in the optimization phase of this system is the optimization level parameter. Its value is defined by the user depending on

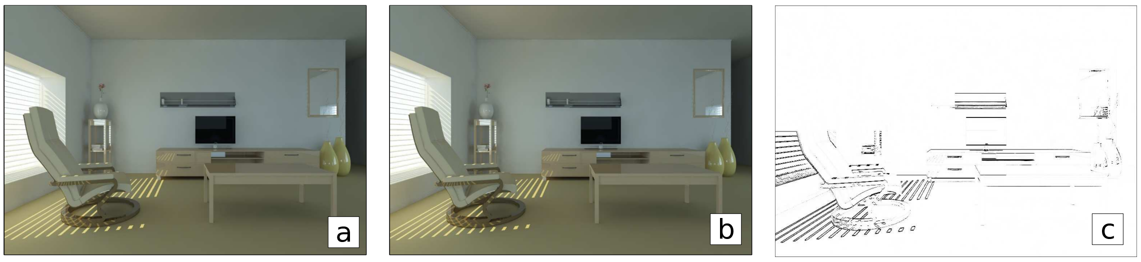

Figure 4: The result of the rendering without using optimizations (a) and using a small optimization level (b).

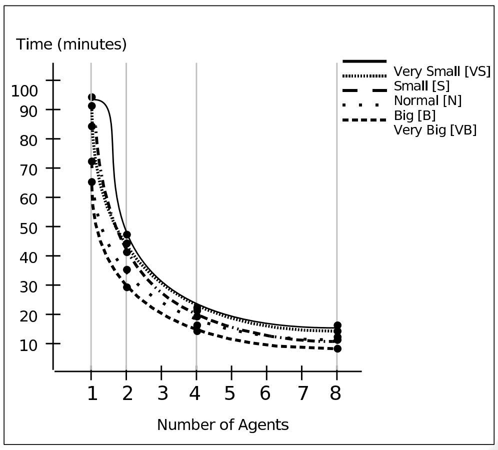

Figure 3: Rendering times by using different optimization levels.

the level chosen, ranging from VS (minimum value) to VB (maximum value). As well as managing this parameter in our tests, we will play by using a variable number of agents, trying to establish an optimal configuration (resources used and time needed). Table 1 shows the time required to render the scene by using different optimization levels (Figure 3). As we can see, the rendering time spent with very small and small optimizations in the case of using only an agent are lightly bigger than the time obtained without optimizations. These negative results are as a consequence of using interpolation bands and repeating the rendering of some part of the work units,as well as analyzing the input scene and composing the final result.

Table 1: Results obtained with different number of agents and different optimization levels.

Agents	VS	S	N	B	VB
1	93:46	90:54	86:54	70:29	67:02
2	48:15	45:29	41:15	35:53	29:20
4	22:08	21:47	19:31	16:24	14:21
8	15:58	15:31	12:17	10:50	09:47

However, this situation can be got around because our intention is to use this system with more than one agent. In fact, excellent results are obtained when four or more agents are used. For example, in the case of applying a small optimization level (Figure 4.a and Figure 4.b), the time required to render the scene is just about 22 minutes by using 4 agents, whereas the rendering time without using QUEKARO was about 87 minutes, that is, we have reduced the time to the fourth. In addition, the time decreases if we use more agents. However, the final rendering time of the scene always depends on the most complex work unit. Moreover, it is important to compare this overall approach to the paralell computing approach in relation to the rendering times required, because the first approach uses a partition scheme which allows us to obtain work units with a similar complexity. Whereas, the last approach is extremely subject to the more complex work unit.

As a final remark, using aggressive optimizations may result in a lose of quality. This problem can appear in several parts of the image (depending on the scene) and can be avoided by using lighter optimizations.

4.1 Comparing Empirical Results

As well as comparing the time obtained using different optimizations, it is also very important to compare the quality of the different results. However, this quality is a subjective topic because there are many factors which determine a final evaluation. For example, a non-expert person may think a 2D image resulting of a rendering has an excellent quality although this result has several differences in relation to a rendering without optimizations. Therefore, we could establish a metric in which the evaluation of an expert person is the reference when we are comparing empirical results. However, and to automate this evaluation, we have defined a metric based on the color difference between pixels, that is, a result using optimizations will be worse than another if this last has a lower difference than the first regarding the result without optimizations.

Implementing a program which uses this metric is quite easy by employing an API. Our implementation has the two 2D images to compare as input and a gray-scale image as output. This gray-scale image represents the difference between pixels of the input images. A pixel of black colour represents there is no difference between the pixels corresponding to the other two images, that is, the minimum difference, and a pixel of white colour represents there is the maximum difference. In the Figure 4.c we can see the difference of quality (depending on the proposed metric) between the results exposed in the Figure 4.a and the Figure 4.b. In this paper we have inverted the colours so that the lector can appreciate the differences between the two images clearly. In this case, they only differ 1.21 %.

5 CONCLUSIONS AND FUTURE WORKS

The obtaining of a system that allows the user to optimize the rendering process is an important step forward because rendering times are usually very high. In addition, the inclusion of QUEKARO into a multi-agent system makes it specially important to tackle the problems related to the rendering process. This overall system can work along the Internet to create a grid system which would be an efficient tool to deal with complex works.

The experimental results shown in the previous section prove the inclusion of fuzzy systems opens very promising research lines in the rendering problem. Moreover, the use of a fuzzy system has provided us with several advantages:

- It allows us to model expert knowledge easily.
- It provides us with a hightly scalable system.
- It provides us with an understandable human representation.

An important problem to be solved, and one of our main research line, is to use different rendering techniques in different work units. Under this approach, we could use complex realistic techniques only if needed, reducing the rendering time drastically. On the one hand, we may use the Pathtracing method in a complex work unit which needs a high-quality result because it contains an important part of the model, such as a glass of wine. On the other hand, we may use the Scanline method, which is a much faster method, in those parts that do not require a high-detail level, such as a wall. However, this idea implies we must adjust the interpolation band size efficiently to obtain a high quality in the final 2D image.

Another important research line is to use more rendering parameters, such as the spacial resolution of the image, the oversampling level, radiosity parameters (patch size, number of elements, etc), the number of photons, and so on. To include all these parameters demands that we model QUEKARO carefully to obtain high-quality results.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their insightful comments that have significantly improved the quality of this paper. This work has been funded by the Junta de Comunidades de Castilla-La Mancha under the Research Projects PBC06-0064 and PAC-060141.

REFERENCES

Chalmers, A., Davis, T., and Reinhard, E. (2002). *Practical Parallel Rendering*. AK Peters Ltd.

Fernandez-Sorribes, J. A., Gonzalez-Morcillo, C., and Jimenez-Linares, L. (2006). Grid architecture for distributed rendering. In *Proceedings of Ibero-American Symposium in Computer Graphics 2006*, pages 141–148.

Foundation for Intelligent Physical Agents FIPA (2002). *FIPA Abstract Architecture Specification*. Foundation for Intelligent Physical Agents FIPA, Geneva, Switzerland.

Kajiya, J. T. (1986). The rendering equation. *Computer Graphics*, 20(4):143–150. Proc. of SIGGRAPH'86.

Kuoppa, R. R., Cruz, C. A., and Mould, D. (2003). Distributed 3d rendering system in a multi-agent platform. In *Proceedings of the Fourth Mexican International Conference on Computer Science (ENC'03)*, pages 168–176.

Schlechtweg, S., Germer, T., and Strothotte, T. (2005). Renderbots – multiagent systems for direct image generation. In *Computer Graphics Forum*, pages 137–148.

Sundstedt, V., Debattista, K., Longhurst, P., Chalmers, A., and Troscianko, T. (2005). Visual attention for efficient high-fidelity graphics. In *Spring Conference on Computer Graphics (SCCG 2005)*, pages 162–168.

Tanaka, K. (1998). *An introduction to fuzzy logic for practical applications*. Springer.

Zadeh, L. A. (1975). The concept of a linguistic variable and its applications to approximate reasoning. Part i, ii, iii. *Information Science*.