CEIG - Spanish Computer Graphics Conference (2012)
Isabel Navazo and Gustavo Patow (Editors)

Differential Ray Marching

Adolfo Muñoz[1]

[1]Universidad de Zaragoza

Abstract

Several participating media rendering algorithms are based on ray marching: they integrate the variations of radiance along the volume covered by the participating media by splitting the path of light into segments and sampling light contribution at each of those segments. This paper revisits the concept of ray marching not as an integration technique, but as the application of a numerical method to solve an initial value differential equation. We present how to apply different numerical methods as ray marching techniques, analyze a wide range of them and study their applicability under different scenarios. Furthermore, we show how each of them improves over traditional ray marching. Any participating media rendering algorithm that is based on ray marching will benefit from the application of our technique by reducing the number of needed samples (and therefore, rendering time) and/or increasing accuracy.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Ray tracing

1. Introduction

One of the most daunting tasks in Computer Graphics is to accurately render participating media. As such, is a very active field of research, in which each of the different techniques finds its own compromise between accuracy and simulation time. A ray of light that traverses a participating medium is altered in several ways at differential level: it may be scattered or absorbed, or its power may increase due to medium emission or in-scattering. All these interactions happen at every differential point of the path of light. As a consequence, light simulation becomes quite hard. How to simulate a phenomena that changes at every differential level?

One of the most widely used techniques to render participating media is ray marching [PH89, Jen01]. Its main idea is to divide the path of light into uniform segments, and approximate all the differential interactions that happen on that segment by a single sample. That is mathematically represented as approximating an integral by a summatory. Depending on the size of each of those segments, the render becomes more accurate (short segments) or the simulation takes shorter time (large segments). The optimal compromise between time and accuracy happens at a specific step size, which is a per-scene parameter.

Ray marching is, therefore, a fundamental technique for participating media simulation. Our work attempts to pick up its key ideas but redefine the underlying mathematical formulation in order to eliminate several of its weaknesses. Instead of approximating the corresponding integral, we solve the Radiative Transfer Equation [Cha60] in its differential form, which actually is an initial value problem, for which there are plenty of numerical methods that can solve it [Gea71, Pre07].

This resolution technique enables a new broad set of ray marching techniques, one for each specific method. We review the literature about numerical methods and study their specific applicability to participating media simulation. Different methods lead to different sampling techniques along the path of light which are not related to per-medium heuristics (which may be partial or ad-hoc) but to the mathematical definition of differential light contribution along the path.

Traditional ray marching actually becomes a particular case of our algorithm, and, as a consequence, any technique that includes traditional ray marching as part of their corresponding render engine may trivially include ours.

DOI: 10.2312/LocalChapterEvents/CEIG/CEIG12/091-098

2. Previous work

Participating media rendering: Many previous papers have dealt with participating media rendering, either based on ray tracing [KvH84], Montecarlo simulations [PM93] or other methods [RT87]. In practice, all of them approximate the Radiative Transfer Equation [Cha60], and some of them are explicitly based on solving that equation [KYNN91], just as our paper is. There are several compendiums that can give a broader overview of the available rendering techniques [GJJD09].

Ray marching: Ray marching has traditionally become one of the key techniques for rendering participating media [PH89]. Since then, it has been adapted to algorithms such as photon mapping for participating media [Jen01] or applied to specific participating media, such as smoke [FSJ01],

Some authors have proposed adaptive ray marching techniques, that adjust their step size to better fit the scene's properties. They can be based on medium heuristics [Jen01] or to the distribution of light samples in the volume [JNSJ11]. On contrast, our algorithm provides a generic framework that, using an adaptive numerical method, can potentially adapt to both medium properties and light distribution without the need of ad-hoc heuristics.

Ray marching has been used in interactive and real-time techniques [ZRL*08, WR08]. We also expect that our algorithm can be easily integrated into a GPU renderer.

Numerical methods for initial value problems: There has been a lot of research regarding the different numerical methods that can be used with our technique [Gea71, CL85, Pre07], and those that we use are discussed in Section 4. In Computer Graphics, such numerical methods have not been widely used for rendering, with few exceptions [GSMA06].

3. Overview

When light traverses a participating medium, it interacts with the medium at every differential step, in three possible ways: it may get absorbed, scattered or even emitted by the medium. The equation that defines this behavior is the Radiative Transfer Equation [Cha60]:

$$\frac{\partial L(\mathbf{x},\omega)}{\partial t} = \sigma_a(\mathbf{x})L_e(\mathbf{x},\omega) - \sigma_t(\mathbf{x})L(\mathbf{x},\omega) + \sigma_s(\mathbf{x})\int_\Omega p(\mathbf{x},\omega',\omega)L(\mathbf{x},\omega')\,d\omega' \quad (1)$$

where $\frac{\partial L(\omega_o)}{\partial t}$ represents the differential variation of radiance along the path of light t, $\mathbf{x}$ is the differential point at which the interaction occurs, ω represents the direction followed by light and ω' represents the direction of other light paths that reach the differential point $\mathbf{x}$. The rest of the symbols represent the properties of the medium:

- σ_a is the absorption coefficient, energy that is absorbed by the medium at every differential step.
- σ_s is the scattering coefficient, energy scattered by particles in the medium at every differential step.
- $\sigma_t = \sigma_a + \sigma_s$ is the extinction coefficient, energy that is either absorbed or out-scattered.
- $p(\mathbf{x},\omega',\omega)$ is the phase function, that defines the angular distribution of light scattering.
- L_e is the medium's emission.

In order to render a participating medium, we need to solve Equation 1 and obtain the radiance $L(\mathbf{x},\omega)$ that reaches the eye. Traditional ray marching solves that equation by approximating its integral form:

$$L(\mathbf{x_t},\omega) = T_r(\mathbf{x_0},\mathbf{x_t})L(\mathbf{x_0},\omega) + \int_0^t T_r(\mathbf{x_0},\mathbf{x_s})\sigma_a(\mathbf{x_s})L_e(\mathbf{x_s},\omega)ds + \int_0^t T_r(\mathbf{x_0},\mathbf{x_s})\sigma_s(\mathbf{x_s})L_i(\mathbf{x_s},\omega)ds \quad (2)$$

where $T_r(\mathbf{x_0},\mathbf{x_t})$ is named *transmittance* and accounts for all the light that has traversed the medium between $\mathbf{x_0}$ and $\mathbf{x_t}$ without getting extinguished due to the medium's properties. $L_i(\mathbf{x_s},\omega)$ represents the in-scattered radiance (energy coming from different light paths). They are defined as follows:

$$T_r(\mathbf{x_0},\mathbf{x_t}) = e^{\int_0^t -\sigma_t(x_s)ds} \quad (3)$$

$$L_i(\mathbf{x_s},\omega) = \int_\Omega p(\mathbf{x_s},\omega',\omega)L(\mathbf{x_s},\omega')\,d\omega' \quad (4)$$

Traditional ray marching techniques [PH89, Jen01] solve this equation by approximating each of the integrals by using the rectangle method [Pre07]. For instance, the third addend of Equation 2 could be approximated as follows:

$$\int_0^t T_r(\mathbf{x_0},\mathbf{x_s})\sigma_s(\mathbf{x_s})L_i(\mathbf{x_s},\omega)ds = \Sigma_{k=0}^{n} T_r(\mathbf{x_0},\mathbf{x_k})\sigma_s(\mathbf{x_k})L_i(\mathbf{x_k},\omega)\Delta s \quad (5)$$

in which the distance t traveled by light is split into a set of n segments (also called *steps*) of size Δs. Each of those segments is a sample of the in-scattering L_i along the path of light. By increasing the number of segments n, image quality gets improved (step size Δs is reduced) but simulation time also increases.

Figure 1 shows a diagram explaining where are those samples located. Notice, however, that each of those samples involves the evaluation of T_r and L_i, which are again integrals, which must be also approximated.

T_r (see Equation 3) is usually evaluated by using the same integration rule as the main integral. Redundant calculations can be cached and performed more efficiently, but still this double integral can result into a performance hit. L_i involves single scattering (light coming from the light sources) which can be easily accounted for, and multiple scattering (light that has bounced several times in the medium) which can be sampled for instance using Montecarlo techniques or others such as photon mapping [Jen01].

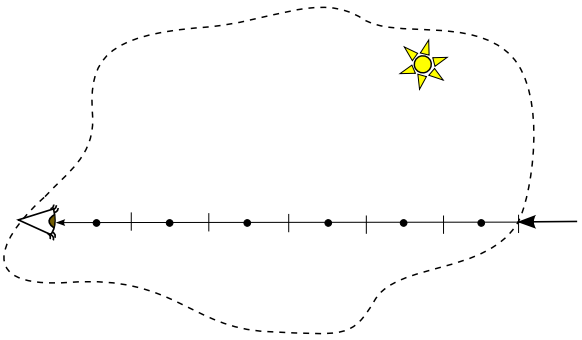

Figure 1: *Ray marching: the path of light inside the medium is split into a set of segments. At each of those segments, the variation of radiance is evaluated using a single sample.*

Our work, on the other hand, attempts to take advantage of the differential form of the radiative transfer equation (Equation 1) which, as it is shown below in the text, presents several advantages over the integral form. First, there is a set of numerical methods with interesting properties (beyond the quadrature rectangle method) that can be applied. Also, solving the equation on its differential form avoids computing the double integral (the inner T_r).

Equation 1 has the following form:

$$y'(t) = f(y,t) \tag{6}$$

considering that t is the distance along the path of light and y the radiance itself. Furthermore, the radiance that enters the medium $y_0 = L(\mathbf{x_0}, \omega)$ is also known (can be calculated from the surface's material properties and the light distribution). Therefore, Equation 6 and the y_0 value form an initial value problem [Gea71, CL85, Pre07] and, as such, there are several kinds of different numerical methods that solve it. In the end, as shown below, this results into smarter ways of choosing the samples along the path of the light that interacts with the medium, which in turn leads to better image quality and / or shorter simulation times.

4. Numerical methods

Considering that Equation 1 has the form of an initial value problem (Equation 6, being the radiance entering the medium its initial value) it can be solved by a number of numerical methods that are tailored to solve such specific differential equations. The key idea is to deduce values from the function based on the previously deduced ones (up to the initial one). All the numerical methods presented in this section have been widely documented in existing literature [Gea71, BF10], but we include brief explanations of the methods tested on this paper for completeness purposes.

The most basic method is Euler's method. Given the initial known value y_0 at the initial position t_0, we can estimate different values of the function as follows:

$$y_{i+1} = y_i + sf(y_i, t_i) \tag{7}$$

$$t_{i+1} = t_i + s \tag{8}$$

where s is the *step size*, a parameter of the method. The method finishes when it reaches the desired final value position t_n. The larger the step size, the faster calculation times (but the lower accuracy). This is similar to box integration.

Notice that the samples are uniformly distributed along t. This means that, when applied to Equation 1, Euler's method is numerically equivalent to traditional ray marching (see Figure 1). However, as with ray marching, Euler's method presents several shortcomings that can be solved by choosing more advanced methods.

4.1. Explicit Runge-Kutta

Euler's method estimates the value at each step from its derivative at a single value. A better approximation can be done if using more than one sample per step. This is the key idea of Runge-Kutta methods. An order two Runge-Kutta method would need two samples on function f and would replace Euler's Equation 7 by:

$$y_{i+1} = y_i + sf\left(y_i + \frac{1}{2}sf(y_i, t_i), t_i + \frac{1}{2}s\right) \tag{9}$$

Notice that there is a whole family of order two Runge-Kutta methods. In literature, the particular method defined by Equation 9 is often named *midpoint method* but for this work we will name it either *order two Runge-Kutta* or, for simplicity *Runge-Kutta 2*.

One of the possible fourth order Runge-Kutta methods (the one used in this paper) requires four samples and it is defined by the following equation:

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + k_4) + \frac{1}{3}(k_2 + k_3) \tag{10}$$

where

$$k_1 = sf(y_i, t_i)$$

$$k_2 = sf\left(y_i + \frac{1}{2}k_1, t_i + \frac{1}{2}s\right)$$

$$k_3 = sf\left(y_i + \frac{1}{2}k_2, t_i + \frac{1}{2}s\right)$$

$$k_4 = sf(y_i + k_3, t_i + s)$$

As it can be deduced, both Runge-Kutta methods used in this paper present a trade-off between per-step time and per-step accuracy. Actually, Euler is the order one Runge-Kutta method.

4.2. Implicit methods

Explicit Runge-Kutta methods (presented above) are prone to instabilities caused by the equation, leading to high errors

on regions in which the equation is specially stiff. Implicit methods deal with this issue: instead of blindly following the function step by step, they solve a per-step system of equations that numerically stabilizes the calculations.

The simplest implicit method (the one we use in this paper) is the Backwards Euler method, which replaces Equation 7 with:

$$y_{i+1} = y_i + sf(y_{i+1}, t_i) \tag{11}$$

which means that, for finding y_{i+1} you actually need to know y_{i+1}. In practice, what this method needs to do at every single step is to solve Equation 11 considering y_{i+1} as the unknown. We do that by using the fixed-point method [BF10], which calculates several iterations until the error is below a given tolerance threshold. Therefore, this method is configured through two parameters: step size and fixed-point's tolerance.

As shown below, implicit methods overcome mathematical singularities such as the ones caused by point lights with little performance hit.

4.3. Predictor-corrector techniques

Implicit methods tend to be slower (but more accurate at stiff equations) than their explicit counterparts. Also, higher order methods are slower (but more accurate) than lower order methods. Predictor-corrector techniques are based on the idea that some steps can be calculated with faster methods and some others require more accuracy and require slower methods.

In this work we test the Trapezoidal (actually, Euler-Trapezoidal) method. It is defined by the following (order two) implicit method equation:

$$y_{i+1} = y_i + \frac{s}{2}\left(f(y_i, t_i) + f(y_{i+1}, t+s)\right) \tag{12}$$

The difference with a pure implicit method in this case is that the first iteration of the fixed-point solver is done with Euler's method, which is faster and, in some cases, enough for an adequate step estimation.

4.4. Adaptive step

The last kind of tested numerical methods are adaptive ones. Their key idea is that larger steps are less accurate, but faster than shorter steps. They estimate the error at every step and increase the step size if the error is very low, or reduce the step size if the error is too high (compared to a specific tolerance parameter).

As the function itself cannot be evaluated (it is not known, it is what we want to solve) the error must be estimated by comparing the solution of two different numerical methods of different orders. In our case, the method we use compares the relative error between Euler (Equation 7) and Runge-Kutta 2 (Equation 9) for adjusting the step size.

5. Error-time analysis

Each of the numerical methods that have been introduced on previous section shows different behavior when applied to solve Equation 1, depending on their parameters:

- Step size
- Tolerance (in implicit, predictor-corrector or adaptive methods).

For the error analysis, we need to find a ground truth to compare our results to. In practice, this means finding an analytical solution to Equation 1 which we do not have (if we did, we would not need ray marching in the first place). However, under several specific conditions (enumerated below) it is possible to find this analytical solution.

These conditions are:

- Homogeneous medium (constant absorption and scattering properties). No medium emission.
- Uniform lighting, without shadows, which in practice can be achieved by an infinite directional light and a scene geometry that does not cast any shadows.
- No multiple scattering.

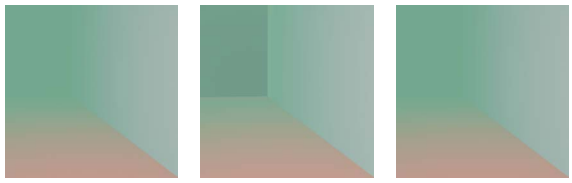

Figure 2: *Test scene. Left: ground truth render. Middle: perceivably wrong render due to inadequate numerical method and method parameters. Right: accurate render due to adequate numerical method.*

Figure 2, left, shows an scene that fulfills those conditions. The scene is very basic: three axis-aligned planes with a single directional light source, no shadows and a participating medium that resembles some kind of green fog. The goal of this scene is not its beauty but the possibility of render it analytically.

Under the conditions presented above, several factors can be taken out of the integral calculus (they become constant) and as a consequence Equation 1 has the following analytical solution (which can be deduced from Equation 2):

$$L(\mathbf{x_t}, \omega) = e^{-t\sigma_t} L(\mathbf{x_0}, \omega) + p(\omega_\mathbf{i}, \omega) L(\omega_\mathbf{i}) \frac{1 - e^{-t\sigma_t}}{\sigma_t} \tag{13}$$

where ω_i represents light direction, and $p(\omega_\mathbf{i}, \omega)$ and $L(\omega_\mathbf{i})$ are now constant for the whole light path (single directional light source).

Equation 13 serves for the purpose of a ground truth render (see Figure 2, left) and helps to check whether a method shows high (see Figure 2, middle) or low errors (see Figure 2, right).

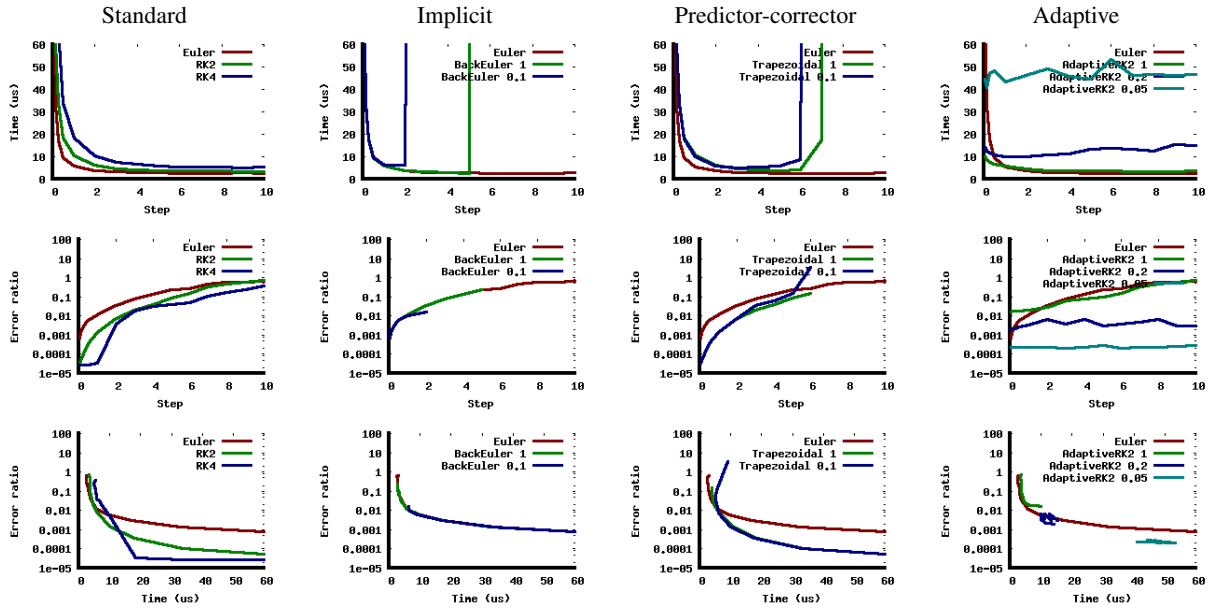

Figure 3: *Error analysis, regarding different kinds of numerical methods. Error is calculated as relative to the ground truth (so a 1 error would mean a 100% deviation over ground truth) and it is shown on logarithmic scale for better visualization purposes. Time is measured in µs per ray (so a million rays would need that specific number of seconds). Three variables are compared: time, error and number of steps of the different methods. Euler method (which is equivalent to traditional ray marching) is included on every graph for direct comparison purposes. RK is the abbreviation for Runge-Kutta. The number besides some methods (Trapezoid, BackEuler, AdaptiveRK2) is the chosen tolerance parameter. First row: time vs. step. Second row: error vs. step. Third row: error vs. time.*

Figure 3 shows the results of the performed error analysis for a set of numerical methods. For standard methods, specifically Euler, order two Runge-Kutta (RK2) and order four Runge-Kutta (RK4) the behavior regarding step size is the expected: the smaller the step (more steps required, therefore), the more rendering time and the more accuracy. However, the error-time analysis shows that each of the methods finds its own compromise between accuracy and simulation time. Euler is the most efficient for low accuracies, while higher-order methods converge faster with step size and therefore are more efficient at giving accurate renders.

Regarding implicit methods, there is no perceivable improvement on using the Backward Euler method as opposed to standard Euler. In contrast, depending on the tolerance, this fixed-point method may fail to converge to an adequate solution for bigger step sizes (based on a too high per-step error). Euler Trapezoidal method (which is a predictor-corrector method): its performance is quite similar than the order two Runge-Kutta method and in some cases may not converge. In this case, this simple scene setup leads to a non-stiff Equation 1, therefore neglecting the advantage of these methods. Other scene setups (see Section 6) include singularities on the equation that explicit methods cannot handle but are easily overcome by implicit and predictor-corrector methods.

Last, adaptive methods show the expected performance. For lower tolerances, the effect of step size is ignored, as the method adapts its step size to the desired overall error (leading again to better accuracy than Euler). This is a very desirable property, as a tolerance parameter is more user-friendly than step size.

6. Results

We have also tested our algorithm in several scene setups, using different numerical methods. First, we have tested the convergence of the standard algorithms in a scene that consists on two car's headlights (point lights inside a conical geometry that represents the headlight's reflector) behind two Stanford bunnies (Figure 4). The simulation of this scene can be quite tricky because of the discontinuities on the derivative of radiance along the path of light, provoked by the sharp shadows cast by both the bunnies and the reflectors.

The scene has been rendered using Euler, order two Runge-Kutta and order four Runge-Kutta methods varying the step size. For large step sizes, the error is quite noticeable in the three numerical methods: the discontinuities due to the

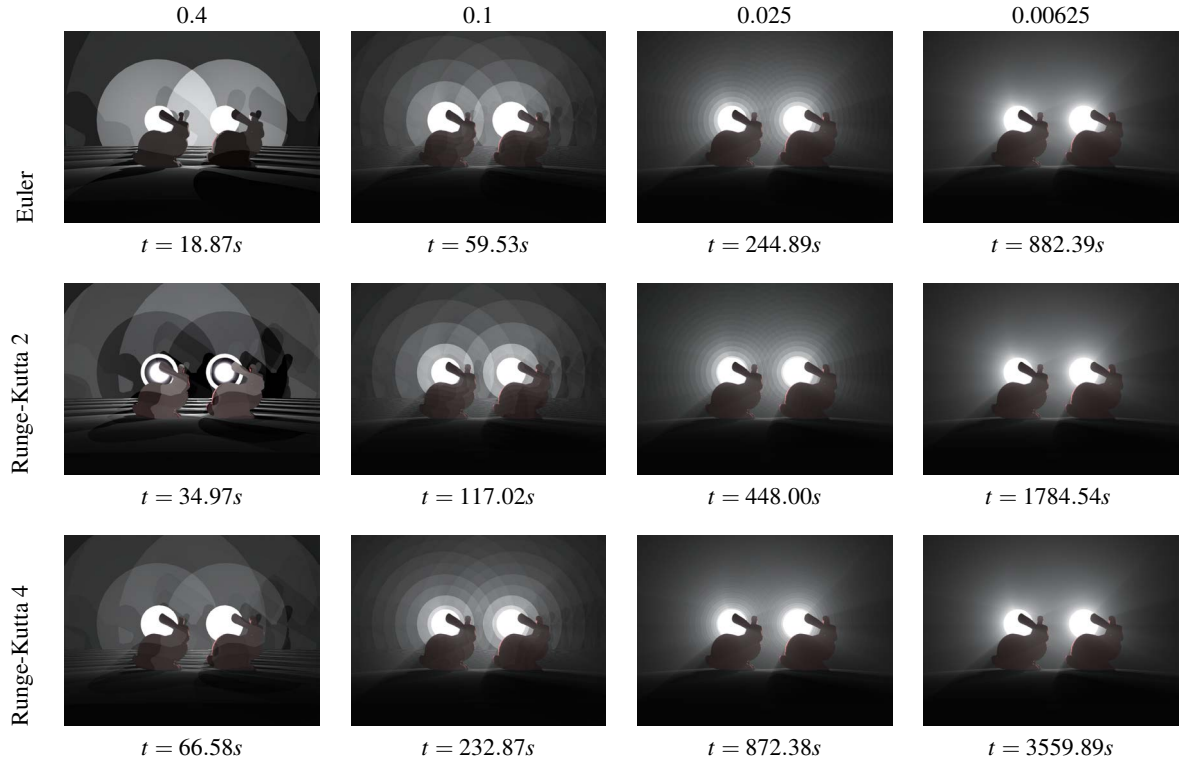

Figure 4: *Convergence of the standard numerical methods according to step size. Each row shows the results for a numerical method. From left to right, step size is reduced. Each cell contains the rendered image using the specific numerical method with the column's step size, plus the image's rendering time measured in seconds.*

shadow boundaries at each step are quite noticeable. However, as the step size shrinks (increasing rendering time), the frequency of these discontinuities increase until they vanish for the shortest step sizes.

Comparing the three methods, there are two relevant facts to be noted: First, rendering time is proportional to the number of samples (double for Runge-Kutta 2, quadruple for Runge-Kutta 4). Last, while the discontinuities are noticeable on the three methods, higher order methods turn them to be less apparent than lower order methods.

Multiple scattering has been approximated by an ambient term for two purposes: first, the errors made on multiple scattering simulation might be impossible to disambiguate from our algorithm's error. Furthermore, multiple scattering tends to *blur* the apparent radiance, therefore neglecting these discontinuities that are pretty relevant for the performance analysis of ray marching techniques. The medium is homogeneous, although this could be trivially extended to heterogeneous media.

Figure 5 presents an analysis on low/high quality results for different numerical methods and how are they related to the corresponding method parameters. The top row shows low quality results with five different methods, while the bottom row presents the high quality results (for the same scene setup) for the same methods, with their corresponding parameters and render time. The scene is a foggy Cornell Box with a point light in the center. Again, this point light becomes an interesting challenge, as the exact position of the light source is a singularity point on Equation 1.

The two order one methods (Euler and its implicit version) show a similar behavior: for similar render times, their low quality version shows clear artifacts for both of them (at the spheres for Euler, and at the background for Backward Euler). On contrast, order two methods present very different behaviors: Runge-Kutta is unable to overcome the effect of the singularity (the point light), while both the Trapezoid and adaptive methods overcome it. On its low quality setup, the adaptive method is faster but less accurate, because its tolerance parameter is far from restrictive, and most probably the step size ends up growing very large. The Trapezoid method is similar than Runge-Kutta 2 in terms of time and quality, except that, being predictor-corrector, is able to overcome the black hole singularity.

The high quality versions are very similar in terms of render result. Order one methods take similar render times. Or-

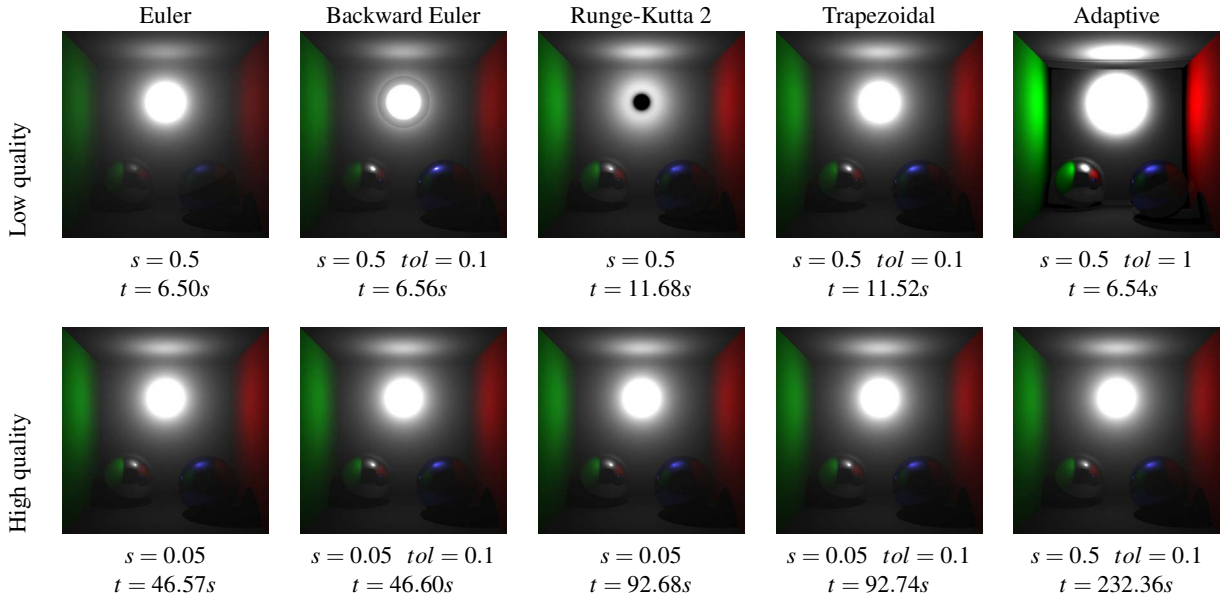

Figure 5: *Cornell box rendered using different combination of parameters of different numerical methods, one per column (s = step size, tol = method tolerance, t = render time in seconds).*

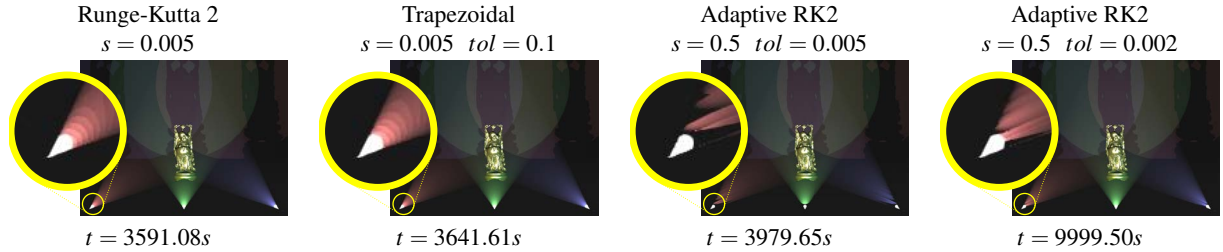

Figure 6: *Scene with high frequency illumination. Runge-Kutta 2 and Trapezoidal methods present similar behavior, while the particular setup of this scene hampers the adaptive method's performance. The insets zoom into the most affected image area.*

der two methods are also similar in render time, except for the adaptive method, because its tolerance parameter is probably very restrictive in this case, leading to very short step sizes.

Our last test scene (Figure 6) is designed to have complex (high-frequency) illumination and to test all the order two methods. Runge-Kutta 2 and Trapezoidal methods give accurate renders using similar CPU times. However, the adaptive method gives artifacts for the same render time, and can still be perceived even for a more restrictive tolerance (and almost triple render time). This is owed to the peculiar design of the scene: the step size under the adaptive framework increases on low error regions (basically, the dark regions). When the method reaches the high-frequency light beam, it totally skips it (due to a large step size). Notice that the rest of the scene is quite accurate. A possible solution for this would be to devise a new error adaptation strategy that include a maximum step size, or use a higher order adaptive method.

All these simulations (both this and previous sections') have been calculated in an Intel Core i5 at 2.4GHz with 4GB RAM. The image resolution is 800x600 (except for the Cornell box, which is 800x800). The code has been parallelized using threads.

7. Conclusions and future work

We have redesigned the ray marching algorithm based on the differential form of the radiance transfer equation, and have studied how do different numerical methods translate into sampling strategies along the path of light, finding their own compromise between render time and accuracy. We have shown that traditional ray marching is a particular case of our algorithm for Euler's method, and shown that other tech-

niques present interesting properties, such as adaptive step size, robustness to stiffness or faster convergence.

We have also shown a numerical validation of our technique, a convergence study and a set of rendered results that show using different numerical methods. We have also compared with traditional ray marching (Euler's method) and overcome some of its limitations, and found more efficient and more accurate solutions. Implicit and predictor-corrector methods overcome scene singularities with no noticeable performance hit, while adaptive methods are easier to configure.

We have considered Equation 1 to be generic, but in fact it presents several properties that might be worth studying in order to avoid redundant calculations, find better ways to solve the corresponding system that is required for implicit numerical methods or use specific (tailored to the properties of the equation) numerical methods. Furthermore, other methods or other error estimation strategies could be tested. For instance, higher order adaptive methods may lead to more accurate results or better error estimations (and therefore less steps).

For a first analysis of the results of the differential ray marching algorithm, multiple scattering has been approximated by an ambient term in order to avoid the influence of the simulation multiple scattering on the overall error. However, it would be interesting to check how Montecarlo multiple ray marching, photon mapping or beam tracing could be included as multiple-scattering estimators in our differential ray marching approach. We would also like to test our technique on non-homogeneous media. Last, it is expected that, as traditional ray marching, our technique can be easily ported to GPU.

We hope that our work is included into renderers that are based on traditional ray marching (should be a trivial task) and that it inspires further research on the topic.

8. Acknowledgements

This research has been funded by the European Commission, Seventh Framework Programme, through the projects GOLEM (Marie Curie IAPP, grant agreement no.: 251415) and VERVE (Information and Communication Technologies, grant agreement no.: 288914), by the Spanish Ministry of Science and Technology (TIN2010-21543) and by the Gobierno de Aragón (project CTPP6/11).

References

[BF10] BURDEN R., FAIRES J.: *Numerical Analysis*. Brooks/Cole, Cengage Learning, 2010. 3, 4

[Cha60] CHANDRASEKHAR S.: *Radiative Transfer*. Dover Publications, Inc., 1960. 1, 2

[CL85] CODDINGTON E. A., LEVINSON N.: *Theory of Ordinary Differential Equations*, 9 ed. Tata Mcgraw-Hill, 1985. 2, 3

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 15–22. URL: http://doi.acm.org/10.1145/383259.383260, doi:10.1145/383259.383260. 2

[Gea71] GEAR C.: *Numerical initial value problems in ordinary differential equations*. Prentice-Hall series in automatic computation. Prentice-Hall, 1971. 1, 2, 3

[GJJD09] GUTIERREZ D., JENSEN H. W., JAROSZ W., DONNER C.: Scattering. In *ACM SIGGRAPH ASIA 2009 Courses* (New York, NY, USA, 2009), SIGGRAPH ASIA '09, ACM, pp. 15:1–15:620. URL: http://doi.acm.org/10.1145/1665817.1665832, doi:http://doi.acm.org/10.1145/1665817.1665832. 2

[GSMA06] GUTIERREZ D., SERON F. J., MUNOZ A., ANSON O.: Technical section: Simulation of atmospheric phenomena. *Comput. Graph. 30*, 6 (Dec. 2006), 994–1010. URL: http://dx.doi.org/10.1016/j.cag.2006.05.002, doi:10.1016/j.cag.2006.05.002. 2

[Jen01] JENSEN H. W.: *Realistic image synthesis using photon mapping*. A.K. Peters, Natick, Massachussets, 2001. 1, 2

[JNSJ11] JAROSZ W., NOWROUZEZAHRAI D., SADEGHI I., JENSEN H. W.: A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Transactions on Graphics (Presented at ACM SIGGRAPH 2011) 30*, 1 (Jan. 2011), 5:1–5:19. 2

[KvH84] KAJIYA J. T., VON HERTZEN B. P.: Ray tracing volume densities. *Computer Graphics 18*, 3 (1984), 165–174. 2

[KYNN91] KANEDA K., YUAN G., NAKAMAE E., NISHITA T.: Realistic visual simulation of water surfaces taking into account radiative transfer. In *Proc. of CAD/Graphics 91* (1991), pp. 25–30. 2

[PH89] PERLIN K., HOFFERT E. M.: Hypertexture. *SIGGRAPH Comput. Graph. 23*, 3 (July 1989), 253–262. URL: http://doi.acm.org/10.1145/74334.74359, doi:10.1145/74334.74359. 1, 2

[PM93] PATTANAIK S. N., MUDUR S. P.: Computation of global illumination in a participating medium by monte carlo simulation. *The Journal of Visualization and Computer Animation 4*, 3 (1993), 133–152. URL: http://dx.doi.org/10.1002/vis.4340040303, doi:10.1002/vis.4340040303. 2

[Pre07] PRESS W.: *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 2007. 1, 2, 3

[RT87] RUSHMEIER H. E., TORRANCE K. E.: The zonal method for calculating light intensities in the presence of a participating medium. *j-COMP-GRAPHICS 21*, 4 (July 1987), 293–302. 2

[WR08] WYMAN C., RAMSEY S.: Interactive volumetric shadows in participating media with single-scattering. *2008 IEEE Symposium on Interactive Ray Tracing* (2008), 87–92. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4634627. 2

[ZRL*08] ZHOU K., REN Z., LIN S., BAO H., GUO B., SHUM H.-Y.: Real-time smoke rendering using compensated ray marching. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 36:1–36:12. URL: http://doi.acm.org/10.1145/1360612.1360635, doi:10.1145/1360612.1360635. 2