INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Practical Self-Shadowing Algorithm for Interactive Hair Animation

Florence Bertails — Clément Ménier — Marie-Paule Cani

N° 5465

Janvier 2005

Thème COG

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--5465--FR+ENG

A Practical Self-Shadowing Algorithm for Interactive Hair Animation

Florence Bertails, Clément Ménier , Marie-Paule Cani

Thème COG — Systèmes cognitifs
Projets Evasion et Movi

Rapport de recherche n° 5465 — Janvier 2005 — 18 pages

Abstract: This paper presents a new fast and accurate self-shadowing algorithm for animated hair. Our method is based on a 3D light-oriented density map, a novel structure that combines an optimized volumetric representation of hair with a light-oriented partition of space. Using this 3D map, accurate hair self-shadowing can be interactively processed (several frames per second for a full hairstyle) on a standard CPU. Beyond the fact that our application is independent of any graphics hardware (and thus portable), it can easily be parallelized for better performance. Our method is especially adapted to render animated hair since there is no geometry-based precomputation and since the density map can be used to optimize hair self-collisions. The approach has been validated on a dance motion sequence, for various hairstyles.

Key-words: Hair self-shadowing, interactive rendering, hair simulation.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Un algorithme pratique d'auto-ombrage pour l'animation interactive de chevelures

Résumé : Ce papier présente une nouvelle approche pour traiter l'auto-ombrage dans les chevelures de manière précise et efficace. Notre méthode se base sur une carte 3D de densité, orientée selon la direction de la lumière; cette structure tire parti à la fois d'une représentation volumique optimisée pour la chevelure, et d'une partition de l'espace orientée selon la lumière. L'utilisation de cette carte de densité nous permet de calculer interactivement l'auto-ombrage au sein d'une chevelure (quelques images par seconde pour une chevelure complète) sur un CPU standard. Independante du matériel graphique (et donc portable), notre application peut également être parallélisée facilement pour améliorer les performances. En outre, la méthode est particulièrement adaptée au rendu de cheveux animés car elle ne requiert aucun précalcul géométrique, de plus la carte de densité s'avère très utile pour optimiser le traitement des collisions entre cheveux. Nous avons validé notre approche sur un mouvement de danse, pour divers styles de coiffures.

Mots-clés : Auto-ombrage, rendu interactif, simulation de cheveux

1 First section

2 Introduction

Self-shadowing is of great relevance to the realistic appearance of hair as it greatly contributes to the impression of volume (see Figure 8). Considering the high number of thin, translucent fibers composing a human hair, this phenomenon is difficult to reproduce both accurately and efficiently.

Our work was especially motivated by the need for a simple, fast and accurate technique to render animated sequences involving dynamic hair. Recently, much effort has been made to achieve interactive frame rates in the simulation of dynamic hair [BKCN03, WL03, BCN03]. But usually, these good performances only include the cost for animation while realistic hair rendering is done offline. Bando *et al.* use billboards to render their hair animations at interactive frame rates, but they do not account for hair self-shadowing. Ward *et al.* [WLL$^+$03] represent hair with different LOD to get interactive simulations. Self-shadowing is processed using one of the GPU-based approaches mentioned hereunder.

Approaches targeting interactive self-shadowing are very recent and mostly rely on advanced GPU's capabilities [MKBR04, KS04]. Though very successful, these methods are currently very dependent on the hardware architecture, and remain difficult to implement. This paper investigates an alternative solution based on the CPU which turns out to be simpler to implement, more flexible, and which still yields interactive frame rates.

2.1 Previous Work

Realistic rendering of human hair requires the handling of both local and global hair properties. Local hair properties describe the way individual hair fibers are illuminated and then represented in the image space, whereas global properties define how the hair fibers interact together. Global hair properties especially include hair self-shadowing which plays a crucial role in volumetric hair appearance and which is the main focus of this paper.

2.1.1 Local Illumination

To render an individual hair strand, Kajiya and Kay earlier proposed a reflectance model [KK89] that has been widely used subsequently. Their model is composed of a lambertian diffuse component and an anisotropic specular component. Many later approaches have subsequently proposed further refinements to this model [LTT91, Ban94, Gol97, Kim02]. Recently, Marschner *et al.* [MJC$^+$03] measured the scattering from real individual hair fibers and proposed a physical-based scattering model accounting for subtle scattering effects (such as multiple specular highlights) observed in their experiments. In our approach, we use Kajya-Kay's reflectance model but our self-shadowing technique could be combined with any other local illumination model.

2.1.2 Self-Shadowing

Two main techniques are generally used to cast self-shadows into volumetric objects[1] : shadow maps and ray casting through volumetric densities.

In traditional depth-based shadow maps, the scene is rendered from the light point of view, and the depth of every visible surface is stored into a 2D *shadow map*. A point is shadowed if the distance between the point and its projection to the light's camera is greater than the depth stored in the shadow map. This algorithm is not adapted to render semi-transparent objects such as hair because it stores only a single depth per pixel. To handle the self-shadowing of semi-transparent objects, Lokovic *et al.* proposed an extension to the traditional shadow maps, called *deep shadow map* [LV00]. For each pixel of the map, the method stores a *transmittance* function (also called *visibility function*) that gives the fraction of light penetrating at every sampled depth along a ray casted from the pixel.

Kim and Neumann proposed a practical implementation of this approach, called the *opacity shadow maps* [KN01], and applied it to hair rendering. In their method, the hair volume is uniformly sliced along the light rays and each hair volume comprised between two consecutive slices is rendered from the light's point of view into the alpha buffer, leading to an opacity shadow map; final rendering is done by interpolating the different opacity shadow maps.

The same idea was recently exploited by other authors [MKBR04, KS04] to get a fast rendering by using recent GPU's capabilities. Koster *et al.* achieved real-time results by accelerating the implementation of the opacity shadow maps and by making some assumptions about the hair geometry. Mertens *et al.* used an adaptive clustering of hair fragments instead of an uniform slicing, which enabled them to interactively build a more accurate transmittance function.

In the case of semi-transparent volumetric objects, self-shadowing is often computed from ray tracing using a volumetric density representation of the object. Such methods were first applied to renderings of clouds or smoke [Bli82, KH84]. In a preprocessing step, Kajiya *et al.* compute a 3D grid of voxels containing density distributions of the underlying geometry. During ray tracing, the brightness of the ray is determined by accumulating the contribution of each voxel traversed by the ray.

To our knowledge, Kajya and Kay were the only ones who applied this kind of technique to hair [KK89]. In their approach, they used texels as a generalization of volume densities in order to account for the specific behavior of the micro surfaces composing hair.

Methods based on ray tracing can often be very prohibitive in terms of rendering time, as they require the calculation and the sorting of multiple intersections between the rays and the objects that need to be shadowed. Conversely, the key benefit of the approaches that are based on shadow maps is the light-oriented sampling of geometry, which makes the computation of accumulative transmittance straightforward. Actually, our method inspires from both. Combining a volumetric representation of density with a light-oriented sampling allows us to define a practical and interactive self-shadowing algorithm.

[1] Please refer to [WPF90] for a complete survey on shadowing methods.

2.2 Overview

Our goal is to provide an easy, accurate and efficient way of casting shadows inside hair. Our method has to be flexible enough to handle and accelerate simulations that involve animated hair.

Our main contribution is to propose a new algorithmic structure called a *3D light-oriented shadow map* that inspires from both traditional 3D density volumes and more recent 2D shadow maps as it combines an optimized volumetric representation of hair with a light-oriented partition of space.

The main advantages of our method are the following :

- Our application is portable, simple to implement and can render a whole hairstyle composed of thousands of hair strands interactively on a standard CPU. Furthermore, it can easily be parallelized to increase performance.
- The approach is especially adapted to animated hair since the algorithmic structures that we used are efficiently updated at each time step. Moreover we show that our data structures provide an inexpensive way of processing hair self-collisions.
- Our technique does not make any assumption about the hair geometry, and thus could be applied to render any hairstyle. It has been validated on various hairstyles, either static or animated with different kinds of motion.

Section 3 describes our 3D light-oriented shadow map structure. Section 4 explains how the self-shadowing process can be efficiently done by using this new structure. Section 5 deals with the two extensions of our method mentioned above : on the one hand, we show that our 3D map is very helpful to process hair self-collisions efficiently; on the other hand we provide a parallelized version of our algorithm that improves the global performance of the simulation. The last two sections discuss our results before concluding.

3 3D Light-Oriented Shadow Map

Our 3D shadow map is a uniform cubic voxel grid that associates to each voxel (or cell) a *density* value and a *transmittance* value.

The different hair models that we want to render are composed of a set of segments, but our algorithm could also apply to other kinds of geometry such as polygonal surfaces for example.

3.1 A Light-Oriented Local Frame

In our method, the light rays are assumed to be parallel (ie. coming from an infinitely distant source), which is a reasonable assumption for handling common lighting conditions like sun lighting. This point will be discussed in conclusion.

Instead of having a fixed-oriented structure like in previous approaches, our map is always aligned with the light direction. More precisely, the map is placed in a local frame $\mathscr{R} = (O, \mathbf{X}_{map}, \mathbf{Y}_{map}, \mathbf{Z}_{map})$

where $\mathbf{X}_{map}$ coincides with the normalized light vector $\mathbf{L}$ and O is the origin of the map (see Figure 2).

As we shall see in Section 4.2, this configuration is very helpful for computing the accumulated transparencies efficiently. Note that for non-animated data requiring "dynamic" lighting (ie. a moving light), this choice would not be appropriate since the material geometry is to be recomputed each time the light moves. But in our case, hair geometry needs to be updated at each time step, so the moving light case does not yield extra cost for us.

3.2 Object Space to Map Space

To occupy a limited memory, our data structure exploits the fact that during animation, the hair volume is always located inside a bounding box of constant spatial dimension : indeed hair always remains attached to a scalp, and hair strands are assumed to be inextensible. Storing hair elements can thus be done inside a bounded structure, provided we build a mapping function from the 3D object space to this 3D bounding space.

The spatial dimension of the map is thus fixed and only depends on the maximal length $l_{max}$of a hair strand. If the dimension of the map is superior or equal to $2 \times l_{max} + h_{max}$, where h_{max} is the maximal dimension of the head, it is ensured that the grid will always represent a bounding volume for the hair at any time step. Of course, the best choice for the dimension of the map is the minimal number satisfying the constraint above.

The size (or resolution) of the map (ie. the number of cells it contains) depends on the desired accuracy of self-shadowing. Some tests have been made in Section 6 to compare results using different map resolutions.

In the remainder of the paper, *NCELLS* will denote the number of cells in each direction $\mathbf{X}_{map}$, $\mathbf{Y}_{map}$ and $\mathbf{Z}_{map}$ of the map frame $\mathscr{R}$, and ds will represent the step of the map, ie. the spatial dimension of a cell (see Figure 1).

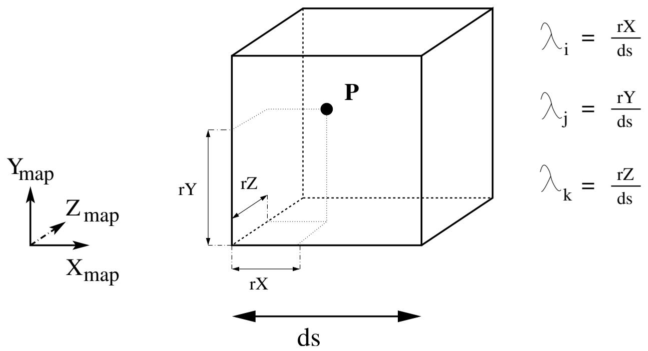

Figure 1: One cell of the map containing a point P. The λ_i parameters give the location of P inside the cell, and will be useful for the filtering process (see Section 4.3).

To find the index of the cell corresponding to a point $P(x,y,z)$, the coordinates of P are first expressed in the map frame $\mathscr{R}$ as $(x_{map}, y_{map}, z_{map})$, and then applied the following mapping function :

$$\begin{array}{cccc} \Psi: & \mathbb{R}^3 & \longrightarrow & [0 \cdots NCELLS]^3 \\ & \begin{bmatrix} x_{map} \\ y_{map} \\ z_{map} \end{bmatrix} & \longmapsto & \begin{bmatrix} \lfloor \frac{x_{map}}{ds} \rfloor \text{ mod } \mathtt{NCELLS} \\ \lfloor \frac{y_{map}}{ds} \rfloor \text{ mod } \mathtt{NCELLS} \\ \lfloor \frac{z_{map}}{ds} \rfloor \text{ mod } \mathtt{NCELLS} \end{bmatrix} \end{array}$$

Figure 2 shows the mapping between the object space and the map space.

Thanks to the mapping function Ψ, access to elements of the map is done in constant time, which greatly contributes to the efficiency of the method.

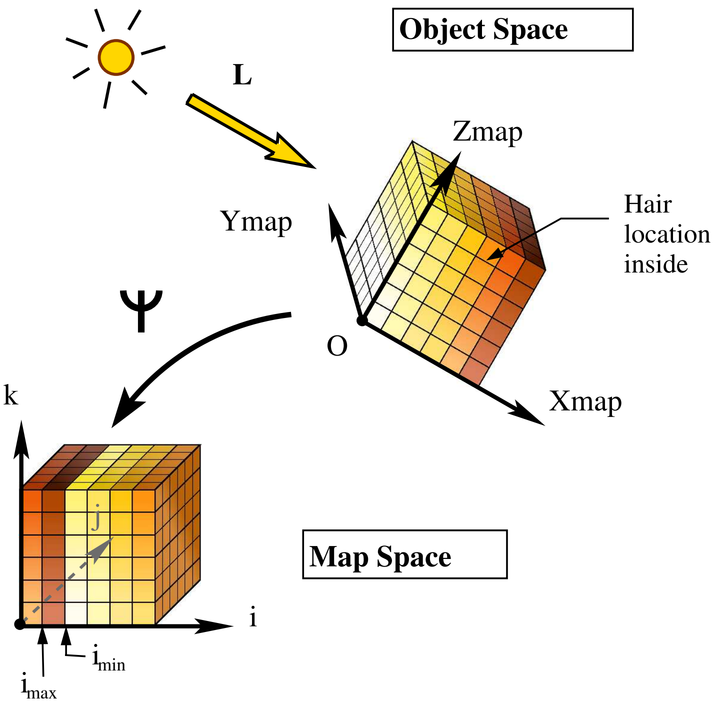

Figure 2: Correspondence between the object space and the map space. Because of the modulo operator in the mapping function Ψ, the first slice of the map (in light order) does not necessarily have the lowest index. The first slice and the last slice have consecutive indexes.

4 Self-Shadowing Algorithm

Our self-shadowing algorithm is composed of three main steps : hair density filling (1), transmittance computation (2), and filtering (3). Initially, each cell of the map has a null density (and we call it an *empty* cell).

The following figure summarizes the whole rendering pipeline[2].

4.1 Filling Hair Density into the Map

The first step of the algorithm consists of filling the map with hair density. This is simply done by traversing the hair geometry and doing the following operations :

- Each hair strand s_i is sampled using a Catmull-Rom spline into *nSmooth* points P_k^i;
- For each point P_k^i, the density of the cell $\Psi(P_k^i)$ is incremented.

[2]In our case, each hair strand is drawn as an OpenGL line strip

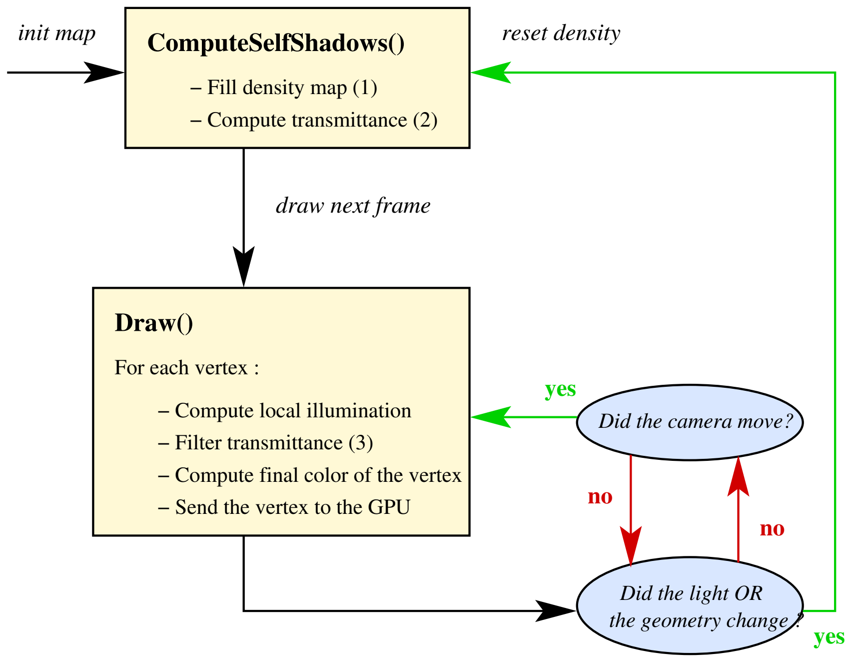

Figure 3: The rendering pipeline.

Of course the resulting density value obtained for one cell only makes sense relative to values of the other cells. Indeed, each isolated density value is arbitrary, and especially depends on the number of sample points used for each strand. Assuming that hair sampling is uniform, which is a reasonable assumption, the relative density multiplied by a scaling factor f approximates the light fall off through the corresponding cell; this quantity is commonly called the *extinction* parameter [LV00].

In practice, our hair sampling is the same as the one that is used at the final drawing stage, in order to ensure that each drawn vertex belongs to a non-empty cell.

4.2 Computing Transmittance

The fraction of light that penetrates to a point P of space can be written as [LV00] :

$$\tau(p) = exp(-\int_0^l \kappa(l')dl') \tag{1}$$

where l is the length of the path from the light to the point, and κ is the extinction function along the path.

The function τ is called the *transmittance* function. A sampled evaluation of τ can be done by accumulating transparencies of sampled regions along the light direction.

In our case, we need to evaluate the transmittance function at each cell of the map. To do this, we compute the transparency of each cell (i, j, k) as :

$$T_{i,j,k} = exp(-\kappa_{i,j,k}ds) \tag{2}$$

where the extinction coefficient $\kappa_{i,j,k}$ is computed using the density value of the cell (i, j, k), as explained before in Section 4.1 : $\kappa_{i,j,k} = f \times d_{i,j,k}$ where $d_{i,j,k}$ is the density of cell (i, j, k) and f is a scaling factor.

The transparencies are then composited together to get the final transmittance of each cell (i, j, k) :

$$T_{i,j,k} = \prod_{i'=i_{min}}^{i} exp(-d_{i',j,k}fds) \tag{3}$$

where i_{min} is the index of the map slice that is the closest to the light (see Figure 2).

As we mentioned in the previous section, the novelty of our approach in comparison with previous algorithms using voxel grids is that cells are sorted along the light direction : accumulating transparencies then becomes straightforward :

- A transmittance parameter $prevTrans$ is first initialized to 1 which is the proper value for a transparent and fully illuminated cell;
- The column (j, k) is traversed, starting from slice i_{min} (the closest slice to the light) until slice i_{max} (the furthest slice) :
  - If cell (i, j, k) is non-empty, its transmittance is set to $prevTrans \times exp(-d_{i,j,k}fds)$ (using Equation 3) and the parameter $prevTrans$ is updated to this value.

– Otherwise cell (i,j,k) is given the transmittance *prevTrans*.

Note that some empty cells might also be in shadow, since filling densities into the map does not necessary yield a connective set of non-empty cells. Even if only vertices belonging to non-empty cells will be drawn, giving a proper transmittance value to empty cells is important because such cells could be involved in the filtering process, if a non-empty cell has empty neighbors (see next section). The algorithm described above guarantees that every cell of the map has a proper transmittance value.

4.3 Filtering and Composing Colors

Before drawing hair primitives, it is necessary to filter transmittance values, otherwise regular patterns aligned with the density map will be quite visible, as shown by Figure 4.

For each point P that has to be sent to the GPU for final drawing :

- We compute the relative position of P $(\lambda_i, \lambda_j, \lambda_k)$ with respect to its corresponding cell $\Psi(P)$ (see Figure 1);
- We compute transmittance at point P by applying a trilinear interpolation as :

$$Trans(P) = \sum_{\substack{i' \in \{i-1,\ldots,i\} \\ j' \in \{j-1,\ldots,j\} \\ k' \in \{k-1,\ldots k\}}} A_{i'} A_{j'} A_{k'} Trans(i', j', k')$$

where $A_{i'} = \begin{cases} \lambda_i & \text{if } i' = i \\ (1-\lambda_i) & \text{otherwise} \end{cases}$

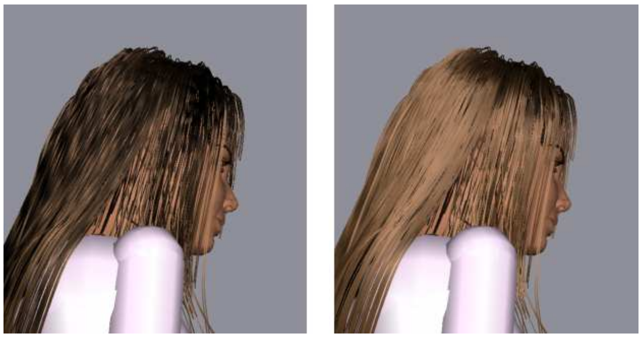

Figure 4: The effect of filtering the transmittance values. Left image : self-shadows without filtering : regular patterns aligned with the map are visible. Right image : self-shadows with filtering : artefacts have vanished, hair looks coherent.

(similar for $A_{j'}$ and $A_{k'}$)

- Finally, the color Φ_P of vertex P is obtained by the following equation :

$$\Phi_P = \Phi_{Ambient} + Trans(P) \times (\Phi_{Diffuse} + \Phi_{Specular}(P))$$

5 Extensions

5.1 Handling Hair Self-Collisions

Because of the high number of hair strands composing a human hairstyle, hair self-collisions represent a difficult and computational expensive issue in hair animation, and it often takes more than 80% of the simulation time.

An acceptable approximation of hair self-interaction consists of considering that internal collisions mainly account for the hair volume [LK01]. Starting from this assumption, hair density information is very useful : if the density is local over a fixed threshold (corresponding to maximum quantity of hair that can be contained in a cell), the hair strands should undergo constraints that spread them out.

Hair is animated using an approach inspired from hair guidance methods [DTKT93, CJY02]. In our case, hair is composed of approximately a hundred wisps where each wisp is simulated through three guide hair strands. Each guide hair strand is animated using a fast rigid links simulation [Ove91]. Final rendered hair strands are simply interpolated from the guide hair strands within each wisp.

Using the density map at each time step, hair self-collisions are processed by applying repulsive forces from the center of each cell having a too high density. The accompanying video shows that results look realistic although the method is extremely simple. Furthermore, this is a very cheap way to handle hair self-collisions (it only takes 2.5% of the whole processing time).

5.2 Parallelization of the Algorithm

One advantage of having a CPU-based algorithm is that parallelization can be considered in order to increase its efficiency. As a matter of fact, the described method is very well suited for such

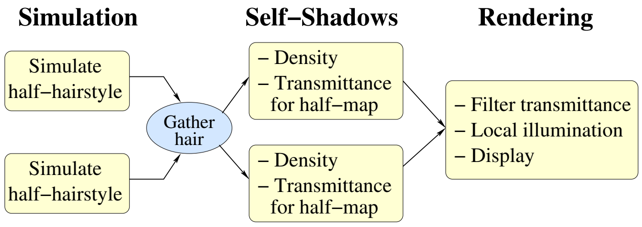

Figure 5: A parallel version of our algorithm.

a technique. We present here the parallel implementation of the simulation and self-shadowing algorithms.

- Simulation: thanks to the use of the density-map for handling self-collisions, each wisp can be simulated independently. This allows for a straight-forward parallelization where each processor computes a part of the hair, gathering at the end their partial results.
- Self-Shadowing: here again a straight-forward parallelization can be applied thanks to the fact that the map is light-oriented. As described in Section 4.2, the calculations for each column (j,k) can be done independently.

We have tested this implementation on a standard PC cluster and were able, using 3 CPUs, to easily double the frame rate in comparison with the single processor results given in the next section.

When trying to use more CPUs, the network gathering and sending of the vertices to the GPU became the main bottleneck. Sending vertex arrays directly to the GPU should reduce this bottleneck.

6 Results and Discussion

Our algorithm has been applied both to static and dynamic hairstyles. In each case we compare it with existing methods in terms of quality and performance.

Our accompanying video shows dance motion sequences rendered with our method.

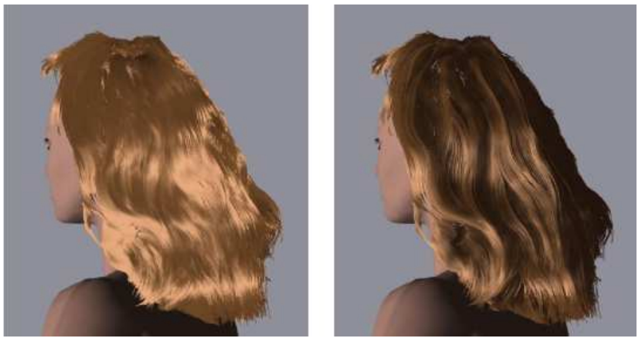

Figure 6: Applying our self-shadowing algorithm to a hairstyle captured from photographs by the method of Paris *et. al* [PBS04]. The hairstyle is composed of 87,500 hair strands (1,123 K segments) and it took 2 seconds to render it.

Figure 7: Evaluation of the quality of self-shadowing, using different map resolutions. From left to right : 32×32 with $ds = 0.5$; 64×64 with $ds = 0.2$, 128×128 with $ds = 0.1$ and 256×256 with $ds = 0.05$.

	Map reset + density	Trans	Filter + draw	Total rendering
Smooth	0.038	0.015	0.037	0.09
Curly	0.062	0.015	0.053	0.13

Table 1: Detailed performance of the rendering process (computing density, transmittance, filtering and final drawing) of a smooth hairstyle composed of 100K segments and a curly hairstyle composed of 200K segments. The results are expressed in **seconds per frame**; they have been obtained using an Intel P4 CPU at 3GHz.

6.1 Rendering Static Hair

Figures 8 and 6 show that our self-shadowing algorithm produces good visual results for merely synthetic hairstyles as well as for hairstyles captured from real hair geometry. We can see in Figure 6 that self-shadows make volumetric wisps stand out, whereas no self-shadows flatten the hair.

Figure 7 shows results obtained on curly hair when using different map resolutions. We can notice that for fine resolutions (128×128 or 256×256), curly wisps are properly shadowed and their shape is thus clearly visible, which is not the case for the coarsest resolutions. In practice, we found that a 128×128 resolution was sufficient to account for small shape details of hair.

In comparison with [MKBR04] our self-shadowing algorithm runs at a higher frame rate (11 FPS instead of 6 FPS for 100K hair segments).

6.2 Rendering Dynamic Hair

Figure 9 shows two snapshots from our hair animations. Our self-shadowing algorithm captures the fine discontinuities observed in a real hair during motion, as illustrated in Figure 10.

Figure 8: A dynamic hair without self-shadowing (left image) and shaded with our algorithm (right image). The 3D light-oriented map storing hair density and transmittance (middle image). The whole simulation (including animation *and* our rendering) is running interactively on a standard CPU.

Figure 9: A smooth brown hairstyle (100 K segments) and a curly red hairstyle (200 K segments) animated with different dance motions and interactively rendered with our algorithm.

	Anim	Hair self-collisions	Rendering	Total simu
Smooth	0.067	0.003	0.09	0.16
Curly	0.254	0.003	0.13	0.557

Table 2: Detailed performance of the simulation (animation, rendering and hair self-collisions) of two hairstyles composed of 134 animated wisps; the smooth hair style is composed of 100K rendered segments and the curly hair style is composed of 200 K rendered segments. The results are expressed in **seconds per frame**; they have been obtained using an Intel P4 CPU at 3GHz.

Table 2 gives the detailed performance of the whole simulation, including animation, hair self-collisions and rendering for both smooth and curly hairstyles. Note that the animation time is not the same for the two hairstyles, because it includes the update and the smoothing of the interpolated hair strands.

A hair composed of 3350 hair strands and 100K segments is thus completely simulated at an interactive frame rate of 6 FPS. Let us mention that for aesthetic results, we have also implemented hair-body collisions using a standard method based on spheres approximation. Handling such collisions makes the performance fall down to 3.5 FPS for the smooth hairstyle, and 1.5 FPS for the curly hairstyle (as mentioned in the video) but no optimization has been developed yet for that specific problem, considering that it was beyond the scope of this paper.

As shown in the video, the hair volume is properly generated using a repulsive force field based on local densities, as explained in 5.1. However, this method does not account for hair anisotropy nor wisps interpenetration. This could be done by adding more information to the map, such as hair orientation.

7 Conclusion and Future Work

We have presented a new hair self-shadowing algorithm based on a 3D-light oriented density map. Our approach can interactively render various hairstyles composed of thousands of hair strands, and yields convincing results. Our algorithm can easily be parallelized to improve the performance. Furthermore, we have shown that our density map is very helpful in accelerating the simulation process, as it can be used to handle self-collisions in an inexpensive way with visually good results. We are planning to use the hair density information again to optimize hair-body collisions.

For simplicity purpose, our approach makes the assumption of an infinitely distant source, which could be a limitation for rendering scenes illuminated by punctual sources. Yet, it seems that we could easily handle the case of punctual sources by only changing our mapping function Ψ : instead

Figure 10: Comparison between a real shadowed hair (left image) and our model (right image) with similar lighting conditions.

of considering an uniform square space partition, the new mapping function Ψ' should account for an angular space partition starting from the source point, and then sampled normally to the light rays. Provided the evaluation of Ψ' has a low cost, our algorithm should remain interactive and the parallelized version should work exactly the same.

Our method could also handle several light sources, by simply adding as many light-oriented maps as sources. The final transmittance of a point P would have to be interpolated between the transmittance values obtained from the different sources.

To get a better precision in our computations for a low cost, an interesting idea would be to follow the same approach than Mertens *et. al* [MKBR04] who build an adaptive slicing along a light ray and thus get a better approximation of the visibility function than approaches using a uniform slicing.

References

[Ban94] D. Banks. Illumination in diverse codimensions. In *Proceedings of ACM SIGGRAPH'94*, Computer Graphics Proceedings, Annual Conference Series, pages 327–334, 1994.

[BCN03] Y. Bando, B-Y. Chen, and T. Nishita. Animating hair with loosely connected particles. *Computer Graphics Forum*, 22(3):411–418, 2003. Proceedings of Eurographics'03.

[BKCN03] F. Bertails, T-Y. Kim, M-P. Cani, and U. Neumann. Adaptive wisp tree - a multiresolution control structure for simulating dynamic clustering in hair motion. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 207–213, July 2003.

[Bli82] James F. Blinn. Light reflection functions for simulation of clouds and dusty surfaces. In *Proceedings of the 9th annual conference on Computer graphics and interactive techniques*, pages 21–29. ACM Press, 1982.

[CJY02] J. T. Chang, J. Jin, and Y. Yu. A practical model for hair mutual interactions. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 73–80, July 2002.

[DTKT93] A. Daldegan, N. M. Thalmann, T. Kurihara, and D. Thalmann. An integrated system for modeling, animating and rendering hair. *Computer Graphics Forum*, 12(3):211–221, 1993.

[Gol97] D. Goldman. Fake fur rendering. In *Proceedings of ACM SIGGRAPH'97*, Computer Graphics Proceedings, Annual Conference Series, pages 127–134, 1997.

[KH84] James T. Kajiya and Brian P Von Herzen. Ray tracing volume densities. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 165–174. ACM Press, 1984.

[Kim02] T-Y. Kim. *Modeling, Rendering and Animating Human Hair.* PhD thesis, University of Southern California, 2002.

[KK89] J. Kajiya and T. Kay. Rendering fur with three dimensional textures. In *Proceedings of ACM SIGGRAPH'89*, Computer Graphics Proceedings, Annual Conference Series, pages 271–280, 1989.

[KN01] T-Y. Kim and U. Neumann. Opacity shadow maps. In *Rendering Techniques 2001*, Springer, pages 177–182, July 2001.

[KS04] M. Koster and H-P. Seidel. Real-time rendering of human hair using programmable graphics hardware. In *Computer Graphics International (CGI)*, pages 248–256, June 2004.

[LK01] D-W. Lee and H-S. Ko. Natural hairstyle modeling and animation. *Graphical Models*, 63(2):67–85, March 2001.

[LTT91] A. M. LeBlanc, R. Turner, and D. Thalmann. Rendering hair using pixel blending and shadow buffers. *The Journal of Visualization and Computer Animation*, 2(3):92–97, – 1991.

[LV00] Tom Lokovic and Eric Veach. Deep shadow maps. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 385–392. ACM Press/Addison-Wesley Publishing Co., 2000.

[MJC$^+$03] S. Marschner, H. Jensen, M. Cammarano, S. Worley, and P. Hanrahan. Light scattering from human hair fibers. *ACM Transactions on Graphics (Proceedings of the SIGGRAPH conference)*, 22(3):281–290, July 2003.

[MKBR04] T. Mertens, J. Kautz, P. Bekaert, and F. Van Reeth. A self-shadow algorithm for dynamic hair using density clustering. In *Proceedings of Eurographics Symposium on Rendering*, 2004.

[Ove91] C. Van Overveld. An iterative approach to dynamic simulation of 3-D rigid-body motions for real-time interactive computer animation. *The Visual Computer*, 7:29–38, 1991.

[PBS04] Sylvain Paris, Hector Briceño, and François Sillion. Capture of hair geometry from multiple images. *ACM Transactions on Graphics (Proceedings of the SIGGRAPH conference)*, 2004.

[WL03] K. Ward and M. C. Lin. Adaptive grouping and subdivision for simulating hair dynamics. In *Proceedings of Pacific Graphics'03*, September 2003.

[WLL$^+$03] K. Ward, M. C. Lin, J. Lee, S. Fisher, and D. Macri. Modeling hair using level-of-detail representations. In *International Conference on Computer Animation and Social Agents (CASA)*, May 2003.

[WPF90] Andrew Woo, Pierre Poulin, and Alain Fournier. A survey of shadow algorithms. *IEEE Computer Graphics and Applications*, 10(6):13–32, 1990.

Contents

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399