Rendu interactif de nuages réalistes

Antoine Bouthors, Fabrice Neyret, Nelson Max, Eric Bruneton, Cyril Crassin

▶ **To cite this version:**

Antoine Bouthors, Fabrice Neyret, Nelson Max, Eric Bruneton, Cyril Crassin. Rendu interactif de nuages réalistes. AFIG '07 - 20èmes journées de l'Association Française d'Informatique Graphique, Association Française d'Informatique Graphique (AFIG), Nov 2007, Marne la Vallée, France. pp.183-195. inria-00510249

HAL Id: inria-00510249
https://inria.hal.science/inria-00510249

Submitted on 27 Aug 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Journées de l'Association Francophone d'Informatique Graphique, Marne la Vallée, 2007

Rendu interactif de nuages réalistes

Antoine Bouthors[1] Fabrice Neyret[1] Nelson Max[2] Eric Bruneton[1] Cyril Crassin[1]

[1]EVASION - LJK / Grenoble Universités - INRIA [2]LLNL - UC Davis

Abstract

Nous proposons un algorithme simulant interactivement la dispersion anisotrope multiple de la lumière (multiple anisotropic scattering) *dans un milieu homogène. Contrairement aux méthodes temps-réel précédentes, nous reproduisons tous les types de chemins lumineux à travers le milieu et nous préservons leur caractère anisotrope.*

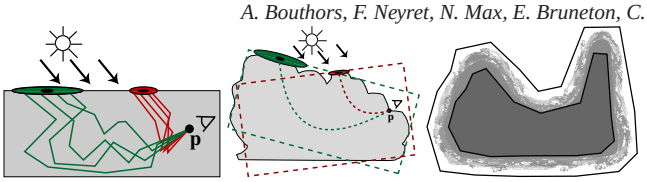
Notre approche consiste à estimer le transport d'énergie depuis la surface éclairée du milieu vers les pixels à rendre, en considérant séparément chaque ordre de dispersion. Nous représentons la distribution des chemins lumineux parvenant à un pixel donné du volume par la moyenne et l'écart-type de leur point d'entrée sur la surface illuminée, que nous appelons "aire collectrice". Au moment du rendu, nous déterminons sur la surface illuminée l'aire collectrice pour chaque pixel et pour différents groupes d'ordre de dispersion, et en déduisons le transport lumineux associé. La recherche rapide de l'aire collectrice et le calcul du transport lumineux est rendue possible grâce à une étude préliminaire de la dispersion multiple dans des formes simplifiées et ne nécessite pas de parcourir le volume.

Le rendu est réalisé efficacement dans un shader *sur le processeur graphique* (GPU)*, en utilisant un maillage de la surface du nuage enrichi par une Hypertexture ajoutant des détails à la forme. Nous présentons notre modèle avec le rendu interactif de cumulus animés et détaillés à 2-10 images par secondes.*

We propose an algorithm for the interactive realistic simulation of multiple anisotropic scattering of light in participating media. Contrary to previous real-time methods we account for all kinds of light paths through the medium and we preserve their anisotropic behavior.

Our approach consists in estimating the energy transport from the illuminated surface to the rendered pixel of the medium for each separate order of multiple scattering. We represent the distribution of light paths reaching a given viewed cloud pixel with the mean and standard deviation of their entry point on the lit surface, which we call the "collector area". At rendering time for each pixel we determine the collector area on the lit cloud surface for different sets of scattering orders, then we infer the associated light transport. The fast computation of the collector area and light transport is made possible thanks to a preliminary analysis of multiple scattering in plane-parallel slabs and does not require slicing or marching through the volume.

Rendering is done efficiently in a shader on the GPU, relying on a surface mesh augmented with a Hypertexture to enrich the shape. We demonstrate our model with the real-time rendering of detailed animated cumulus and cloudy sky at 2-10 fps.



(a) (b) (c)

Figure 1: *Vue d'ensemble de nos contributions.* (a) *Dans une analyse préliminaire, nous caractérisons le transport de la lumière dans une dalle via une aire collectrice représentant la zone d'entrée de la lumière pour chaque ordre de dispersion.* (b) *Nous utilisons cette caractérisation pour trouver les zones collectrices sur une forme de nuage quelconque et calculer le transport de la lumière.* (c) *Nos nuages sont représentés par une Hypertexture, avec des détails procéduraux sur les bords et un noyau homogène.*

1. Introduction

Le rendu réaliste de nuages est un problème difficile. La dispersion anisotrope de la lumière doit être simulée dans un volume, le manque d'absorption rend la convergence très lente, et les nuages détaillés requièrent des représentations volumiques à haute résolution. Dans l'optique du rendu temps-réel, ceci est encore plus difficile: les méthodes de radiosité volumique ou de Monte-Carlo ne peuvent pas converger en temps-réel, et le rendu volumique ne peut pas être effectué avec suffisamment de résolution pour des nuages détaillés tels que les cumulus. Les précalculs empêchent l'animation des nuages ou de la source de lumière, et la plupart des modèles temps-réel ne reproduisent pas les effets dépendant du point de vue ou des aspects visuellement importants tels que la rétro-dispersion (*backscattering*).

Pour résoudre ces problèmes, notre approche est de représenter les nuages par des volumes délimités par des surfaces et optimise le calcul du transport lumineux depuis la surface illuminée du nuage vers les pixels à rendre. Pour cela, nous étudions et caractérisons le transport de la lumière pour chaque ordre de dispersion.

Nos contributions sont:

- un nouveau modèle caractérisant le transport de la lumière (*i.e.*, la quantité d'énergie lumineuse transmise) entre la surface d'une dalle homogène de nuage (*i.e.*, un volume de nuage limité par deux plans parallèles) et un point **p** quelconque dans la dalle, pour chaque ordre de dispersion (voir figure 1(a));
- un nouveau modèle caractérisant la distribution, sur la surface d'une dalle, des points d'entrée des chemins lumineux arrivant en **p**, pour chaque ordre de dispersion. Nous appelons cette zone d'entrée l'*aire collectrice* ou collecteur (voir figure 1(a));
- un algorithme itératif déterminant cette aire collectrice sur un nuage de forme quelconque pour un point **p** quelconque, et calculant le transport lumineux associé (voir figure 1(b));
- une représentation efficace de la forme détaillée d'un nuage utilisant une Hypertexture [PH89] sur un maillage surfacique (voir figure 1(c));
- une implémentation sur GPU de ces contributions, résultant en un rendu détaillé de nuages animés en temps interactif prenant en compte la dispersion anisotrope multiple de la lumière à tous les ordres.

Contrairement à la plupart des méthodes temps-réel, notre modèle de dispersion multiple

- prend en compte les ordres élevés de dispersionj responsables des effets de diffusion et de rétro-dispersion, ainsi que les faibles ordres de dispersion responsables de la gloire, des silhouettes lumineuses et de l'apparence des parties fines;
- utilise la fonction de phase de Mie fortement anisotrope, basée sur la physique de la dispersion de la lumière par les petites particules (et non pas une fonction de Rayleigh, une Gaussienne, ou une fonction isotrope);
- ne nécessite pas de parcourir le volume du nuage, mais seulement sa surface;
- ne se base pas sur des précalculs dépendants de la forme du nuage.

Représenter la forme du nuage par une Hypertexture sous un maillage de surface nous permet de rendre efficacement des nuages dont les bords sont hétérogènes aussi bien que nets, contrairement aux méthodes basées sur des volumes tranchés. Nous pouvons ainsi rendre rapidement des nuages nets, cotonneux ou vaporeux.

2. Physique des nuages

2.1. Densité et taille des gouttelettes de nuages

Les nuages convectifs réels ne sont pas flous sur leur bords, mais nets ou effilochés (voir figures 5(c) et 5(d)). Les parties où la convection est faible peuvent avoir une couche de filaments plus large. Ces hétérogénéités dans le contenu en eau d'un nuage (*i.e.*, sa densité) constituent un aspect visuel fort des nuages. A l'intérieur d'un nuage, cette densité peut aussi varier – en particulier à l'approche des conditions de pluie – à cause de la coalescence des gouttelettes.

La taille de ces gouttelettes est caractérisée en un point donné par une distribution de taille de gouttelettes (*droplet size distribution* – DSD), qui est généralement modélisée dans la littérature par une distribution log-normale ou Gamma modifiée [Lev58]. Les propriétés optiques d'un nuage dépendent fortement de la taille des gouttelettes. Prendre la DSD en compte change radicalement la fonction de phase résultante, et donc l'apparence visuelle.

Cependant, ces données pour un nuage ne sont pas souvent disponibles et la physique qui les régit (*e.g.*, le mécanisme de coalescence ou l'évolution de la DSD) n'est pas totalement comprise. Étant donné que les applications en informatique graphique requièrent en priorité une plausibilité visuelle, les approximations de ces valeurs sont classiques. À titre d'exemple, les variations de densités sont visuellement plus importantes sur les bords des nuages – où elles sont directement visibles – qu'à l'intérieur – où elles n'influence l'apparence qu'indirectement, via les grands ordres de dispersion. Habituellement, la DSD est considérée comme constante pour tout un nuage (si tant est qu'une DSD soit utilisée).

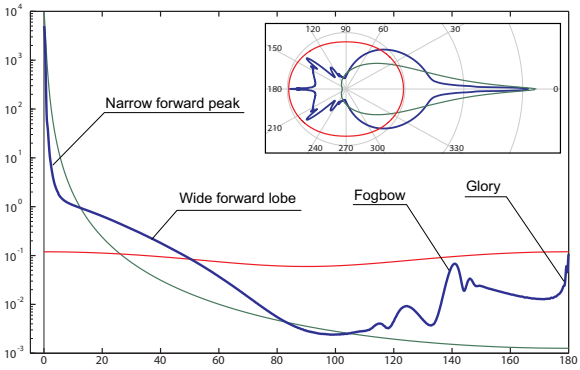


Figure 2: *Graphe logarithmique (encart: graphe logarithmique en coordonnées polaires) de fonctions de phases couramment utilisées.* Rouge: *Rayleigh.* Vert: *Henyey-Greenstein avec* $g = .99$. Bleu: *Mie.*

2.2. Fonction de phase

La fonction de phase d'une gouttelette d'eau est déterminée par la théorie de Mie et ne peut pas être approximée précisément par une Gaussienne, une fonction de Henyey-Greenstein, ou d'autres modèles. Comme on peut le voir figure 2, la fonction de Mie combine un fort pic en avant (51% de l'énergie), un large lobe en avant (48%), et un lobe arrière complexe comprenant des pics. Ces trois caractéristiques apportent chacune des effets très spécifiques et visibles à l'échelle du nuage. Une absence de pics arrières signifie que l'on n'aura ni gloire, ni arcs blancs (sortes d'arc-en-ciel créés par les nuages). Ignorer le pic avant implique une forte sous-estimation de la transmittance globale et de l'anisotropie. Les fonctions Gaussienne et Henyey-Greenstein contiennent bien un lobe en avant et facilitent les calculs, mais sont loin de donner des résultats visuels précis. La fonction de dispersion de Rayleigh est encore moins appropriée puisqu'elle est symétrique (50% de la dispersion est en arrière) et correspond à la dispersion par les molécules de l'air (qui donnent la couleur bleue au ciel), et non pas par les gouttelettes d'eau.

Dans le spectre du visible, l'albédo d'une gouttelette peut être considéré comme valant 1 puisque'il n'y a pas d'absorption (toute la lumière est dispersée). À noter que certains phénomènes atmosphériques communément attribués aux nuages sont en fait causés par d'autres éléments (*e.g.*, les arc-en-ciel sont produits par la pluie, les parhélies par les cristaux de glace en suspension dans l'atmosphère).

2.3. Anisotropie

La dispersion multiple est forte dans les nuages et montre souvent un caractère anisotrope. Les nuages couvrent des centaines, voire des milliers de mètres et le libre parcours moyen d'un rayon de lumière dans un nuage est d'environ 20 m. En conséquence, la plupart des rayons seront dispersés de multiples fois avant de sortir du nuage. A cause de la nature hautement anisotrope de la fonction de phase de Mie

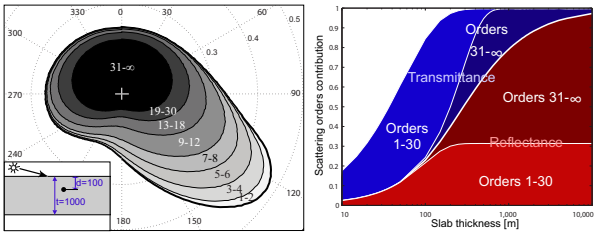


Figure 3: *Quelques résultats de notre analyse du transport de la lumière.* **Gauche :** *BSDF pour un point de vue à une profondeur de* $d = 100m$ *à l'intérieur d'une dalle de nuage d'épaisseur* $t = 1000m$*, avec un angle incident d'illumination de* $\phi_L = 75°$*. Les aires représentent la contribution de différents ordres de dispersion. Même les ordres élevés (jusqu'à 30 événements de dispersion) montrent un comportement anisotrope, tandis que les ordres* > 30 *montrent un comportement isotrope.* **Droite :** *Contribution des différents ordres de dispersion pour la réflectance (en rouge) et la transmittance (en bleu) d'une dalle d'épaisseur variable. Les ordres isotropes (31-*∞*) commencent à apparaître aux épaisseurs* $> 100m$*. Les ordres anisotropes (1-20) jouent un rôle dans la transmittance jusqu'à des épaisseurs de* $1000m$*, et contribuent à la réflectance à hauteur de* $30\% - 100\%$*.*

(99% de la lumière est dispersée en avant), même la dispersion multiple peut être anisotrope. D'après notre analyse du transport de la lumière dans une dalle, nous estimons que le comportement de la lumière est isotrope seulement après au moins 30 événements de dispersion. Par isotropie, nous entendons que la dispersion multiple se comporte comme si la fonction de phase du milieu était isotrope, et non que la BSDF (*Bidirectional Scattering Distribution Function*, fonction décrivant la distribution de la lumière selon l'angle de vue en un point donné) résultante est isotrope. En conséquence, l'illumination d'un nuage n'est pas dominée par les faibles ordres de dispersion, mais les ordres élevés ne se comportent pas tous de façon isotrope (voir figure 3).

3. État de l'art

Simuler la dispersion anisotrope multiple par des méthodes classiques de Monte-Carlo ou de radiosité volumique n'est pas possible en temps-réel. Les optimisations actuelles se basent sur des simplifications variées, que nous présentons dans cette section.

3.1. Fonction de phase

Étant donné que la fonction de Mie est complexe et coûteuse à calculer, elle est souvent approximée en synthèse d'images par d'autres fonctions comme la fonction de Henyey-Greenstein [Max94], une Gaussienne [PAT*04] ou même la fonction de Rayleigh [HL01] (voire commentaires en section 2.2). [REK*04, BNL06] précalculent la fonction de phase de Mie pour une DSD donnée, ce qui reproduit les aspect de la réalité (gloire, arc blancs, etc.). Nous utilisons la même approche pour notre fonction de phase.

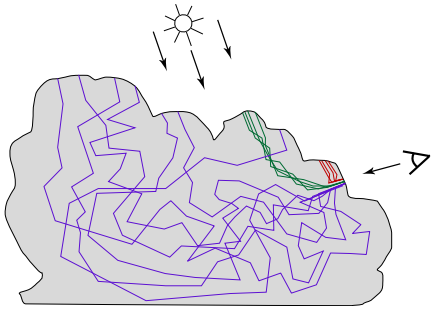


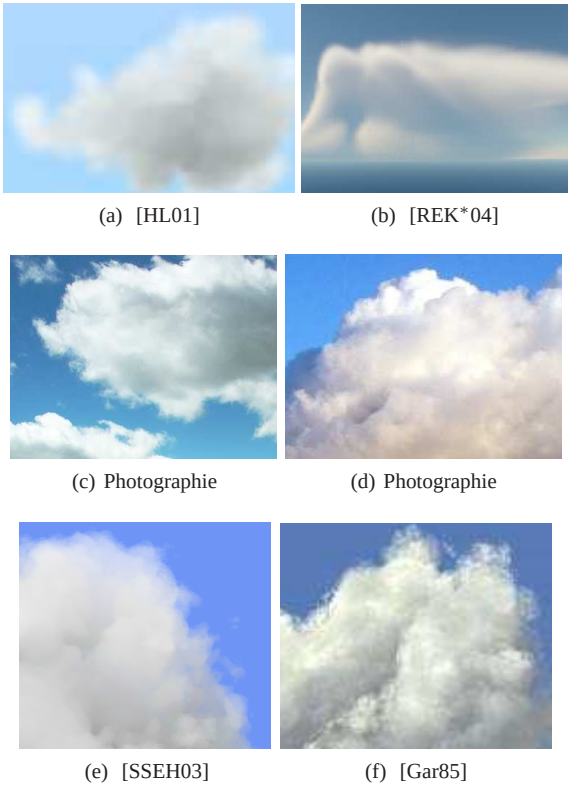
Figure 4: *Différent types de chemins lumineux dans les milieux participants, apportés par différents ordres de dispersions. Les faibles ordres de dispersion apportent des chemins courts et très courbés (en rouge). Les ordres plus élevés apportent des chemins longs et peu courbés (en vert). Les ordres les plus élevés apportent des chemins complexes, très étalés et diffusifs (en bleu). L'approche des* chemins les plus probables *[PAS03] reproduit les chemins verts, mais sous-estime les chemins rouges et bleus.*

3.2. Transport de la lumière

[Kv84] et [Bli82] considèrent soit un albédo faible (la lumière est peu diffusée), soit une densité faible. Dans ces cas, seul le *single scattering* (on ne prend en compte que les rayons ayant été dispersés une seule fois) est considéré. Ceci supprime tous les effets dus à la dispersion multiple. L'hypothèse que le transport de la lumière est principalement vers l'avant amène des algorithmes temps-réel en une seule passe telle que l'accumulation de tranches du volume [DKY*00, HL01, REK*04] mais empêche certains effets de la dispersion multiple tels que la rétro-dispersion. Calculer la dispersion multiple [NND96] seulement pour les ordres les plus faibles a les mêmes conséquence. L'approximation de diffusion [Sta95, JMLH01] permet des calculs efficaces mais néglige l'anisotropie dans la dispersion multiple.

[PAS03] ont introduit l'idée de *chemins les plus probables* (*Most Probable Paths* – MPP) dans les milieux participants. L'idée principale est que la majorité des photons arrivant à un point dans une direction ont grossièrement suivi le même chemin. En conséquence, intégrer le transport de la lumière uniquement le long de ce chemin est suffisant pour prendre en compte la majeure partie du transport. De plus, [PAT*04] accélèrent cette technique et tiennent compte de l'étalement spatial de la lumière autour de ce chemin moyen à travers une formulation analytique. Cette approche a été portée au temps-réel [HAP05] en utilisant le matériel graphique pour trancher le volume dans une manière similaire à [HL01, REK*04]. Ces méthodes basées sur deux passes de tranchage (accumuler le flux depuis la source de lumière dans les tranches, puis des tranches vers l'œil) limitent la variété de chemins pris en compte.

Comme indiqué dans [PAS03, HAP05], la principale restriction de l'approche MPP est que les chemins qu'elle calcule sont principalement d'ordre élevé et peu courbés. En conséquence, les chemins d'ordre faible, ainsi que les chemins diffusifs, sont sous-estimés. Ces chemins contribuent à la majorité de la luminance dans les parties fines et dans la rétro-dispersion, et ne devraient donc pas être négligés (voir figure 4).



(a) [HL01] (b) [REK*04]

(c) Photographie (d) Photographie

(e) [SSEH03] (f) [Gar85]

Figure 5: En haut: *cumulus virtuels temps-réels* **(a)** *et interactifs* **(b)** *utilisant des* billboards *ou des tranches.* Au milieu: *Cumulus réels effilochés* **(c)** *et nets* **(d)**. En bas: *Nuages virtuels utilisant un bruit 3D* **(e)** *et des surfaces avec détails procéduraux* **(f)**. *Ajouter des détails procéduraux donne une apparence moins floue et plus contrastée, augmentant le réalisme.*

Nous abordons ces restrictions en traitant les chemins lumineux de tous les ordres. Nous proposons également une nouvelle approche du calcul du transport lumineux, plus rapide, qui ne requiert pas de trancher ou parcourir le volume. Notre approche est également inspirée par les études sur le transfert radiatif dans des formes simples telles que les dalles [Cha60, KE86, Mob89, MMKW97]. Nous utilisons des représentations adaptées au matériel graphique telle que les *depth maps* (textures de profondeur) [DS03] pour implémenter notre algorithme de rendu sur GPU.

3.3. Représentation des champs de densités des nuages

Étant donné qu'il est difficile de mesurer ou simuler les données de densité de nuages, les premiers travaux ont

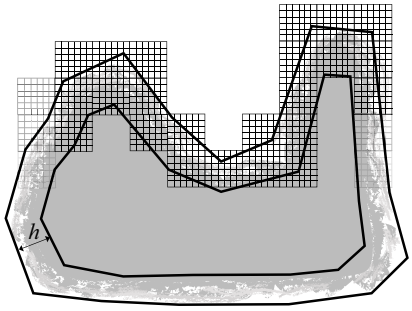


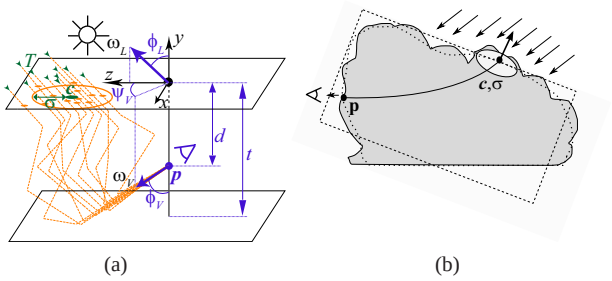
Figure 6: *Notre représentation de nuages. Un maillage est utilisé pour décrire la bordure extérieure à basse résolution. Une Hypertexture procédurale volumique ajoute des détails sous la bordure jusqu'à une certaine profondeur h à l'intérieur du nuage. Le cœur est considéré comme homogène.*

utilisé des modèles procéduraux [Gar85, DKY*00]. Les approches récentes utilisent des techniques de simulation de fluides [HL01] ou des données atmosphériques [TB02, REK*04]. Cependant, la simulation de fluides en 3D ne peut être faite qu'a une résolution grossière dans le cas d'applications temps-réel, et les données atmosphériques sont à basse résolution. En conséquence, il est nécessaire d'ajouter des détails de haute fréquence afin d'éviter une apparence floue (voir figure 5). [Ebe97] combine un bruit de Perlin [Per85] solide et des surfaces implicites. [SSEH03] advectent une texture de bruit [Ney03], mais ne tiennent pas compte du bruit dans le calcul d'illumination, ce qui donne au nuages une apparence uniforme (voir figure 5(e)). Ces idées nous ont inspiré pour combiner deux représentations – des maillages et des textures 3D – pour modéliser les nuages à deux échelles différentes.

Des primitives de rendu variées ont été utilisées pour rendre des nuages. Étant donné que les nuages sont une distribution spatiale de gouttelettes, des grilles volumiques ont souvent été utilisées pour les décrire, et les techniques de rendu volumique pour les rendre. Les méthodes utilisant des *billboards* ou des tranches de volume [HL01, PAT*04, REK*04, DKY*00] résultent en beaucoup d'*overdraw*[1], ce qui a un coût de rendu important. Si les tranches texturées sont une façon efficace de rendre des phénomènes gazeux, elles ne sont pas la meilleure options pour les nuages denses, où la plupart des pixels des tranches lointaines sont cachés par les plus proches, et donc inutiles à rendre. De plus, la mémoire disponible pour les textures 3D limite la résolution de tels modèles, ce qui résulte en des silhouettes floues et un manque de détails (voir figures 5(a), 5(b)).

Étant donné que les cumulus sont denses et ont souvent une interface nette, ils ont également été représentés par des volumes limités par des surfaces, tels que des ensembles d'ellipsoïdes [Gar85, ES00] ou des maillages [TB02].

[1] *i.e.*, un pixel donné est *rasterisé* beaucoup de fois par différentes primitives rendues.



(a) (b)

Figure 7: **(a)** *Schéma et notation pour le transport lumineux canonique T entre une aire collectrice $(\mathbf{c},\sigma)$ et un point* **p** *dans une dalle, dépendant des paramètres d'entrées $(\phi_V,\psi_V,\phi_L,d,t)$ pour chaque ordre de dispersion. À noter que ψ_L est toujours égal à 0 (le repère de référence est aligné avec la direction de la lumière).* **(b)** *Transport de la lumière dans un nuage entre sa surface éclairée et un point* **p**. *Pour chaque groupe d'ordres de dispersion, la lumière arrivant en* **p** *est considérée comme arrivant d'une aire collectrice. Nous caractérisons ce transport en associant une dalle à cette aire collectrice, puis en utilisant la fonction de transport canonique pour cette dalle.*

Pour éviter l'apparence "dure" des surface polygonales, on utilise un *shader* procédural simulant la silhouette détaillée et donnant une impression de volume (voir figure 5(f)). Le transport de la lumière n'est pas simulé. [BNL06] simulent le transport de la lumière à l'intérieur d'un maillage, mais ne considèrent aucun enrichissement sur les silhouettes, qui apparaissent ainsi polygonales et opaques.

Nous tirons parti des deux représentations (volumes et surfaces) en représentant les bordures des nuages à grande échelle par un maillage, et les variations de densité à haute fréquence sur les bords des nuages par une Hypertexture [PH89]. Le reste de l'intérieur du nuage est considéré homogène, ainsi seulement une mince couche de voxels est nécessaire (voir Figure 6).

4. Vue d'ensemble de notre méthode

Notre approche de rendu est basée sur une analyse du transport de la lumière pour chaque ordre de dispersion. Observant que les chemins d'ordres différents ont des comportements d'anisotropie et d'étalement différents, nous les traitons séparément:

- L'ordre 1 (*single scattering*), qui est le plus anisotrope et dépend des détails les plus fins, est calculé en utilisant une forme analytique de l'équation de dispersion (voir section 6.2);
- Les ordres 2-∞ (dispersion multiple) sont calculés en 8 groupes différents (2, 3-4, 5-6, 7-8, 9-12, 13-18, 19-30, 31-∞) en utilisant notre algorithme basé sur l'aire collectrice (voir section 6.1);
- L'opacité est calculée en intégrant la fonction d'extinction à travers le volume du nuage (voir section 6.3).

Notre approche basée sur l'aire collectrice suit et étend l'idée des *chemins les plus probables* (MPP) [PAS03]. Nous

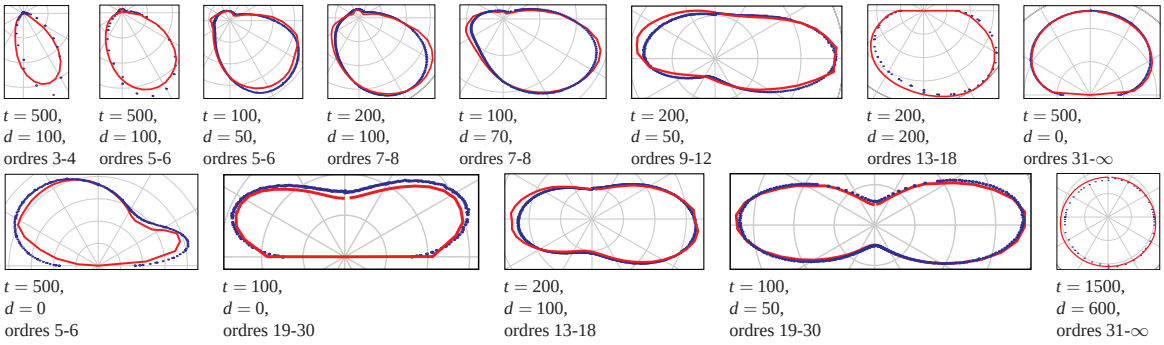


Figure 8: *Quelques BSDF résultant de notre étude du transport de la lumière. T est tracée en fonction de ϕ_V (ψ_V est fixé à $0°$) pour différentes valeurs d'entrées ($\phi_L = 15°$).* En bleu : *simulations de Monte-Carlo.* En rouge : *notre ajustement.*

considérons un chemin le plus probable et son étalement associé par groupe d'ordres de dispersion. Plus spécifiquement, nous considérons une *aire collectrice*, qui est la portion de surface à travers laquelle entre 95% de la lumière parvenant au pixel rendu dans la direction de vue (voir figure 7(b)). Cette aire collectrice est définie par son centre $\mathbf{c}$ sur la surface illuminée du nuage et par sa taille σ. Pour un pixel donné, et pour chaque groupe d'ordres de dispersion, nous cherchons ce collecteur et calculons le transport lumineux correspondant.

Pour trouver ce collecteur sur la surface illuminée, nous utilisons un algorithme qui, itérativement, associe le collecteur le plus probable à la surface du nuage. Cet algorithme est décrit en section 6.1.

Par définition, le rôle de la surface du nuage en dehors du collecteur est négligeable. Nous approchons donc localement la forme du nuage par une dalle alignée sur le collecteur afin de simplifier le calcul du transport de la lumière (voir figure 7(b)).

Calculer le transport anisotrope de la lumière même pour une forme simple comme une dalle est cependant encore très complexe [Cha60]. Pour accélérer le calcul du transport de la lumière, nous caractérisons le transfert radiatif dans une dalle à travers une *fonction de transport canonique*. Ceci est décrit en section 5. Nous obtenons cette fonction en étudiant de nombreuses simulations de Monte-Carlo pour différents paramètres de dalle (voir annexe A).

Nous implémentons notre approche de rendu complète sur le GPU, en utilisant des *depth maps* pour représenter la surface du nuage. Ceci est décrit en section 7. En section 8 nous introduisons notre enrichissement de la forme du nuage et en particulier de sa silhouette dans une Hypertexture. Le rassemblement de toutes ces étapes à l'intérieur d'un shader est synthétisé en section 8.1. Nous présentons nos résultats et performances en section 9.

5. Caractérisation du transport lumineux canonique

5.1. Simulation

Dans cette section nous décrivons notre mise en place expérimentale dans le cas canonique d'une dalle d'épaisseur t (voir figure 7(a)). Pour chaque groupe d'ordres de dispersion, nous calculons le transport lumineux T (*i.e.*, la proportion d'énergie lumineuse transmise), le centre du collecteur $\mathbf{c}$ et sa taille σ.

Ces valeurs $\langle T, \mathbf{c}, \sigma \rangle$ sont calculées en fonction de 5 paramètres d'entrées : l'épaisseur de la dalle t, la profondeur du point de vue dans la dalle d, les angles de vue (ϕ_V, ψ_V) et l'angle incident d'illumination ϕ_L, dans le repère de référence figure 7(a). Nous définissons $\mu_V = \cos(\phi_V)$, $\mu_L = \cos(\phi_L)$, $\vec{\omega}_V$ = direction de vue, $\vec{\omega}_L$ = direction d'illumination, $\cos\theta = \vec{\omega}_V \cdot \vec{\omega}_L$, $\mathbf{p} = (0, -d, 0)$ = point de vue. Étant donné que nous voulons caractériser le transport de la lumière vers un point $\mathbf{p}$ quelconque, nous prenons des valeurs de la profondeur du point de vue d à l'intérieur de la dalle ($0 \leq d \leq t$). Les points de vue hors de la dalle correspondent à $d = 0$ ou $d = t$.

Nous lançons des simulations de Monte-Carlo des chemins lumineux pour différentes valeurs de ces 5 paramètres (voir annexe A pour plus de détails). Nous stockons les résultats $\langle T, \mathbf{c}, \sigma \rangle(\phi_V, \psi_V, \phi_L, d, t)$ dans une table 5D pour chaque groupe d'ordres de dispersion. Nous appelons $\langle T, \mathbf{c}, \sigma \rangle(\phi_V, \psi_V, \phi_L, d, t)$ la *fonction de transport canonique*.

5.2. Compression des résultats

Ces tables 5D (une par groupe d'ordres de dispersion) encodent la *fonction de transport canonique*, c'est-à-dire le comportement macroscopique de la lumière dans une dalle de nuage en fonction des paramètres d'entrée $(\phi_V, \psi_V, \phi_L, d, t)$. Cet ensemble de tables 5D ne peut pas être stocké en mémoire GPU. Nous les compressons en approximant les résultats calculés par des fonctions empiriques. L'annexe A décrit notre schéma de simulation et de fitting.

• Pour le transport lumineux T, cet ajustement donne

$$T = P \cdot A \cdot X \cdot \mu_L \frac{\log(d+D)}{\log(B+D)} e^{-\frac{(d-B)^2}{2C^2}}$$

avec $A = \mathbf{A}(t,\mu_V)$, $B = \mathbf{B_1}(t,\mu_V) - \mathbf{B_2}(t,\mu_L)$, $C = \mathbf{C}(t,\mu_V)$, $D = \mathbf{D}(t,\mu_V)$, $X = \mathbf{X}(t,\mu_L)$, $P = \mathbf{P}(\theta)$, *i.e.*, une table 5D pour T est réduite en une fonction analytique et sept tables 2D. **P** encode l'anisotropie du résultat et vaut 1 si le comportement est isotrope (ordres 31-∞). **X** module le résultat selon l'angle d'illumination incident. **A**, $\mathbf{B_1}$, $\mathbf{B_2}$, **C**, **D** sont les paramètres d'une fonction Gaussienne "déformée" représentant le comportement de la lumière selon la profondeur du point de vue. La figure 8 montre les résultats de cet ajustement.

• Pour le centre du collecteur **c**, notre compression résulte en $\mathbf{c} = (c_x, 0, c_z)$ avec

$$\begin{aligned}
c_x &= A_x \log(1 + E \cdot d) + B_x, \\
c_z &= A_z \log(1 + E \cdot d) + B_z, \\
A_x &= F \sin(\psi_V) \sin(G \cdot \phi_L), \\
B_x &= H \sin(\psi_V) \sin(\phi_L), \\
A_z &= I + J[\cos(\psi_V) \sin(K \cdot \phi_L) + L\phi_L], \\
B_z &= M + N \cos(\psi_V)
\end{aligned}$$

avec $E = \mathbf{E}(\mu_V)$, $F = \mathbf{F}(\mu_V)$, $G = \mathbf{G}(\mu_V)$, $H = \mathbf{H}(\mu_V)$, $I = \mathbf{I}(\mu_V)$, $J = \mathbf{J}(\mu_V)$, $K = \mathbf{K}(\mu_V)$, $L = \mathbf{L}(\mu_V)$, $M = \mathbf{M}(\mu_V)$, $N = \mathbf{N}(\mu_V, \mu_L)$.

• Notre compression de la taille du collecteur σ donne

$$\sigma = \mathbf{O} + \mathbf{Q} \cdot t \cdot \log(1 + \mathbf{R} \cdot d) + \mathbf{S} \cdot \log(1 + \mathbf{T} \cdot t)$$

où **O**, **Q**, **R**, **S**, **T** sont des constantes pour un ordre de dispersion donné.

6. Estimation du transport lumineux dans les nuages

6.1. Dispersion multiple

Pour un pixel donné du nuage à rendre (qui correspond à une position **p** dans le nuage), pour chaque groupe d'ordres de dispersion, nous cherchons l'aire collectrice $(\hat{\mathbf{c}}, \hat{\sigma})$, *i.e.*, l'origine des chemins lumineux dispersés sur la surface illuminée qui sont parvenus en **p** dans la direction de vue (voir figure 7(b)). Sachant qu'un nuage est un volume de densités hétérogènes, définir sa surface est difficile. Nous la définissons comme l'iso-surface où la densité ρ vaut une valeur ρ_0 définie par l'utilisateur. Comme expliqué en section 4, nous assumons que le nuage se comporte localement comme une dalle tangente à la surface à la position du collecteur. L'orientation de la surface est ici une notion dépendante de l'échelle. Étant donné que nous sommes intéressés par la surface d'entrée des chemins lumineux *dispersés*, notre orientation locale de surface est obtenue en *filtrant* la surface du nuage selon l'écart-type de l'étalement σ (forme du nuage en pointillés dans les figures 7(b) et 9). Les sections 7 et 8 donnent plus de détails sur ce filtrage.

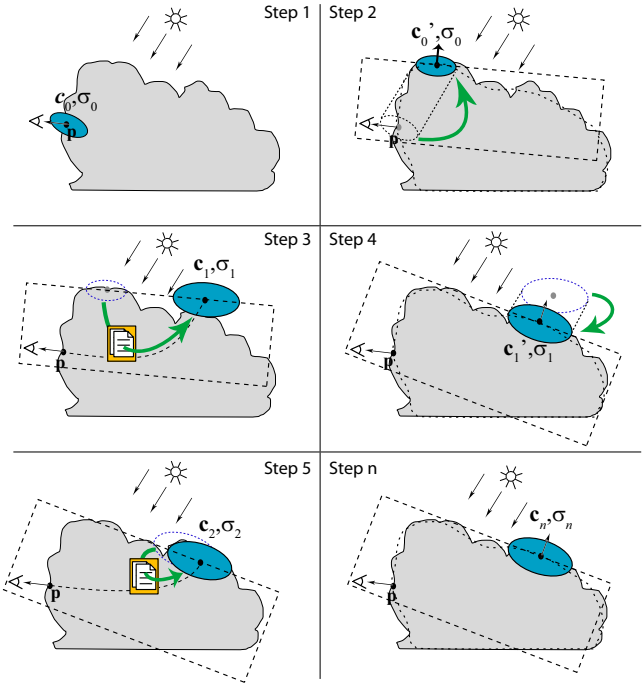


Figure 9: *Résumé de notre algorithme itératif permettant de trouver le collecteur (aire bleue). Forme de nuage solide : surface du nuage non filtrée. Forme de nuage en pointillés : surface filtrée par σ. Rectangle en pointillés : dalle tangente à la surface filtrée en* **c**. *Les symboles oranges représentent l'utilisation de notre fonction de transport canonique.*

Pour trouver la position de ce collecteur, nous itérons comme illustré en figure 9. Nous démarrons à l'étape 1 avec une taille initiale σ_0 et une position initiale $\mathbf{c}_0$ que nous projetons à l'étape 2 dans la direction de la lumière sur la surface du nuage en $\mathbf{c}'_0$. La dalle correspondante est tangente à la surface (filtrée selon σ_0) en $\mathbf{c}'_0$. Les paramètres de vue et d'illumination $(\phi_V, \psi_V, \phi_L, d, t)$ par rapport à cette dalle sont obtenus par simples transformations géométriques. A l'étape 3, la fonction de transport canonique nous donne la position du collecteur $\mathbf{c}_1(\phi_V, \psi_V, \phi_L, d, t)$ et sa taille $\sigma_1(\phi_V, \psi_V, \phi_L, d, t)$ correspondant à une telle configuration, qui est probablement différente de $\mathbf{c}'_0$. Nous projetons $\mathbf{c}_1$ sur la surface du nuage filtrée par σ_1 en étape 4, et itérons jusqu'à convergence, c'est-à-dire $\mathbf{c}_n \simeq \mathbf{c}'_n$. Nous prenons alors $(\hat{\mathbf{c}}, \hat{\sigma}) = (\mathbf{c}_n, \sigma_n)$. Une fois le collecteur trouvé, nous pouvons obtenir le transport lumineux T correspondant (*i.e.*, la quantité d'énergie transpose depuis ce collecteur jusqu'à l'œil) avec les mêmes paramètres d'entrée. Nous effectuons cet algorithme pour chaque groupe d'ordres de dispersion sur le GPU (voir section 7).

Cet algorithme itératif est en fait une méthode du point fixe appliquée sur **c**, *i.e.*, nous cherchons la valeur x qui satisfait $f(x) = x$, où x représente nos paramètres de collecteur $(\mathbf{c}, \sigma)$ et f représente notre fonction de transport canonique.

Étant donné que l'espace de recherche est restreint à la surface illuminée du nuage, cet algorithme ne peut pas diverger. Cependant, comme n'importe quel méthode du point fixe basique, il peut atteindre un état où il boucle d'une valeur à une autre sans converger (*e.g.*, $\mathbf{c}_i \neq \mathbf{c}_{i-1}$ et $\mathbf{c}_i = \mathbf{c}_{i-2}$). Il peut également prendre des pas trop grand et manquer la solution. Pour éviter ces cas, nous limitons la taille de chaque pas ($\mathbf{c}_i$ est contraint tel que $\|\mathbf{c}_i - \mathbf{c}_{i-1}\| < \sigma_{i-1}$). Nous démarrons l'algorithme avec une position initiale $\mathbf{c}_0 = \mathbf{p}$. Nous choisissons une taille initiale σ_0 grande. En effet, si le collecteur initial comprend la totalité du nuage, le résultat projeté à l'étape 2 sera une approximation au premier ordre de la surface illuminée totale, ce qui est un bon point de départ pour notre algorithme en terme de vitesse de convergence. Les étapes suivantes vont ensuite "raffiner" la surface illuminée à la position du collecteur le plus probable.

6.2. Single scattering

Pour le premier ordre de dispersion, qui constitue un cas dégénéré pour notre algorithme basé sur le collecteur, nous intégrons analytiquement la dispersion de Mie le long de la direction de vue. Ceci nous permet de prendre en compte les densités hétérogènes et les détails fin le long de la direction de vue.

Cette intégration est fait en considérant des segments successifs comme illustré en figure 10 (*i.e.*, des échantillons espacés exponentiellement sont lus le long de la direction de vue). Pour un segment i, la formule de la dispersion primaire donne

$$\begin{aligned} T_i &= \mathcal{M}ie(\theta) \int_{x_i}^{x_{i+1}} \kappa e^{-\kappa(x+l(x))} dx \\ &= \mathcal{M}ie(\theta)(e^{-\kappa(x_{i+1}+l_{i+1})} - e^{-\kappa(x_i+l_i)}) \end{aligned} \quad (1)$$

où $\mathcal{M}ie$ est la fonction de phase de Mie et κ le coefficient d'extinction décrit par l'équation 4, annexe A. À noter que puisque la fonction de phase de Mie est dépendante de la longueur d'onde, nous encodons et utilisons $\mathcal{M}ie$ comme une fonction RVB. En revanche, pour la dispersion multiple, cette dépendance est atténuée et il est suffisant de représenter T par un scalaire.

6.3. Opacité

L'opacité α est calculée en intégrant l'extinction le long de la direction de vue. Ceci résulte en $\alpha = \int_0^\infty e^{-\kappa x} dx$. Puisque le coefficient d'extinction κ varie dans l'espace, nous discrétisons dette intégrale en segments le long de la direction de vue, comme pour le *single scattering*. Pour chaque segment i, nous lisons κ dans le volume du nuage et accumulons l'opacité

$$\alpha_i = \int_{x_i}^{x_{i+1}} e^{-\kappa x} dx = \frac{e^{-\kappa x_{i+1}} - e^{-\kappa x_i}}{\kappa}. \quad (2)$$

Ceci est une procédure standard de *ray marching*.

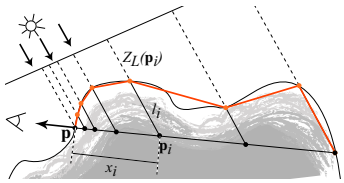


Figure 10: *Intégration linéaire par morceaux des chemins de la dispersion primaire.*

7. Implémentation sur GPU

Au moment du rendu, le maillage du nuage est rendu en utilisant un *fragment shader* qui calcule le transport lumineux depuis la surface illuminée comme expliqué en section 6. Étant donné que ce processus est lourd, nous tirons parti des capacités du GPU pour le rendre efficace, en particulier pour le calcul des distances à la surface filtrée, que nous détaillons dans cette section.

Les 22 fonctions intermédiaires **A** à **X** décrites en section 5.2 et nécessaires pour calculer le la fonction de transport canonique sont discrétisées et stockées sous forme de textures, ainsi que la fonction de phase $\mathcal{M}ie$ utilisée dans la dispersion primaire. En tout, cela donne quelques textures occupant moins de 2Mo en mémoire vidéo.

Conversion de la géométrie en *depth maps*
Pour gérer efficacement le traitement de la surface du nuage (*i.e.*, la filtrer et calculer les distances à cette surface filtrée) sur le GPU, nous nous basons sur des *depth maps*. Nous créons deux *depth maps* $Zmin_L$ et $Zmax_L$ et une *normal map* N_L depuis le point de vue de la lumière. Ceci est effectué à chaque image pour permettre l'animation de la forme du nuage et de la lumière.

Une façon pratique d'approximer le filtrage de la surface par un noyau de taille σ est de se baser sur le MIP-mapping des valeurs de profondeur Z dans la *depth map*. Pour calculer le MIP-mapping correctement (*i.e.*, sans prendre en compte les pixels de la *depth map* où il n'y a pas de nuage), nous ajoutons un canal alpha A et suivons la même procédure que pour les textures à alpha prémultiplié [DS03]. En fait cette prémultiplication est faite automatiquement lorsqu'on règle la valeur par défaut de Z à 0. Une fois que la pyramide de MIP-map $(Z^0, ..., Z^n)$ est calculée, une valeur Z filtrée est obtenu par $\frac{Z^i(P)}{A^i(P)}$.

Le calcul des distances d (section 6.1) et l_i (équation 1) à la surface filtrée dans la direction de la lumière consiste à simplement accéder à la *depth map* au niveau de MIP-map associé. Pour calculer l'épaisseur t, nous soustrayons $Zmax_L - Zmin_L$. L'orientation de la surface filtrée est obtenue en lisant dans la *normal map* MIP-mappée N_L. Grâce à ces représentations, tous les paramètres d'entrée $(\phi_V, \psi_V, \phi_L, d, t)$ peuvent être calculés et l'algorithme complet décrit en section 6 peut être implémenté dans un fragment shader. Pour chaque groupe d'ordres de dispersion, le shader commence par trouver le collecteur $(\hat{\mathbf{c}}, \hat{\sigma})$ avec notre

algorithme décrit en section 6.1, puis calcule l'intensité T associée en utilisant la fonction de transport canonique compressée décrite en section 5.

8. Enrichissement volumique du modèle de nuage

Les approches pour les nuages basées sur les surfaces reposent généralement sur un shader de transparence donnant l'illusion d'hétérogénéités sur la silhouette. Les méthodes basées sur des volumes sont limitées en résolution en raison de besoins en mémoire importants, ce qui en général donne une silhouette manquant de détails (voir figure 5). Notre approche combine le meilleur des deux : nous définissons une Hypertexture volumique dans une couche sous la surface du maillage afin d'avoir des effets 3D à haute résolution sur les bords du nuage en utilisant peu de mémoire (voir figure 6). Ainsi, la représentation volumique ajoute toutes les fréquences de détails du nuage qui ne sont pas représentées par la géométrie.

Cette Hypertexture dépend d'un *distance field* $D(\mathbf{p})$ à la surface (qui vaut 0 sur la surface et augmente à l'intérieur du nuage). Les détails ajoutés par ce volume sont définis procéduralement en modulant la densité N_0 (équation 3) et le coefficient d'extinction κ (équation 4) avec $\rho(\mathbf{p}) = S(D(\mathbf{p}) + noise(\mathbf{p}))$ où S est une fonction sigmoïde et *noise* un bruit de Perlin scalaire 3D [Per85]. Étant donné le coût de calcul d'un *distance field*, $D(\mathbf{p})$ est calculé et stocké dans une texture volumétrique. $noise()$ est évalué à la volée dans le shader. Nous nous basons sur [ano08] pour calculer et voxéliser rapidement le champs de distance à haute résolution en utilisant uniquement la mémoire minimum nécessaire.

Cet enrichissement volumique est utilisé en trois endroits du shader. Lorsque nous calculons l'opacité du nuage (section 6.3), nous effectuons un *ray marching* sur le GPU [ano08] à travers cette couche pour intégrer la densité du nuage. Lorsque nous calculons la dispersion simple (section 6.2), le coefficient d'extinction κ (équation 4) est également modulé pour chaque segment par $\rho(\mathbf{p})$. Enfin, pour prendre en compte ces détails dans le calcul de la dispersion multiple, (section 6.1), la *depth map* $Zmin_L$ est modulée : nous stockons dans $Zmin_L$ la profondeur du premier voxel ayant $\rho > \rho_0$.

À noter que pour la plupart des pixels hormis ceux sur la silhouette et sur les parties fines, la forte opacité arrêtera le rayon peu après le point d'entrée, la longueur moyenne du *ray marching* (et donc son coût) est donc limitée.

8.1. Rendu final

Les précalculs pour l'image courante sont limités à la reconstruction des *depth maps* et de la *normal map*. Ceci nous permet de changer aussi bien les conditions de vue et d'illumination que la forme du nuage. En cas d'animation du nuage, nous utilisons l'algorithme décrit dans [ano08] pour recalculer en temps-réel le champ de distance utilisé par notre bruit procédural.

Le terrain et les objets opaques sont dessinés en premier. Ensuite le maillage du nuage est dessiné par *deferred shading* en utilisant notre pixel shader. Ce pixel shader :

- trouve itérativement les collecteurs $(\hat{\mathbf{c}}, \hat{\sigma})$ pour chacun des 8 groupes d'ordres de dispersion grâce à l'algorithme décrit en section 6.1;
- calcule le transport lumineux T associé à chaque groupe comme expliqué en section 5, et multiplie chacun par les conditions d'illumination (intensité, couleur, visibilité);
- calcule la dispersion primaire (section 6.2);
- calcule l'opacité (section 6.3).

Nous prenons en compte l'illumination de l'environnement de la même façon que l'illumination du soleil. Une source de lumière de couleur bleue est ajoutée au-dessus du nuage et une autre de couleur marron en dessous. Nous utilisons notre algorithme de dispersion multiple avec ces deux sources additionnelles. Les scènes extérieures requièrent la prise en compte de l'épaisseur atmosphérique (*aerial perspective*). Nous utilisons le modèle de [HP03]. Le rendu d'objets très lumineux tels que les nuages demande également une gestion du *tone mapping*. Nous utilisons une version simplifiée et non floue de [GWWH03].

À noter que notre représentation nous permet implicitement de prendre en compte les points de vue à l'intérieur des nuages, puisque notre fonction de transport canonique tient compte de ce cas.

9. Résultats

Nos tests ont été conduits sur un Pentium 4 à 1.86 GHz avec une carte graphique nVidia 8800 GTS, à une résolution de 800×600.

Nous avons testé notre méthode sur différentes formes de nuages: une dalle de nuage, une couche de stratocumulus animée sans bruit procédural, et un cumulus avec bruit procédural. La dalle nous permet de tester notre algorithme et valider directement ses résultats par rapport à la fonction de transport canonique. Elle nous permet également de voir des effets tels que la réflectance anisotrope, la transmittance anisotrope dans les dalles fines et isotrope dans les dalles épaisses. La couche de stratocumulus nous permet de tester notre méthode sur une forme proche de la dalle canonique et de valider la gestion des animations. Elle est composée de 130 000 triangles et la vitesse est de 10 images par seconde. Nous pouvons voir sur cet exemple tous les effets recherchés caractéristiques des nuages : la dispersion anisotrope, la diffusion, la rétro-dispersion, la gloire, les arcs blancs. Le modèle de cumulus nous permet de tester notre algorithme sur une forme quelconque. Il est composé de 5 000 triangles et d'une Hypertexture de résolution 512^3. Nous obtenons une

vitesse de 2 images par secondes sur ce modèle. La majeure partie du temps est passée à évaluer le bruit à la volée dans l'Hypertexture. Sans ce bruit, le rendu monte à 10 images par secondes. Ce modèle présente également tous les effets recherchés, et possède les détails nécessaires au réalisme. 10 itérations de notre algorithme de recherche de collecteur sont suffisantes dans tous les cas pour converger. La figure 11 montre les résultats de notre méthode sur des cas variés.

10. Discussion et Perspectives

Nous abordons les limites des autres approches temps-réel [HL01, REK*04, HAP05] en traitant précisément les chemins lumineux de tous les ordres sur un modèle de nuage plus détaillé. Nous utilisons la fonction de phase de Mie pour les gouttelettes d'eau, qui nous donne la bonne anisotropie et nous permet de reproduire les effets visuels tels que la gloire et les arcs blancs. Au contraire de [SSEH03], nous tenons compte des détails procéduraux dans le calcul du transport lumineux. Bien que des détails similaires peuvent être obtenus par les autres techniques temps-réel [HL01, REK*04, HAP05] en augmentant la résolution volumique de leurs modèles (au détriment de la vitesse et dans les limites de la mémoire disponible), notre méthode apporte des effets qui ne sont pas pris en compte par ces approches tels que la rétro-dispersion et la diffusion. Nous permettons l'animation du point de vue, de la lumière et de la forme du nuage.

Comme pour les autres méthodes utilisant une *depth map* pour résoudre des problèmes de dispersion [DS03], cette représentation nous pose quelques problèmes vis-à-vis de la résolution et des formes non convexes. La précision de cette représentation est limitée par la résolution de la *depth map*. En conséquence, le passage à l'échelle de la méthode requièrerait probablement des techniques de *depth maps* alternatives [SD02, Arv07]. Dans le cas de formes non convexes, le transport lumineux est légèrement sous-estimé dans les parties ombrées (voir figure 12(a)) : l'algorithme assume qu'il y a de la matière entre les surfaces à l'ombre et la surface éclairée, ce qui est incorrect. Il en résulte que la lumière est considérée plus atténuée que ce qu'il faudrait. En pratique, cette erreur survient dans les parties ombragées, où l'illumination par l'environnement (*i.e.*, par le ciel et le sol) est prédominante. Comme indiqué dans [DS03], qui souffrent des même problèmes, ce point peut être résolu en utilisant une technique de *depth peeling* pour traiter les formes non convexes comme une collection de formes convexes. L'utilisation de *deep shadow maps* [LV00] aiderait également à résoudre ce problème et augmenterait la précision de notre calcul de dispersion primaire.

Une limitation de notre approche est la supposition que la lumière arrivant à l'utilisateur passe par seulement un seul collecteur connexe (*i.e.*, un seul chemin le plus probable) par groupe d'ordres de dispersion. Comme indiqué sur la figure 12(b), cela n'est pas toujours le cas. Dans ces configurations, notre algorithme ne prend en compte qu'un collecteur, et sous-estime donc la quantité de lumière transmise. Cependant, puisque nous cherchons plusieurs collecteurs par pixel (un pour chacun des 8 groupes d'ordres de dispersion), cette erreur ne concerne qu'une fraction de la couleur d'un pixel. Une solution pourrait être de chercher plusieurs collecteurs par groupe d'ordres de dispersion, avec différentes valeurs initiales.

Rechercher un collecteur $\hat{\mathbf{c}}$ correspond à chercher le point d'entrée $\hat{\mathbf{c}}$ sur la surface illuminée d'un maximum local du transport de la lumière T. Notre algorithme itératif correspond à une méthode du point fixe : nous cherchons le point fixe $f(\hat{\mathbf{c}}) = \hat{\mathbf{c}}$ où f représente notre fonction de transport canonique. Dans le futur, de meilleures techniques du milieu de l'optimisation pourraient être utilisées pour garantir une convergence plus rapide et pour traiter le cas de plusieurs maxima de T.

Notre calcul du transport lumineux sur GPU utilisant notre algorithme de recherche de collecteur (*i.e.*, la principale contribution de cet article) est suffisamment rapide pour des applications temps-réel (10 images par seconde). La rapidité de cette méthode peut encore être augmentée. De plus, le calcul du bruit procédural et le *ray marching* sur GPU dans notre implémentation sont des versions non optimisées de [ano08] et prennent un temps de calcul conséquent (80% du coût de rendu).

Dans le futur, nous aimerions prendre en compte les effets lumineux de plus haut niveau tels que les inter-réflexions entre les nuages et entre les lobes des nuages. Aussi, la méthode décrite dans [BNL06] peut être appliquée à notre modèle pour calculer les inter-réflexions entre les nuages et le sol, qui ont une certaine importance visuelle.

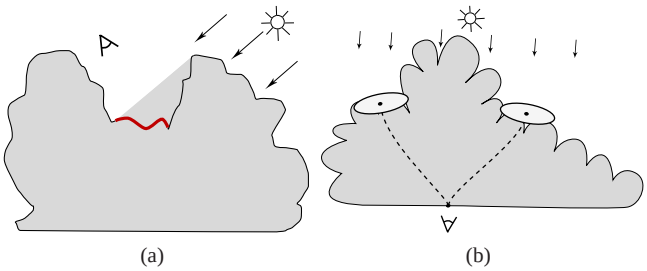


Figure 12: ***(a)*** *Limites de l'utilisations de depth maps. L'illuminance de la surface schématisée en rouge est légèrement sous-estimée. Cependant, l'erreur est de faible importance visuelle.* ***(b)*** *Limites de la recherche d'un seul collecteur par groupe d'ordres de dispersion. Dans cet exemple, deux collecteurs d'importances équivalentes existent, mais notre méthode n'en cherchera qu'un.*

11. Conclusion

Nous avons présenté une étude du transport lumineux débouchant sur une nouvelle formulation du comportement macroscopique de la lumière dans les milieux participants adaptée à la synthèse d'image. Nous avons présenté une nouvelle façon de calculer le transfert radiatif à travers un vol-

ume homogène de milieu participant avec une méthode optionnelle dédiée au traitement des bordures inhomogènes. Contrairement aux approches précédentes, notre méthode n'a besoin que de parcourir les bordures du milieu et non pas le volume entier. Nous avons démontré la validité de cette technique sur des nuages détaillés de type cumulus. Comme on peut le voir dans nos résultats, nous reproduisons correctement les effets visuels tels que la rétro-dispersion, la dispersion anisotrope multiple, la gloire, etc.

12. Remerciements

De très gros mercis vont à Guillaume Piolat pour les jours et les nuits passés sur le code et à Laurence Boissieux et Florent Moulin pour les week-ends passés à modéliser les nuages. Merci également à Sébastien Barbier et Paul Kry pour les relectures. Merci enfin à Frédéric Saint-Marcel et Eric Ragusi pour l'utilisation et la maintenance du cluster et de la base de données. Le LLNL a également mis ses clusters à disposition. Les fonctions de Mie ont été générées grâce au précieux outil MiePlot de Philip Laven.

Appendix A: Mesurer le transport lumineux dans une dalle de nuage

Comme expliqué en section 2, la fonction de phase appropriée pour les nuages est la fonction de Mie. C'est une fonction complexe et oscillante dépendant de la taille de la gouttelette et de la longueur d'onde de la lumière. Comme [BNL06], nous pré-intégrons l'effet de la DSD sur la fonction de phase et utilisons l'approximation *Mie modifiée* dans laquelle le pic en avant (5°) est supprimé de la fonction de phase et traduit comme une densité diminuée de 50% [Len85]. Cette approche a été montrée comme donnant des résultats équivalent tout en nécessitant de traiter deux fois moins d'événements de dispersion.

Pour la DSD, nous utilisons la distribution Gamma modifiée [Lev58] pour les gouttelettes de nuages définie par

$$N(r) = \frac{\rho N_0}{\Gamma(\gamma) r_n} \left(\frac{r}{r_n}\right)^{\gamma-1} e^{-\frac{r}{r_n}}. \qquad (3)$$

Cette fonction décrit la densité $N(r)$ des gouttelettes de rayon r, avec r_n le rayon caractéristique de la distribution, γ la largeur de la distribution, et N_0 la densité totale de gouttelettes. Γ est la fonction Gamma. ρ est un facteur de modulation de la densité totale apporté par notre enrichissement procédural décrit en section 8. Une distribution de gouttelettes pour nuage est ainsi décrite par les seuls trois paramètres N_0, r_n et γ. En terme de propriétés optiques, le rayon efficace correspondant à cette distribution est $r_e = (\gamma+2) r_n$. Nous prenons comme paramètres typiques pour un cumulus $r_e = 6\ \mu$m, $\gamma = 2$, $N_0 = 4.10^8 \mathrm{m}^{-3}$.

Le coefficient d'extinction κ est défini par

$$\kappa = \rho N_0 \pi r_e^2. \qquad (4)$$

Il donne la fonction d'extinction $e^{-\kappa x}$ qui est la probabilité de traverser le nuage le long d'un segment de longueur x sans toucher une gouttelette. Il est utilisé dans toutes les équations de la dispersion, c'est-à-dire dans l'opacité (équation 2), le *single scattering* (équation 1) et la dispersion multiple (équation 5).

L'équation de la dispersion multiple a été présentée maintes fois dans la littérature [Cha60, Kv84]. Elle peut être écrite sous la forme

$$-\frac{\vec{\omega}}{\kappa} \cdot \frac{dT(\mathbf{p},\vec{\omega})}{d\mathbf{p}} = T(\mathbf{p},\vec{\omega}) + \frac{1}{4\pi} \int_{\vec{\omega}'} P(\vec{\omega}\cdot\vec{\omega}') T(\mathbf{p},\vec{\omega}') d\vec{\omega}' \qquad (5)$$

pour l'intensité T parvenant à un point $\mathbf{p}$ dans la direction $\vec{\omega}$, avec $P(\theta)$ la fonction de phase du milieu (ici, $\mathcal{M}ie$). Cette équation intégro-différentielle peut être résolue numériquement par une simulation de Monte-Carlo. La contribution de chaque ordre dans l'intensité T, ainsi que les informations du collecteur $(\mathbf{c},\sigma)$ peuvent être aisément récupérées pendant cette intégration. Pour notre caractérisation du transport lumineux (section 5), nous avons calculé la solution de l'équation 5 dans une dalle de nuage pour des valeurs variables de $(\phi_V, \psi_V, \phi_L, d, t)$. Nous avons stocké la contribution T de chaque ordre de dispersion dans une base de données, ainsi que les données de collecteurs $(\mathbf{c},\sigma)$. Ces calculs ont été fait avec un intervalle de confiance de 5% au niveau 95%. Nous les avons lancées pour 10 millions de valeurs différentes de $(\phi_V, \psi_V, \phi_L, d, t)$, résultant en une taille de base de donnée brute d'environ 25 Go. Ces calculs ont pris plusieurs semaines sur un cluster de 100 nœuds d'Itanium-2 dual-core à 900MHz. La base de données et les résultats de fitting seront disponibles en ligne.

L'analyse de ces résultats (*i.e.*, trouver des fonctions de plus basse dimension reproduisant les résultats) a été faite empiriquement en traçant $\langle T, \mathbf{c}, \sigma \rangle$ selon les paramètres d'entrées et en recherchant des comportements remarquables. Elle a été inspirée par des approches précédentes telles que les fonctions X et Y de Chandrasekhar. L'ajustement lui-même a été fait par optimisation standard non linéaire au sens des moindres carrés avec MATLAB.

References

[ano08] ANONYMOUS: Real-time rendering of large detailled volumes. In *submitted to the ACM SIGGRAPH Symposium on Interactive 3D graphics and games (I3D)* (2008).

[Arv07] ARVO J.: Alias-free shadow maps using graphics hardware. *journal of graphics tools 12*, 1 (2007), 47–59.

[Bli82] BLINN J. F.: Light reflection functions for simulation of clouds and dusty surfaces. In *SIGGRAPH'82* (1982), pp. 21–29.

[BNL06] BOUTHORS A., NEYRET F., LEFEBVRE S.: Real-time realistic illumination and shading of stratiform

clouds. In *Eurographics Workshop on Natural Phenomena* (sep 2006).

[Cha60] CHANDRASEKHAR S.: *Radiative transfer*. New York: Dover, 1960, 1960.

[DKY*00] DOBASHI Y., KANEDA K., YAMASHITA H., OKITA T., NISHITA T.: A simple, efficient method for realistic animation of clouds. In *SIGGRAPH'00* (July 2000), pp. 19–28.

[DS03] DACHSBACHER C., STAMMINGER M.: Translucent shadow maps. In *Eurographics Workshop on Rendering (EGWR)* (2003), pp. 197–201.

[Ebe97] EBERT D. S.: A cloud is born. In *SIGGRAPH'97* (1997), p. 245.

[ES00] ELINAS P., STÜRZLINGER W.: Real-time rendering of 3D clouds. *J. Graph. Tools 5*, 4 (2000), 33–45.

[Gar85] GARDNER G. Y.: Visual simulation of clouds. In *SIGGRAPH'85* (1985), ACM Press, pp. 297–304.

[GWWH03] GOODNIGHT N., WANG R., WOOLLEY C., HUMPHREYS G.: Interactive time-dependent tone mapping using programmable graphics hardware. In *Eurographics Workshop on Rendering (EGRW)* (2003), pp. 26–37.

[HAP05] HEGEMAN K., ASHIKHMIN M., PREMOŽE S.: A lighting model for general participating media. In *ACM SIGGRAPH Symposium on Interactive 3D graphics and games (I3D)* (2005), pp. 117–124.

[HL01] HARRIS M. J., LASTRA A.: Real-time cloud rendering. *Computer Graphics Forum 20*, 3 (2001), 76–84.

[HP03] HOFFMAN N., PREETHAM A. J.: *in Graphics programming methods*. Charles River Media, Inc., 2003, ch. Real-time light-atmosphere interactions for outdoor scenes, pp. 337–352.

[JMLH01] JENSEN H. W., MARSCHNER S. R., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. In *SIGGRAPH'01* (2001), pp. 511–518.

[KE86] KRIM M. S. A., EL WAKIL S. A.: Angular distribution of radiation in an isotropically scattering slab. *Astrophysics and Space Science 121* (Apr. 1986), 137–145.

[Kv84] KAJIYA J. T., VON HERZEN B. P.: Ray tracing volume densities. In *SIGGRAPH'84* (1984), pp. 165–174.

[Len85] LENOBLE J.: *Radiative transfer in scattering and absorbing atmospheres: standard computational procedures*. A. Deepak Publishing, 1985.

[Lev58] LEVIN L. M.: Functions to represent drop size distribution in clouds. the optical density of clouds. *Izv. Akad. Nauk. SSSR, Ser. Geofiz. 10* (1958), 198–702.

[LV00] LOKOVIC T., VEACH E.: Deep shadow maps. In *SIGGRAPH'00* (2000).

[Max94] MAX N. L.: Efficient light propagation for multiple anisotropic volume scattering. In *Eurographics Workshop on Rendering (EGWR)* (1994), pp. 87–104.

[MMKW97] MAX N. L., MOBLEY D., KEATING B., WU E.: Plane-parallel radiance transport for global illumination in vegetation. In *Eurographics Workshop on Rendering (EGWR)* (1997), pp. 239–250.

[Mob89] MOBLEY C.: A numerical model for the computation of radiance distributions in natural waters with wind-roughened surfaces. *Limnol. Oceanogr. 34* (1989), 1473–1483.

[Ney03] NEYRET F.: Advected textures. In *ACM SIGGRAPH/EG Symposium on Computer Animation (SCA)* (july 2003).

[NND96] NISHITA T., NAKAMAE E., DOBASHI Y.: Display of clouds taking into account multiple anisotropic scattering and sky light. In *SIGGRAPH'96* (Aug. 1996), pp. 379–386.

[PAS03] PREMOŽE S., ASHIKHMIN M., SHIRLEY P.: Path integration for light transport in volumes. In *Eurographics Symposium on Rendering (EGSR)* (2003), pp. 52–63.

[PAT*04] PREMOZE S., ASHIKHMIN M., TESSENDORF J., RAMAMOORTHI R., NAYAR S.: Practical rendering of multiple scattering effects in participating media. In *Eurographics Symposium on Rendering (EGSR)* (June 2004), pp. 363–374.

[Per85] PERLIN K.: An image synthesizer. In *SIGGRAPH'85* (July 1985), pp. 287–296.

[PH89] PERLIN K., HOFFERT E. M.: Hypertexture. In *SIGGRAPH'89* (July 1989), pp. 253–262.

[REK*04] RILEY K., EBERT D. S., KRAUS M., TESSENDORF J., HANSEN C.: Efficient rendering of atmospheric phenomena. In *Eurographics Symposium on Rendering (EGSR)* (June 2004), pp. 375–386.

[SD02] STAMMINGER M., DRETTAKIS G.: Perspective shadow maps. In *SIGGRAPH'02* (2002).

[SSEH03] SCHPOK J., SIMONS J., EBERT D. S., HANSEN C.: A real-time cloud modeling, rendering, and animation system. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2003), pp. 160–166.

[Sta95] STAM J.: Multiple Scattering as a Diffusion Process. In *Eurographics Workshop on Rendering (EGWR)* (1995), pp. 41–50.

[TB02] TREMBILSKI A., BROSSLER A.: Surface-based efficient cloud visualisation for animation applications. In *WSCG* (2002), pp. 453–460.

Figure 11: *Quelques résultats de notre méthode.*