Visual Comput (2007) 23: 493–502
DOI 10.1007/s00371-007-0122-y

ORIGINAL ARTICLE

Adolfo Muñoz
Diego Gutierrez
Francisco J. Serón

Optimization techniques for curved path computing

Published online: 24 May 2007

A. Muñoz · D. Gutierrez · F.J. Serón (✉)
Grupo de Informatica Grafica Avanzada (GIGA), University of Zaragoza, Spain
{adolfo, diegog, seron}@unizar.es

Abstract Participating media with an inhomogeneous index of refraction make light follow curved paths. Simulating this in a global illumination environment has usually been neglected due to the complexity of the calculations involved, sacrificing accurate physical simulations for efficient visual results.
This paper aims to simulate non-linear media in a more reasonable time than previous works without losing physical correctness. Accuracy is achieved by solving the Eikonal equation of geometrical optics, which describes the path followed by a light beam that traverses a non-linear medium. This equation is used in the context of a photon mapping extension.
To improve the efficiency of the method, we study the existing correlation between numerical methods and the description of the non-linear medium, in terms of simulation time and error. Also, by taking advantage of several features of the scenes that include non-linear media, new optimization techniques that can be applied both for ray tracing and photon mapping will be developed. Flight or driving simulators could greatly benefit from this work.

Keywords Non-linear media · Atmospheric phenomena · Optimization

1 Introduction

One of the most common simplifications in rendering algorithms is to consider that light travels in straight paths, a fact that is only true in a vacuum or in medium with a homogeneous index of refraction. This simplification works well for most cases, although it fails to explain several natural phenomena, like mirages, visual distortions caused by fire or sunset effects. In fact, the index of refraction of most media (like the atmosphere) is not homogeneous, and therefore light travels a curved path. In most cases the effects of this curved path are negligible due to the short distances involved, but under the right circumstances the effects are clearly visible and quite spectacular.

Accurate simulation of non-linear media (with inhomogeneous index of refraction) requires several minutes of rendering time (see Sect. 6 for more accurate times). This paper will focus on several optimization strategies for physically accurate simulation of non-linear media, to reduce rendering times without losing physical correctness in the results. By using very simple assumptions in conjunction with the appropriate numerical method, we can achieve speedups of up to five. While this is still far from being interactive, we believe it is a step towards that goal.

1.1 State of the art

There are not many previous works that take into account inhomogeneous media for rendering. Some of them are specific for atmospheric phenomena, in which the index of refraction varies at different heights. Berger et al. [1] and Lintu et al. [11] divide the atmosphere into several layers

(perpendicular to height) and compute paths for each of them. Musgrave [13] proposes a model based just on total reflection. These models are limited to certain scenes and lose physical accuracy.

Gröller [5] proposes a more general approach for non-linear ray tracing, although the work focuses on mathematical and physical systems instead of on the effect of the index of refraction on the path of light. Stam and Languenou [14] solve this problem by using geometrical optics but the distribution of the index of refraction is still limited.

Gutierrez et al. [8] present a model that takes into account the Eikonal equation, including a model of the Earth's atmosphere by describing its varying index of refraction. Muñoz et al. [12] build on this algorithm and presents optimization techniques that reduce computation times without reducing physical accuracy. This paper is an extension of that work, which includes an analysis of different numerical methods plus the adaptation of the proposed optimization techniques into a global illumination context.

This paper is organized as follows. Section 2 describes the physical basis that gives the curved path of light and introduces the algorithm which will be optimized; Sect. 3 introduces several considerations about optimization techniques; Sects. 4 and 5 present the optimization techniques for ray tracing and for particle tracing. Section 6 shows the time results of the tests realized so far, considering several numerical methods and different test scenes.

2 Theoretical background

The physical basis that describes how the light behaves in a non-linear medium, as well as the algorithm that simulates it, are discussed here. Extensive information on this topic can be found in [6].

When the medium traversed by light is inhomogeneous, with the index of refraction varying continuously from point to point, the trajectory of light is affected at each differential step. The final result of this differential change is a curved path that distorts the normal view of a real scene, but that path cannot be efficiently calculated by using traditional ray tracing. Fermat's principle [4] can be applied to obtain the curved trajectory of the light rays. Performing the appropriate deduction [7, 9] Fermat's principle gives the following equation:

$$\frac{\mathrm{d}}{\mathrm{d}l}\left(n_\lambda \frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}l}\right) - \nabla n_\lambda = 0 \longleftrightarrow \frac{\mathrm{d}}{\mathrm{d}l}\left(n_\lambda \frac{\mathrm{d}x_j}{\mathrm{d}l}\right) - \frac{\partial n_\lambda}{\partial x_j} = 0 \quad (1)$$

where l is the length of the arc, n is the index of refraction at each differential point of the medium and $(j = 1, 2, 3)$ are the three coordinates of a point.

The Earth's atmosphere is itself an inhomogeneous media, and thus light travels following a curved path. Under standard circumstances,the effects are negligible. It is under non-standard circumstances when we really see some amazing atmospheric phenomena, like mirages, the green flash or some distortions in the solar disc at sunrise and sunset.

2.1 The simulation algorithm

There are three key components in our approach for a correct simulation of the atmospheric phenomena:

1. An accurate model of the atmosphere itself gives an exact index of refraction at every differential point.
2. The Eikonal equation, based on Fermat's principle gives the exact trajectory of the light as it traverses an inhomogeneous medium, like the atmosphere.
3. A numerical method solves the equation.

An accurate model of the atmosphere can be found in [8]. This model is based on the USA 1976 Standard Atmosphere [16], and includes a user-driven atmospheric profile manager, that enables the inclusion of inversion layers (zones in which temperature does not follow the standards) for de-standardization. A remarkable feature of this model is that the resulting index of refraction is a function of just height (one dimension), a fact that enables us to do several optimizations of the algorithm, as we will see in the coming sections.

Equation 1 can be used as an initial value problem [2] with initial value the origin point and direction of the corresponding ray. The equation can then be solved numerically by any numerical method. The results of the numerical method are a set of points $x_0, x_1, \ldots, x_n$, each of them representing a point along the path. In order to perform collision detection with the objects of the scene, intersections between the straight segments x_{i-1}, x_i and the geometry are calculated. Several numerical methods will be tested in this paper.

2.2 Global illumination

The algorithm in Sect. 2.1 works for computing paths in which the origin point and outgoing direction is known (as from the camera or from the lights), and the destination point is unknown and has to be computed. Although straight paths are trivially traced between two points, obtaining the path that obeys the differential equation (1) and traverses those two points is computationally very expensive.

In [7], a solution to this problem is introduced which avoids shooting shadow rays by computing direct illumination using photon mapping [10]. Based on photon mapping, it consists of two different steps: photon tracing from light sources and radiance estimation from the stored photons. Curved photon mapping is based on the same idea, but both photon and ray paths are curved, and direct light is also included in the photon map. As a consequence, it is not needed to trace shadow rays, and therefore

global illumination in inhomogeneous media can be simulated. As a tradeoff, the computation of direct illumination becomes by itself less efficient than using a direct ray tracing approach.

3 General considerations for optimization techniques

The presented algorithm, although capable of providing a complete solution for light transport in non-linear media, takes several minutes per frame. Assuming that the medium traversed by light or the scene has certain properties, several optimizations techniques (presented below) can be employed. This results in a significant reduction of the computation time. Also, different numerical methods seem to be better suited on a per-scene basis, depending on the gradients of the index of refraction.

For simplification, the coordinates used in the explanations will be camera coordinates: x and y axes will determine horizontal and vertical in the projection plane, while the z axis will determine the perpendicular to the projection plane (see Fig. 1).

3.1 Factors of computation time

There are several factors that influence the computation times of the curved ray tracing algorithm. Some of these are related to the standard ray tracing and photon mapping algorithms, while others are specific to the curved ray tracing algorithm presented in Sect. 2.1.

The factors coming from traditional direct ray tracing, as well as from photon mapping, are:

- Number of rays (n_{rays}). The number of rays that will be traced.
- Time for intersection ($t_{\text{intersection}}$). Average time of intersecting a straight segment with the geometry. This depends mainly on the number of polygons of the geometry and on the structure and the way they are stored and intersected (kd-tree, octree, BSP tree, grid of voxels or instant ray tracing [17]).
- Number of photons ($n_{(\text{photons})}$). The number of photons that will be traced from the light sources to the scene.

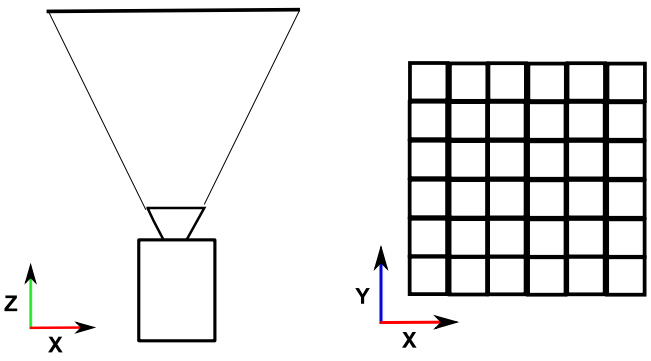

Fig. 1. Camera coordinates. (*Left*) View from top. (*Right*) View of the projection plane of the camera

On the other hand, the factors that are inherent to the introduced curved ray tracing algorithm are:

- Number of steps (n_{steps}). The average number of steps required for the numerical method until the intersection with the geometry, depending on the geometry itself, on the chosen numerical method and in the traversed medium.
- Time for step advance (t_{step}). This is the average time required by the numerical method for computing a new step (calculating net point x_{i+1} from net point x_i), that depends only on the chosen numerical method.

Following these factors and this notation, the time required for the curved ray tracing algorithm without any optimization is:

$$t_{\text{total}} = n_{\text{rays}} \cdot n_{\text{steps}} \cdot (t_{\text{intersection}} + t_{\text{step}}) \tag{2}$$

while for photon shooting:

$$t_{\text{total}} = n_{\text{photons}} \cdot n_{\text{steps}} \cdot (t_{\text{intersection}} + t_{\text{step}}). \tag{3}$$

For the different optimization techniques that will be presented in the next few sections, 3D transformations and interpolations are required. For completeness we introduce the corresponding symbols for both operations here; the influence of these times will be explained in the corresponding sections:

- Linear transformation time ($t_{\text{transform}}$). This is defined as the time required for transforming a 3D point using a linear transform, basically multiplying vector and matrix.
- Interpolation time ($t_{\text{interpolation}}$). The time required for interpolating (linear interpolation) a 3D point from a set of 3D points.

Specific data structures for geometry (kd-tree, octree, etc.), polygon number reduction techniques or instant ray tracing [17] are techniques that also improve efficiency, as they do on standard ray tracing. However, being already well-documented, their effect are out of the scope of this paper.

4 Optimization techniques for ray tracing

As stated before, this paper is an extension of the work presented in [12] whereseveral optimization techniques were introduced. These optimization techniques are here further developed by adapting them also for photon map-

ping. An overview can be seen in next subsections, while Sect. 5 will present the photon mapping adaptation.

4.1 Parallelization

Although the presented algorithm is not parallel in [8], being based on ray tracing an adequate parallelization is straightforward. The computation of each single curved ray is independent of the computation of the rest of rays, and therefore can take place in a different processing unit. As a consequence, the overhead of rendering on multiple processing units is negligible compared to the improvement on performance. Our system includes parallel computation on a single machine (shared memory).

4.2 Symmetries

Most atmospherical effects (like mirages) can be seen even when the index of refraction of the medium depends only on the Y axis (vertical), and as a consequence, they present a symmetry in YZ plane. If the index of refraction presents this symmetry, then the path of different light rays will also follow this symmetry (trivial deduction from Eq. 1). As a consequence, we can use a linear transform (symmetry) to calculate the different net points $x_0, x_1, \ldots, x_n$ of the right half of the curved rays from the points of the left half.

The computing time using this technique is:

$$t_{\text{total}} = n_{\text{rays}} \cdot n_{\text{steps}} \cdot \left(t_{\text{intersection}} + \frac{t_{\text{step}} + t_{\text{transform}}}{2} \right). \quad (4)$$

4.3 Undersampling

Assuming the gradient of the index of refraction in the medium is locally moderate, the technique of undersampling can be used. Some paths are computed normally, while other paths have their net points $x_0, x_1, \ldots, x_n$ interpolated from the net points of nearby paths (traversing nearby pixels). Since we have assumed moderate local gradients, the gradients do not differ greatly from one path to nearby paths, and as the gradient is what defines the path (see Eq. 1), the interpolated path and the path which would have been computed with the algorithm, will not present visible errors.

Let α be the rate of interpolated paths compared to the total paths, then the computing time of a frame is:

$$t_{\text{total}} = n_{\text{rays}} n_{\text{steps}} (t_{\text{intersection}} + (1-\alpha) t_{\text{step}} + \alpha t_{\text{interpolation}}) \quad (5)$$

4.4 Path caching

If the camera is static, whereas lights and geometry are dynamic, the path of each of the primary rays do not change from frame to frame. Taking advantage of this circumstance, it is possible to keep in several linked lists the net points of the paths calculated in one frame and re-use them in following frames, so from frame to frame the only calculations that have to be performed are:

- The intersection of each of the segments of the path with the geometry
- The extra segments that were not calculated in the previous frame because the path intersected on a segment and stopped the calculations

As a consequence, the time required in the first frame remains the same as if no optimization had been done (Eq. 2), and in the best circumstances (no extra segments required) the computing time of the rest of the frames is:

$$t_{\text{total}} = n_{\text{rays}} \cdot n_{\text{steps}} \cdot t_{\text{intersection}}. \quad (6)$$

Notice that in this way we have removed the t_{step} time completely, as the calculations of the step are no longer required. This technique becomes really useful with complex numerical methods that give an accurate solution in just a few steps.

4.5 Smart step selection

After any optimization, the expression given for time is given by the following equation:

$$t_{\text{total}} = n_{\text{rays}} \cdot n_{\text{steps}} \cdot t_{\text{intersection}} + C \quad (7)$$

where C is a constant that represents the overall time related to the numerical method, which is affected directly by the previous optimizations. If we were able to predict exactly at which step the segment finds its first collision, then we would require just one intersection per ray. Consequently, the resulting time would be:

$$t_{\text{total}} = n_{\text{rays}} \cdot t_{\text{intersection}} + C \quad (8)$$

and if compared to Eq. 7, the time is n_{steps} faster.

In order to make an intelligent prediction of the chosen step, we take into account the following:

- Spatial vicinity. If one of the rays hits a surface in the step n, there is a high probability that the rays corresponding to nearby pixels/solid angles will hit the same surface in the step $n-1$, n or $n+1$.
- Temporal vicinity. If one of the rays hits a surface in a frame in the step n, there is a high probability that in the following frame, the ray corresponding to the same pixel/solid angle will hit the same surface in the step $n-1$, n or $n+1$.

This way, the prediction can be done by looking at the segment at which either the same ray in the previous frame or the nearby rays have collided. Therefore, if the original ray collided at step n, we first check for intersections in the step n of the new ray, then in the step $n-1$ and then in the

step $n+1$. If none detect a collision, then the intersections are checked as in the original algorithm (starting from the first step). In the best case (step n being the right prediction) the resulting computation time remains as stated in Eq. 8, since each ray requires the computation of a single intersection.

5 Optimization techniques for particle tracing

Section 2.2 shows how a physically correct global illumination in inhomogeneous media can be simulated by the curved photon mapping algorithm, thus avoiding tracing shadow rays. The previously presented optimization techniques can also be optimized. In the next subsections we show how to adapt curved ray tracing optimization techniques so they can be applied for curved photon mapping.

5.1 Parallelization

Photon shooting can be as easily parallelized as was shown with ray tracing. However, it is important to remark that while building the photon map, if two or more threads are inserting photons in the map, it could happen that one thread writes the photon at the last position before updating the photon counter, so another thread can write a photon in the same position, erasing the information of the previous photon. This means that the structure of the photon map is not thread-safe (on writing). As a consequence, writing on the photon map (adding photons) should be done inside a mutual exclusion section. This means a slight reduction of performance in parallelization. However, querying the photon map for radiance estimation is thread-safe, as it does not require writing on it.

5.2 Symmetry

Outgoing directions from light sources are sampled (either uniformly or using importance sampling) randomly. However, the fact that this sampling is random is not compatible with the symmetry technique (Sect. 4.2), since it only works if the directions of the primary rays are known.

The symmetry technique can be somewhat adapted by sampling just one of the symmetric hemispheres instead of the whole sphere of directions of the light. For each sampled direction on one hemisphere, two particles are cast: one towards the sampled direction and the other one towards the symmetric direction. One of the two paths is computed using the algorithm presented in Sect. 2.1, while the other path is computed symmetrically to the first one. The application of this technique reduces the time of the first interaction of the photons with the scene to:

$$t_{\text{total}} = n_{\text{photons}} \cdot n_{\text{steps}} \cdot \left(t_{\text{intersection}} + \frac{t_{\text{step}} + t_{\text{transform}}}{2} \right). \quad (9)$$

5.3 Undersampling

Undersampling (Sect. 4.3) requires that the directions of the primary rays are known in advance. However, directions from light sources are sampled randomly, fact that limits its adaption to particle racing, as it does with symmetry (Sect. 5.2). However, this adaption is still possible. First of all, a threshold t parameter is set. This threshold is an angle and is measured in degrees. The second parameter to be set is n, a number which sets the number of paths that are taken into account to interpolate a new path.

When a new outgoing direction ($\boldsymbol{d}$) is randomly sampled, the undersampling algorithm acts in two different ways depending on the new sampled direction:

- If there exists a number of previously stored paths $n_p \geq n$ with direction $\boldsymbol{d}_i (i = 1, \ldots, n_p)$ that fulfill $\arccos(\boldsymbol{d}_i \cdot \boldsymbol{d}) \leq t$ (closer to $\boldsymbol{d}$ than the angle t) then the path of this sample is computed by interpolating it from the n closest stored paths, and is not stored for future interpolations.
- If the previous condition is not met, then the path followed by the photon is computed using the algorithm presented in Sect. 2.1. Every point $x_0, x_1, \ldots, x_n$ computed by the algorithm will be stored, associated to the direction $\boldsymbol{d}$. As a consequence, this path can be used for interpolations afterwards.

Our tests show that $t = 2$ and $n = 3$ are adequate parameters that reduce time while keeping negligible errors. The terms $t > 2$ and $n < 3$ will lead to higher errors and inaccuracies, while $t < 2$ and $n > 3$ will lead to more accurate but much more time-consuming simulations.

Let α be the rate of directions that are interpolated. Considering that directions are sampled uniformly:

$$\alpha = \left(\frac{2\pi \left(1 - \cos \frac{t}{2}\right)}{4\pi} \right)^n = \left(\frac{1 - \cos \frac{t}{2}}{2} \right)^n \quad (10)$$

while the time required for computing the paths coming from the light sources will be:

$$t_{\text{total}} = n_{\text{photons}} n_{\text{steps}} (t_{\text{intersection}} + (1-\alpha) t_{\text{step}} + \alpha t_{\text{interpolation}}). \quad (11)$$

This algorithm could in principle be extended for interactions with surfaces, by using the BRDF instead of the emission function, and thus the improvement on efficiency could be applied to interactions the same way that it is applied to the emission from lights. However, in standard photon mapping each interaction generates a single new photon, and therefore there would not be any net improvement.

5.4 Path caching

Adapting path caching (Sect. 4.4) to photon shooting is straightforward: instead of storing the paths followed by the rays, the paths followed by the photons that are shot from the light sources are stored. If a fairly large number

of photons is required, more paths have to be cached and thus more memory will be used. On the other hand, as it was shown in Sect. 4.4, this leads to a huge increase in performance on the photon shooting stage.

If light sources are static but the rest of the elements of the geometry are dynamic, this technique becomes very useful. Paths are stored in the first frame and are not calculated again. This technique can be easily combined with temporal coherence photon techniques [15], which would further increase performance. The time required for computing the paths coming from the light sources is given by:

$$t_{\text{total}} = n_{\text{photons}} \cdot n_{\text{steps}} \cdot t_{\text{intersection}}. \tag{12}$$

It is important to highlight that only the paths of the photons that are shot from the light source are required. The path of secondary photons (after interaction with any surface) might be stored too, but this would obviously require more memory, and if the previous interactions change position in image space, the stored paths should be discarded.

5.5 Smart step selection

The smart step selection technique (Sect. 4.5) can also be applied to particle tracing from the light source. A threshold angle t is set as the parameter of the method (depending on the scene, from two to four degrees will yield optimal results). For each sampled direction, the ordinal number of the step in which it has collided is stored. For each new sampled direction $\boldsymbol{d}$, if there exists a previously stored direction $\boldsymbol{d}_i$ so that $\arccos(\boldsymbol{d}_i \cdot \boldsymbol{d}) \leq t$ (closer to $\boldsymbol{d}$ than the threshold t), then the step associated to $\boldsymbol{d}_i$ is first checked for collisions in the new sample $\boldsymbol{d}$. This way spatial vicinity is considered. In order to also consider temporal vicinity, the same rule is applied, but directions $\boldsymbol{d}_i$ from the previous frame are also taken into account.

If there is more than one $\boldsymbol{d}_i$ that can be considered for a new sampled direction $\boldsymbol{d}$, then each has a priority. First, the ones from spatial vicinity are checked, and then, the ones from temporal vicinity are checked (previous frame). Inside each of the lists (spatial and temporal), the directions with smaller ordinal number of steps are checked first.

In the best case (always guessing the right step) the time needed for computing the paths coming from the light sources is:

$$t_{\text{total}} = n_{\text{photons}} \cdot t_{\text{intersection}} + C \tag{13}$$

where C is a constant that depends on the rest of the optimization techniques (see Sect. 4.5).

6 Results

6.1 Study of numerical methods

The chosen numerical method greatly influences computation time and efficiency, depending on the gradient of the index of refraction in the medium. We compare the following four methods [2]:

- Euler method. The simplest method, its main advantage is its low computation time, and main disadvantage is its inaccuracy. This method requires a very small step size for it to be of use.
- Order 2 Runge–Kutta method. This method is more accurate than the Euler method.
- Order 4 Runge–Kutta method. This method is more accurate than the order 2 version.
- Dormand–Prince method [3]. Similar to an order 4 Runge–Kutta method, this method uses an adaptive variable step size. Instead of setting the step size as a parameter (as the previous methods), it requires a tolerance, that is the maximum error that can be committed in a specific step. The step size varies in order to fit this tolerance.

The errors of the figures have been computed by choosing a very simple medium in which the index of refraction varies just in the vertical axis. The variation follows the equation of a straight line:

$$n = mx + b \tag{14}$$

where n is the index of refraction, x is the vertical axis and m and b are the parameters that define the slope and y intercept.

In [8] it is shown that this specific configuration of the index of refraction can be solved both analytically and numerically, so we compare the analytical solution with the numerical solution given by each of the numerical methods. The error of each numerical method is the distance between the end point given by the method and the end point given analytically. The time includes tracing a single ray through the medium in seconds.

Figure 2 includes the results of the several tests on the performance of the different numerical methods. Figures 2a, b show the performance of the methods considering variation in the corresponding parameter (step or tolerance). The distance of the ray is fixed to 1000 and the gradient (slope of the straight line) is 0.1. As expected, time increases when the step becomes smaller. Euler is more efficient than Runge–Kutta, and order 2 is more efficient than order 4. The Dormand–Prince method takes more time as tolerance is reduced, and is not comparable to other methods because it uses different parameters.

Regarding the committed error, Dormand–Prince (Fig. 2d) is more stable, because as tolerance decreases, error decreases. This is due to the fact that this method uses an adaptive step. The methods which have a fixed-step (Fig. 2c) present a different behavior. Error is large both when the step is either too large or too small. If the step is too large, the error committed at each step is accordingly too large; if

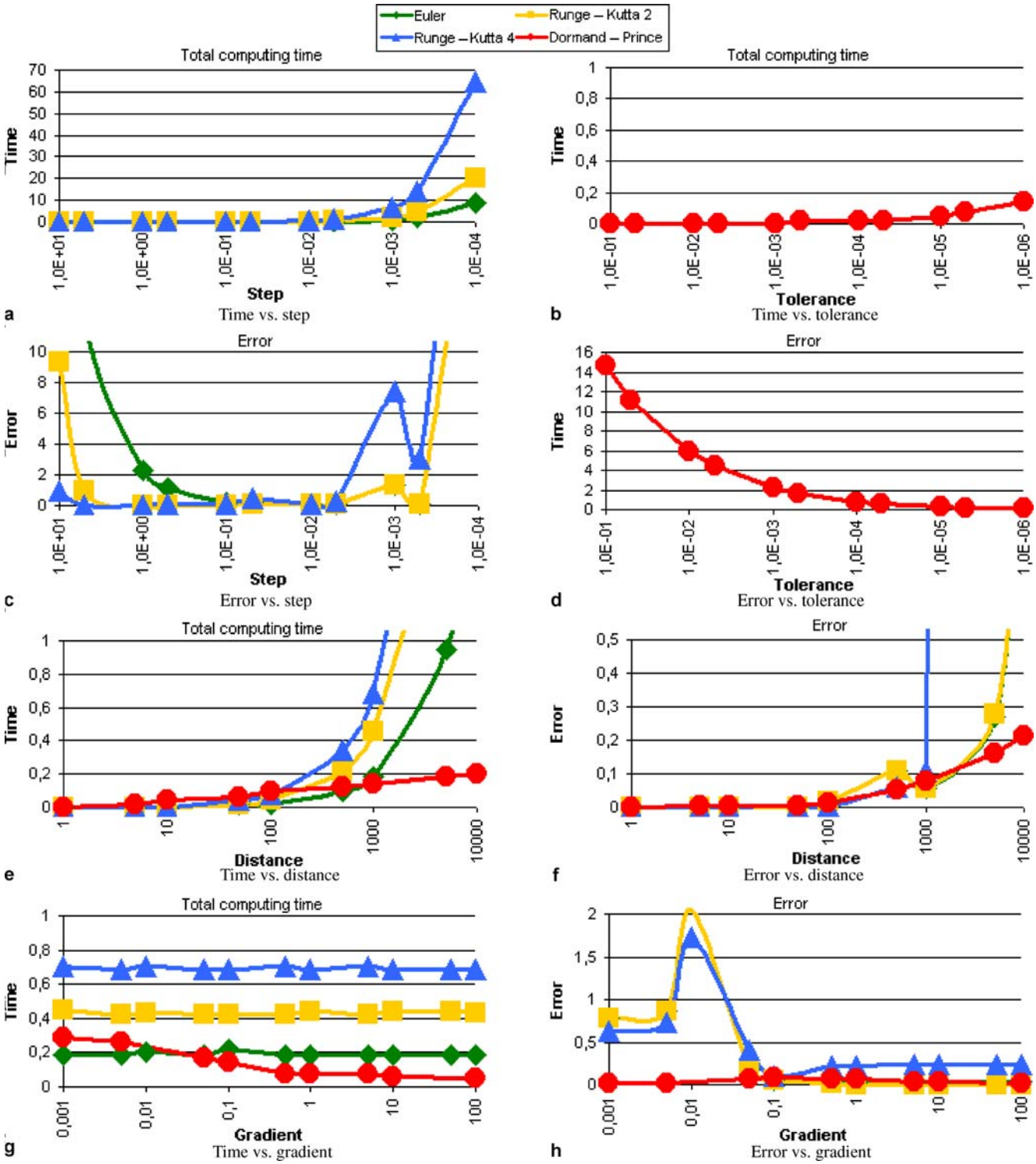

Fig. 2. Performance of numerical methods

the step is too small, the error at each step is too small. But, a huge amount of steps is required, and as a consequence the accumulated error increases.
The influence of the distance traveled by the ray has also been studied, with step size 5×10^{-3} for the methods with fixed-step and tolerance 10^{-6} for Dormand-Prince. The slope is still 0.1. Figures 2e, f show the result. Surprisingly, in short distances, the fixed methods perform slightly better both in time and error. This is due to a correct choice of the step size. In long distances, the Dormand-Prince method performs much better by far.
Figures 2g, h show the influence of the gradient (slope) on the performance of each method. The parameters are the same than in the study of the effect of distance, and the distance is fixed to 1000. Time is not affected by the gradient for fixed-step methods, as the number of steps that they compute is exactly the same. However committed varies depending on the gradient (being too high at low gradients). On the other hand, Dormand–Prince, having an adaptive step size, has a varying time depending on the gradient (being high at low gradients). However the committed error is quite low under all gradients, as the step adapts in order to reduce it.
In conclusion, for specific circumstances (short distances and specific gradients) fixed-step methods seem to perform better, if choosing the right step, with Euler being more efficient and Runge–Kutta (order 4) more accurate. However, it is not always possible to find exactly the correct step for a certain simulation, plus under general circumstances, Dormand–Prince performs better. As a consequence, Dormand–Prince will be used in all the cases, unless the scene is especially simple and small (circumstances under which fixed-step methods perform better).

6.2 Test scenes

In order to check rendering times and committed error, four very different test sceneshave been taken into account. Figure 3a presents an inferior mirage. The most interesting characteristic of this scene for our simulation methods is its simplicity. The medium has just one gradient of its index of refraction (near the floor). The shuttle is located 2 km from the camera. Figure 3b presents a Novaya Zemlya effect. The medium is as simple as the inferior mirage. However, the geometry (the sun) is much further away (150×10^6 km from the camera) and as a consequence, numerical methods require far more steps. Figure 3c presents the Fata Morgana effect. The geometry is close (12 km) as in the inferior mirage, but the medium is more complex, with two strong gradients of the index of refraction at different heights. Figure 3d presents a gradient of a medium in a scene in which light enters a room through a cross-shaped window. The gradient has been exaggerated for demonstration purposes. The dimensions of the room are $5 \times 5 \times 5$ m. In this test scene, the effect of the optimization techniques with particle tracing is considered.

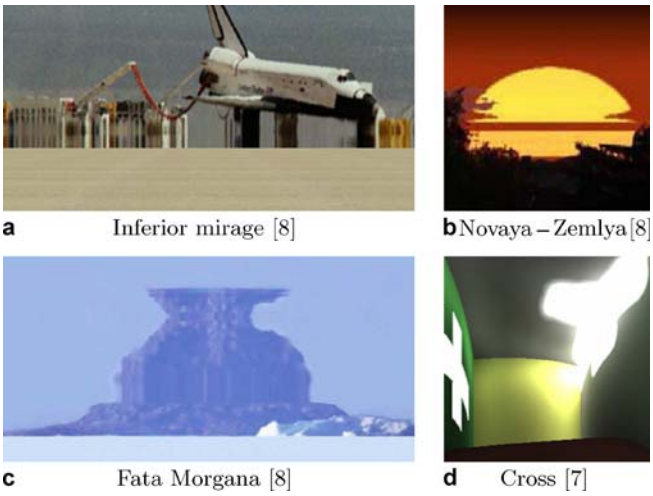

Fig. 3. Test scenes

The index of refraction of the inferior mirage, Fata Morgana and Novaya Zemlya are based on the model described in [8]. In the cross scene, the index of refraction follows the equation $n = 0.1y + 1$, where n is the index of refraction and y is the vertical axis.

6.3 Efficiency due to optimization techniques

In Table 1, the computation times (in the format minutes:seconds) and the committed errors (considering kilometers as the unit of the scene, errors compared to the ground truth) in a single frame of each test scene are presented. The kilometers are considered as the average error of each collision point (of rays in the three previous scenes and of photons in the cross scene, sampling the same shooting directions).

The undersampling technique for ray tracing has a parameter $\alpha = \frac{1}{2}$, photon mapping undersampling has the parameters $t = 2$ and $n = 3$, and smart step selection in photon mapping has a parameter $t = 3$.

Ground truth refers to the scene simulated using the algorithm presented in Sect. 2.1 with the Dormand–Prince numerical method, with a tolerance of 10^{-6} km. Inferior mirage and Fata Morgana scenes have been rendered at 800×400 pixel resolution, while Novaya Zemlya and cross scenes have been simulated at 400×400 pixel resolution. A total of 500 000 photons were shot on the cross scene, and the radiance estimation was done using 300 photons. All the scenes have been rendered using one ray per pixel, on a Pentium IV at 2.8 GHz (with hyperthreading technology).

As stated before, the technique of path caching is related to the frame-to-frame coherency and therefore to animations. However, in the time data of Table 1, we are considering a single frame. In order to achieve meaningful single frame data we computed before the previous frame

Table 1. Results of the tests

Technique	Inferior mirage		Novaya Zemlya		Fata Morgana		Cross	
	Time	Error	Time	Error	Time	Error	Time	Error
Ground truth	8:15	0.000	10:12	0.000	9:42	0.000	14:10	0.000
Parallelization	4:32	0.000	5:58	0.000	5:11	0.000	11:32	0.000
Symmetries	6:04	0.000	8:14	0.000	7:32	0.000	12:26	0.000
Undersampling	5:54	0.005	8:01	0.230	6:58	0.011	11:51	0.004
Path caching	4:12	0.000	4:42	0.000	4:22	0.000	11:05	0.000
Smart step sel.	3:57	0.000	4:21	0.000	3:56	0.000	11:10	0.000
Combination	1:55	0.019	2:51	0.032	2:02	0.042	9:31	0.002

Fig. 4. Sample scene that has been simulated using the different techniques

(the one that generates the initial rays) and then the measured frame. As for the combination technique, this refers to a single frame, computed using parallelization, undersampling and the smart step selection techniques.

The effect of the optimizations is not as impressive in the cross scene, due to the fact that the bottleneck of the simulation is not the curved paths but the radiance estimation of the photon mapping technique. On the other hand, the effect of these techniques happens to be more perceivable on the Fata Morgana scene, in which the traverse medium is more complex than the rest of the scenery.

Although most of the techniques yield no error because of the underlying theory behind them, the undersampling technique does. However, its errors are negligible compared to the dimensions of the geometry of the scene. Furthermore, the resulting images are visually indistinguishable from the original. For instance, conducting a visual test with the inferior mirage scene, we obtain the results of Fig. 4. The two simulations that have some error are Figs. 4b, c, and they show that the errors are not perceptible.

7 Conclusions

Several tests have been done with different numerical methods, to check their suitability and efficiency on simulating non-linear media. As a general conclusion, the Dormand–Prince method usually yields an overall better efficiency, yielding less errors.

A wide range of optimization techniques have been developed for the curved ray tracing algorithm. The techniques and algorithm can be combined to improve the efficiency of the rendering process of the scenes in which the medium traversed by light has an inhomogeneous index of refraction. Consequently, simulation of atmospheric phenomena requires less computation time. Those techniques have also been adapted for photon mapping, in order to accelerate the simulation of global illumination in non-linear media.

Also, different optimization techniques can be applied under different scenes, animation setups and media profiles. As a consequence, a wide range of possible phenomena can be simulated, applying one or more of the presented techniques, and taking advantage of the reduced simulation time.

We have obtained speedups of up to five over the original algorithm; additionally, the extra efficiency gained from the use of the correct numerical method has also further reduced computation times.

8 Future work

In order to further optimize the algorithm, the possibilities offered by GPUs should be taken into account. The different combinations of the presented algorithm, its optimizations and existing GPU algorithms might reduce further the computation times, maybe even reaching interactive rates. Also, parallelization on multiple machines is being considered.
We are considering the possibility of using new numerical methods, like multi-step and predictor-corrector methods [2], in order to check if they improve efficiency and how they adapt to different optimization techniques and geometrical complexities.

Other global illumination methods should be studied to obtain a full light transport solution. It should be checked whether they can be used when non-linear media are present, the errors they commit, their efficiency and the suitability of the different optimization techniques and numerical methods.

Acknowledgement This research was partly done under the sponsorship of the Spanish Ministry of Education and Research through the project TIN2004-07672-C03-03TIN2004.

References

1. Berger, M., Trout, T., Levit, N.: Ray tracing mirages. IEEE Comput. Graph. Appl. **10**(3), 36–41 (1990)
2. Burden, R.L., Faires, J.: Numerical Analysis, 4th edn. PWS-Kent, Boston (1988)
3. Dormand, J., Prince, P.: A family of embeded Runge–Kutta formulae. J. Comput. Appl. Math. **6**(1), 19–26 (1980)
4. Glassner, A.S.: Principles of Digital Image Synthesis. Morgan Kaufmann, San Francisco (1995)
5. Gröller, M.E.: Nonlinear raytracing: visualizing strange worlds. Vis. Comput. **11**(5), 263–274 (1995)
6. Gutierrez, D., Muñoz, A., Anson, O., Serón, F.J.: Non-linear volume photon mapping. In: Rendering Techniques, pp. 291–300 (2005)
7. Gutierrez, D., Serón, F.J., Anson, O., Muñoz, A.: Chasing the green flash: a global illumination solution for inhomogeneous media. In: SCCG'04: Proceedings of the 20th Spring Conference on Computer Graphics, pp. 97–105. ACM, New York (2004)
8. Gutierrez, D., Serón, F.J., Muñoz, A., Anson, O.: Simulation of atmospheric phenomena. Comput. Graph. **30**(6), 994–1010 (2006)
9. Hall, R.: Illumination and color in computer generated imagery. Springer, Berlin Heidelberg New York (1989)
10. Jensen, H.: Realistic image synthesis using photon mapping. Peters, Natick, MA (2001)
11. Linu, A., Haber, J., Magnor, M.: Realistic solar disc rendering. In: V. Skala (ed.) WSCG 2005 Full Papers Conference Proceedings, pp. 79–86 (2005)
12. Muñoz, A., Gutierrez, D., Serón, F.J.: Efficient physically-based simulation of non-linear media. In: GRAPHITE '06: Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia, pp. 97–105. ACM, New York (2006)
13. Musgrave, F.K.: A note on ray tracing mirages. IEEE Comput. Graph. Appl. **10**(6), 10–12 (1990)
14. Stam, J., Languenou, E.: Ray tracing in non-constant media. In: Proceedings of the Eurographics Workshop on Rendering Techniques '96, pp. 225–ff. Springer, Berlin Heidelberg New York (1996)
15. Tawara, T., Myszkowski, K., Dmitriev, K., Havran, V., Damez, C., Seidel, H.P.: Exploiting temporal coherence in global illumination. In: SCCG '04: Proceedings of the 20th Spring Conference on Computer Graphics, pp. 23–33. ACM, New York (2004)
16. USGPC: U.S. Standard Atmosphere. United State Government Printing Office, Washington, D.C. (1976)
17. Wald, I., Kollig, T., Benthin, C., Keller, A., Slusallek, P.: Interactive global illumination using fast ray tracing. In: EGRW '02: Proceedings of the 13th Eurographics Workshop on Rendering, pp. 15–24. Eurographics Association, Aire-la-Ville, Switzerland (2002)

ADOLFO MUÑOZ received his B.S. and M.S. degrees in Computer Science Engineering from the University of Zaragoza, Spain, in 2004. He is currently working on his Ph.D. degree and working as an Assistant Professor at the University of Zaragoza. He is also a member of the Advanced Computer Graphics Group (GIGA). His research interests include global illumination techniques, simulation of participating media and subsurface scattering and photorealistic and physically based rendering.

DIEGO GUTIERREZ is an Assistant Professor at the University of Zaragoza, Spain, where he received his Ph.D. in Computer Science, earning the Ph.D. Extraordinary Award at UZ. He joined the University's Advanced Computer Graphics Group in 1996. He is (or has been) a member of several committees, including Eurographics, the SIGGRAPH sketches program, Pacific Graphics and other ACM or EG-related conferences; he was also co-chair for ACM GRAPHITE 2006. His areas of interest include computational photography, global illumination, high dynamic range and perception. He has published more than 60 papers in international conferences and journals.

FRANCISCO J. SERÓN is a professor of Computer Science at the Technical School of Engineering at the University of Zaragoza. He received a Physical Science Degree from the University of Zaragoza in 1977 and a Ph.D. from the same University in 1984. At present he is the head of the Advanced Computer Graphics Group within the Computer Science Department. His research interests include simulation of natural phenomena, illumination engineering, and virtual reality.