



HAL
open science

Efficient models for representing sub-pixel appearances

Guillaume Loubet

► **To cite this version:**

Guillaume Loubet. Efficient models for representing sub-pixel appearances. Graphics [cs.GR]. Université Grenoble Alpes, 2018. English. NNT : 2018GREAM033 . tel-01849666v2

HAL Id: tel-01849666

<https://theses.hal.science/tel-01849666v2>

Submitted on 7 Nov 2018

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.

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : Mathématiques et Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Guillaume LOUBET

Thèse dirigée par **Fabrice NEYRET**, CNRS

préparée au sein du **Laboratoire Jean Kuntzmann** dans l'**École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Représentations efficaces de l'apparence sous-pixel

Efficient models for representing sub-pixel appearances

Thèse soutenue publiquement le **25 juin 2018**,
devant le jury composé de :

Monsieur FABRICE NEYRET

DIRECTEUR DE RECHERCHE, CNRS DELEGATION ALPES, Directeur
de thèse

Monsieur DANIEL MENEVEAUX

PROFESSEUR, UNIVERSITE DE POITIERS, Président

Monsieur STEVE MARSCHNER

PROFESSEUR, UNIV. CORNELL A D'ITHACA, NEW-YORK - USA,
Rapporteur

Madame EMILIE GUY

CHERCHEUSE, SOCIETE DYNAMIXYZ - RENNES, Examineur

Monsieur WENZEL JAKOB

PROFESSEUR ASSISTANT, ECOLE POLYTECH. FEDERALE
LAUSANNE SUISSE, Examineur



Abstract

We address the problem of rendering extremely complex virtual scenes with large amounts of detail. We focus on high-end off-line rendering algorithms based on path tracing that are intensively used in the special effects and 3D animation industries. The large amounts of detail required for creating believable virtual worlds raise important efficiency problems that paralyze production rendering pipelines and greatly complicate the work of 3D artists. We introduce new algorithms for prefiltering complex 3D assets while preserving their appearance at all scales, in order to reduce loading times, ray intersection costs, shading costs and Monte Carlo noise, without lowering the quality of rendered frames. Our main contributions are a new hybrid LOD approach that combines the benefits of meshes and volumetric representations for prefiltering complex 3D assets, as well as a new approach for prefiltering high-resolution heterogeneous participating media.

L'objectif de cette thèse est le rendu de scènes virtuelles extrêmement détaillées. Nous nous intéressons plus particulièrement aux algorithmes de rendu de haute qualité reposant sur du path tracing qui sont très largement utilisés dans l'industrie des effets spéciaux et pour calculer le rendu de films d'animations. Les grandes quantités de détails nécessaires à la modélisation de mondes virtuels crédibles soulèvent de sérieux problèmes d'efficacité du rendu qui paralysent les studios et compliquent grandement le travail des artistes. Nous introduisons de nouveaux algorithmes pour préfiltrer les objets 3D complexes à des échelles arbitraires, afin de réduire les temps de chargement, les coûts d'intersection des rayons lumineux, le calcul des matériaux et la quantité de bruit dans les images, le tout sans porter atteinte à la qualité du rendu. Nos contributions principales sont une nouvelle approche hybride de niveaux de détail qui combine les avantages des maillages et des représentations volumiques pour le préfiltrage des objets complexes à des échelles arbitraires, ainsi qu'une nouvelle approche de préfiltrage pour le cas des volumes hétérogènes de haute résolution.

Acknowledgements

Le rôle de Fabrice Neyret a largement dépassé celui de directeur de thèse pendant ce doctorat. Sa vision érudite et humaniste de la science comme du reste est un apport de grande valeur qui se mesurerait en pétacrédits si l'école doctorale le prenait en compte. Je conseille à quiconque de prétexter un travail de thèse pour côtoyer Fabrice.

I thank Steve Marschner, Daniel Menevaux, Emilie Guy and Wenzel Jakob for their time and most of all for their interesting questions and remarks during the defense. I feel very lucky to have had such a PhD committee.

I owe a lot to Antoine Bouthors and his knowledge of LODs, and I want to thank Derek Gerstmann and Daniel Lond for helping during my internship at Weta Digital. I also thank Jed Wojtowicz and Mariko Tosti for giving me the unique opportunity to discover Weta Digital and New Zealand.

Patrick Kelly and Sean Palmer have been very welcoming at the Walt Disney Animation Studios and I feel exceptionally fortunate to have joined them during a few months. I thank Brent Burley, Daniel Teece, Yining Karl Li, Joe Longson, David Adler, Ralf Habel, Peter Kutz and Matt Chiang for their help and the interesting discussions about production path tracing. Raquel and Scott have been wonderful hosts and I owe them a better understanding of LA culture as well as some English tips. Alena has been a great cubicle neighbor and a precious friend.

Je remercie tous les permanents de Maverick : Romain pour sa bonne humeur malgré ses courtes nuits de sommeil, Cyril pour son humour unique et ses astuces toujours surprenantes, Georges-Pierre et ses avis contrastés sur la musique, JD qui est parti trop tôt, Laurence pour les sessions chant aux journées Maverick, Nicolas pour sa disponibilité et son expertise de tous les sujets pointus et délicats dont on préfère généralement ne pas s'occuper. Joëlle a toujours une oreille attentive et de bons conseils, j'ai été ravi de l'avoir sous le coude pendant ce doctorat.

Diane Courtiol a été d'une grande aide pour les problèmes administratifs en tout genre et un soutien moral dans les moments difficiles, je l'en remercie grandement.

Un grand merci à Hugo, Léo, Petit Benoît, et en particulier Grand Benoît pour ses conseils et son intarissable espièglerie, ainsi qu'aux jeunes doctorants, Alexandre pour sa tasse, bien sûr, Jérémy le co-bureau parfait, Alban pour toutes ces discussions passionnantes sur la sécurité routière et l'art de l'ire, Vincent pour prendre la relève du support technique et perpétuer la lignée des Fafaiens. Prashant, Pascal, Paul, Jérémy, et la sympathique Beibei contribuèrent aussi à faire de Maverick une équipe conviviale. Merci aux ex-doctorants Cyril, Jonathan, Laurent, et Eric qui a sauvé ma première publication d'un rejet très probable.

Un immense merci à mes parents pour leur aide tout au long du doctorat, ainsi que d'avoir insisté (lourdement, c'est vrai) pour être conviés à ma soutenance. Vous avoir eus à mes côtés était un grand plaisir. Merci à Ben pour son oreille toujours attentive, et à Yo (celui-là, c'est notre petit expert en houmous).

Un merci tout particulier à Gaella et Maylis pour m'avoir prêté tant de fois leurs notes de cours au CPP et à l'Ensimag, et pour m'avoir évité de très nombreuses catastrophes administratives.

Un grand merci à Valéry du conservatoire de Grenoble, qui est indirectement et malgré elle à l'origine

de ce doctorat.

Merci à Marc pour les débats scientifiques, politiques et rythmiques, parfois constructifs, à Jade pour son recul et ses points de vue complémentaires de ceux de mes camarades informaticiens, et à Violaine pour sa bonne humeur contagieuse. Vivre avec vous est l'une des meilleures choses qui me sont arrivées pendant le doctorat, et grâce à vous ces deux dernières années sont constellées de bon souvenirs. Merci à Charlotte pour son soutien inconditionnel et réconfortant. Merci à François pour son recul et les discussions sur LA et sur la recherche scientifique et mycologique. Grâce à Nicolas, je me souviendrai des quelques mois de rédaction comme étant la période la plus heureuse de mon doctorat, ce dont peu de docteurs peuvent se vanter.

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction: efficient rendering with level-of-detail representations | 11 |
| 1.1 | Rendering algorithms | 12 |
| 1.2 | Rendering large amounts of detail | 13 |
| 1.3 | Rendering with level-of-detail representations | 14 |
| 1.4 | Overview of the thesis | 15 |
| 1.5 | Summary of our contributions | 16 |
| 1.6 | Industrial collaborations | 16 |
| 2 | Introduction : rendu efficace avec des niveaux de détail | 19 |
| 2.1 | Les algorithmes de rendu | 20 |
| 2.2 | Le rendu de scènes très détaillées | 21 |
| 2.3 | Le rendu efficace avec des niveaux de détail | 22 |
| 2.4 | Aperçu de la thèse | 24 |
| 2.5 | Résumé de nos contributions | 24 |
| 2.6 | Collaborations industrielles | 25 |
| I | Level-of-detail techniques for efficient rendering | 27 |
| 3 | Previous work on geometric level-of-detail representations | 29 |
| 3.1 | Mesh simplification algorithms in computer graphics | 29 |
| 3.1.1 | Overview of mesh simplification algorithms | 29 |
| 3.1.2 | Coarsening operations for triangle meshes | 31 |
| 3.1.3 | Quad mesh simplification | 31 |
| 3.1.4 | Quantifying and minimizing geometric errors | 32 |
| 3.1.5 | Taking into account reflectance attributes in quality metrics | 34 |
| 3.2 | Other geometric level-of-detail representations | 35 |
| 3.2.1 | Surfels as LODs for global illumination | 35 |
| 3.2.2 | Point set surfaces | 35 |
| 3.2.3 | Representing surfaces with volumetric data | 36 |
| 3.3 | Subdivision, tessellation, displacement | 37 |
| 3.3.1 | Subdivision surfaces | 37 |
| 3.3.2 | Displacement mapping | 38 |
| 3.4 | Summary of the chapter | 39 |
| 3.5 | Résumé du chapitre | 39 |
| 4 | Previous work on surface-space prefiltering | 41 |
| 4.1 | Physical units and notations | 41 |
| 4.1.1 | Directions and solid angles | 41 |
| 4.1.2 | Radiant flux | 42 |
| 4.1.3 | Irradiance | 42 |
| 4.1.4 | Radiance | 42 |
| 4.1.5 | Bidirectional reflectance distribution function (BRDF) | 42 |
| 4.1.6 | The local illumination equation | 43 |
| 4.2 | Prefiltering flat surfaces | 43 |
| 4.2.1 | Problem statement | 43 |
| 4.2.2 | Prefiltering strategies | 45 |
| 4.2.3 | Correlation between BRDF parameters | 48 |
| 4.3 | Prefiltering rough surfaces | 48 |
| 4.3.1 | Problem statement | 49 |
| 4.3.2 | Useful hypotheses | 49 |
| 4.4 | Previous work | 49 |
| 4.4.1 | Mip mapping | 49 |
| 4.4.2 | Normal map prefiltering | 50 |
| 4.4.3 | Prefiltering displaced surfaces | 52 |

| | | |
|-----------|---|-----------|
| 4.5 | Summary of the chapter | 53 |
| 4.6 | Résumé du chapitre | 53 |
| 5 | Previous work on prefiltering aggregate detail | 55 |
| 5.1 | The appearance of aggregate detail | 55 |
| 5.1.1 | The appearance of hair and fur | 55 |
| 5.1.2 | The appearance of fabrics and clothes | 55 |
| 5.1.3 | The appearance of granular materials | 56 |
| 5.1.4 | The appearance of foliage | 57 |
| 5.1.5 | Other complex appearances | 57 |
| 5.1.6 | Relation to participating media. | 57 |
| 5.2 | Stochastic simplification | 57 |
| 5.3 | Background: participating media models in computer graphics | 58 |
| 5.3.1 | The radiative transfer equation | 58 |
| 5.3.2 | Anisotropic RTE and microflake model | 59 |
| 5.4 | Volumetric approximations of aggregate geometry | 61 |
| 5.4.1 | Types of volumetric models | 61 |
| 5.4.2 | Early volumetric approximations | 61 |
| 5.4.3 | Prefiltering with hierarchical volume representations | 62 |
| 5.4.4 | Approximating hair and fur | 62 |
| 5.4.5 | Volumetric models of fabrics and clothes | 63 |
| 5.4.6 | Approximating granular materials | 63 |
| 5.4.7 | Prefiltering foliage with volumes | 64 |
| 5.5 | Summary of the chapter | 64 |
| 5.6 | Résumé du chapitre | 65 |
| II | A new hybrid prefiltering method with meshes and volumes | 67 |
| 6 | Problem analysis: prefiltering production assets at arbitrary scales | 69 |
| 6.1 | Production assets | 69 |
| 6.1.1 | The production pipeline | 69 |
| 6.1.2 | Geometry of base meshes | 70 |
| 6.1.3 | Subdivision surfaces | 70 |
| 6.1.4 | Heterogeneous or inappropriate resolution | 72 |
| 6.1.5 | Displacement | 72 |
| 6.1.6 | Textures and materials | 72 |
| 6.1.7 | Modeling practices | 73 |
| 6.1.8 | Other rendering primitives: curves, volumes | 74 |
| 6.2 | Using LODs in production pipelines | 74 |
| 6.2.1 | Per asset, per shot and per frame precomputation | 74 |
| 6.2.2 | LODs and instancing | 74 |
| 6.2.3 | Useful properties of LOD methods for production pipelines | 75 |
| 6.3 | Types of appearance and efficient representations | 75 |
| 6.3.1 | Distant appearance of textured meshes | 76 |
| 6.3.2 | Identifying important characteristics of appearance | 76 |
| 6.3.3 | Automatic analysis of complex geometry | 77 |
| 6.3.4 | Prefiltering surface-like appearances | 77 |
| 6.3.5 | Prefiltering sub-resolution intricate geometry | 78 |
| 6.4 | Our new framework for generating hybrid LODs | 79 |
| 6.5 | Summary of the chapter | 80 |
| 6.6 | Résumé du chapitre | 80 |
| 7 | A new algorithm for detecting macrosurfaces | 83 |
| 7.1 | Definition of macrosurfaces | 83 |
| 7.1.1 | Ambiguities and requirements | 83 |
| 7.1.2 | Our definition of macrosurfaces | 84 |
| 7.1.3 | Examples | 84 |

| | | |
|-----------|--|------------|
| 7.2 | Implementation | 85 |
| 7.2.1 | Evaluating our macrosurface criterion | 85 |
| 7.2.2 | Per-triangle evaluation and cleaning step | 86 |
| 7.3 | Results and discussion | 87 |
| 7.4 | Summary of the chapter | 87 |
| 7.5 | Résumé du chapitre | 88 |
| 8 | Resolution-based mesh simplification with material prefiltering | 89 |
| 8.1 | Requirements | 90 |
| 8.1.1 | Resolution-based simplification and stopping criteria | 90 |
| 8.1.2 | Seamless transitions | 90 |
| 8.1.3 | Mapping for reflectance prefiltering | 90 |
| 8.1.4 | Preserving manifold surfaces | 92 |
| 8.2 | A new vertex placement strategy for robust mesh simplification with edge collapses | 92 |
| 8.2.1 | Background | 93 |
| 8.2.2 | Our approach | 94 |
| 8.2.3 | Implementation and discussion | 94 |
| 8.3 | A new material prefiltering method for the edge collapse framework | 94 |
| 8.3.1 | Introduction | 95 |
| 8.3.2 | Notations | 95 |
| 8.3.3 | Preserving the mean radiance | 96 |
| 8.3.4 | Our approach | 96 |
| 8.3.5 | Application: using our weights for prefiltering microfacet distributions | 97 |
| 8.3.6 | Results and discussion | 98 |
| 8.4 | Prefiltering macrosurfaces by collapsing face loops | 100 |
| 8.4.1 | Background | 100 |
| 8.4.2 | Face loops | 101 |
| 8.4.3 | Collapsing face loops | 101 |
| 8.4.4 | Mappings with the face loops collapse operations | 101 |
| 8.4.5 | Resolution-based stopping criteria | 102 |
| 8.4.6 | Topology verifications | 103 |
| 8.4.7 | Algorithm | 103 |
| 8.4.8 | Discussion | 103 |
| 8.5 | Summary of the chapter | 103 |
| 8.6 | Résumé du chapitre | 104 |
| 9 | Prefiltering sub-resolution intricate geometry | 105 |
| 9.1 | Background: voxelization algorithms | 105 |
| 9.1.1 | Voxelization | 105 |
| 9.1.2 | Aliasing in voxelization algorithms | 106 |
| 9.2 | A new aliasing-free voxelization algorithm with ray casting | 106 |
| 9.2.1 | Occlusion estimation with ray casting | 107 |
| 9.2.2 | Artifact-free voxelization | 107 |
| 9.3 | Prefiltering sub-resolution intricate geometry using the microflake model | 109 |
| 9.3.1 | Estimating density parameters for the microflake model | 109 |
| 9.3.2 | Estimating other scattering parameters: albedos, microflake distributions | 110 |
| 9.4 | Discussion | 111 |
| 9.5 | Summary of the chapter | 111 |
| 9.6 | Résumé du chapitre | 111 |
| 10 | Results of our hybrid LOD approach | 113 |
| 10.1 | Implementation of our hybrid LOD approach | 113 |
| 10.1.1 | Choice of scattering models | 113 |
| 10.1.2 | Seamless transitions between LODs | 113 |
| 10.1.3 | Other implementation details | 114 |
| 10.2 | Results | 114 |
| 10.3 | Limitations | 115 |
| 10.4 | Discussion | 115 |
| 10.5 | Summary of the chapter | 119 |

| | | |
|------------|---|------------|
| 10.6 | Résumé du chapitre | 119 |
| III | A new microflake model with microscopic shadowing for volume downsampling | 121 |
| 11 | Problem analysis: LODs of voxel-based heterogeneous participating media | 123 |
| 11.1 | Introduction | 123 |
| 11.1.1 | High-resolution heterogeneous participating media | 123 |
| 11.1.2 | Appearance-preserving volume downsampling | 123 |
| 11.1.3 | Evaluating the quality of volume LODs | 124 |
| 11.2 | Case study | 125 |
| 11.2.1 | Examples where linear downsampling is accurate | 125 |
| 11.2.2 | Examples with correlations | 127 |
| 11.2.3 | Incorrect opacity with linearly prefiltered density | 127 |
| 11.2.4 | Examples with incorrect effective albedo and effective phase function | 128 |
| 11.2.5 | Examples of view-dependent appearances | 129 |
| 11.3 | Previous work on volume appearance matching | 129 |
| 11.3.1 | The lack of analytical models for effective albedos and phase functions | 129 |
| 11.3.2 | Previous work on the relationships between scattering parameters and the appearance of media | 131 |
| 11.3.3 | Downsampling scattering parameters [ZWDR16] | 131 |
| 11.4 | Summary of the chapter | 133 |
| 11.5 | Résumé du chapitre | 134 |
| 12 | A new microflake model with microscopic self-shadowing | 137 |
| 12.1 | The microscopic self-shadowing model | 138 |
| 12.1.1 | Notations | 138 |
| 12.1.2 | Microscopic self-shadowing function (A) | 138 |
| 12.1.3 | Attenuation coefficient (σ_t) | 139 |
| 12.1.4 | Single scattering coefficient (σ_{ss}) | 139 |
| 12.1.5 | The single scattering phase function (f_{ss}) | 139 |
| 12.1.6 | Microscopic multiple scattering | 139 |
| 12.2 | Simplified model with isotropic self-shadowing | 140 |
| 12.3 | Summary of the chapter | 140 |
| 12.4 | Résumé du chapitre | 141 |
| 13 | Implementing the microscopic self-shadowing model | 143 |
| 13.1 | Main results | 143 |
| 13.1.1 | Choosing the self-shadowing function A | 143 |
| 13.1.2 | Using trigonometric lobes for the microflake normal distribution | 144 |
| 13.1.3 | Attenuation coefficients for \cos^{2n} and \sin^{2n} lobes | 144 |
| 13.1.4 | Single scattering coefficients | 144 |
| 13.1.5 | Evaluating and sampling f_{ss} and f_{ms} | 145 |
| 13.1.6 | Limitations | 145 |
| 13.1.7 | Implementing the simplified self-shadowing model using the SGGX distribution | 145 |
| 13.2 | Implementing the self-shadowing model with trigonometric lobes | 145 |
| 13.2.1 | Choice of a self-shadowing function | 146 |
| 13.2.2 | Trigonometric lobes: $D_{\cos}(\omega, \xi, n)$ and $D_{\sin}(\omega, \xi, n)$ | 146 |
| 13.2.3 | Normalization factors | 146 |
| 13.2.4 | Proof that distributions $D_{\sin}(\omega, \xi, n)$ can be written as a sum of distributions $D_{\cos}(\omega, \xi, n)$ | 147 |
| 13.2.5 | Closed-form expression of the attenuation coefficient $\sigma_t(\omega)$ | 148 |
| 13.2.6 | Single scattering coefficients σ_{ss} for D_{\cos} and D_{\sin} microflake distributions | 148 |
| 13.2.7 | Closed-form expression for $\int \sigma_t$ and $\int \sigma_{ss}$ | 149 |
| 13.2.8 | Sampling the distribution of visible normals for a cosine lobe $D_{\cos}(m, \xi_D, n)$ | 153 |
| 13.2.9 | Sampling the distribution of visible normals for a sine lobe $D_{\sin}(m, \xi_D, n)$ | 155 |
| 13.2.10 | Sampling the single scattering phase function f_{ss} for $D_{\cos}(m, \xi_D, n)$ | 155 |
| 13.2.11 | Sampling the single scattering phase function f_{ss} for $D_{\sin}(m, \xi_D, n)$ | 155 |
| 13.2.12 | Sampling the multiple scattering phase function f_{ms} | 156 |

| | | |
|-------------------|---|------------|
| 13.2.13 | Sampling a normal from D_{\cos} | 156 |
| 13.2.14 | Sampling a normal from D_{\sin} | 157 |
| 13.2.15 | Sampling a cosine lobe | 157 |
| 13.2.16 | Case of isotropic distribution $D(\omega) = \frac{1}{4\pi}$ | 157 |
| 13.2.17 | Combining trigonometric lobes | 158 |
| 13.3 | Implementing the simplified self-shadowing model with the SGGX distribution | 159 |
| 14 | Volume downsampling with our microscopic self-shadowing model | 161 |
| 14.1 | Downsampling algorithms | 161 |
| 14.1.1 | Overview | 161 |
| 14.1.2 | Algorithm Linear | 162 |
| 14.1.3 | Algorithm Transp | 162 |
| 14.1.4 | Downsampling using our self-shadowing model (Iso and Aniso) | 163 |
| 14.2 | Implementation and results | 165 |
| 14.2.1 | Precomputation time | 165 |
| 14.2.2 | Storage savings | 166 |
| 14.2.3 | Results | 167 |
| 14.2.4 | Limitations | 167 |
| 14.3 | Summary of the chapter | 171 |
| 14.4 | Résumé du chapitre | 171 |
| IV | Conclusion | 173 |
| 15 | Conclusion | 175 |
| 15.1 | Important ideas | 175 |
| 15.2 | Open problems | 176 |
| 16 | Conclusion | 179 |
| 16.1 | Idées importantes | 179 |
| 16.2 | Questions ouvertes | 180 |
| Appendices | | 183 |
| A | Macrosurface analysis: finding the topology of I | 183 |
| B | Rejection sampling | 184 |
| C | Derivation of general closed-form expressions for $\sigma_t(\omega)$ | 185 |
| D | Tabulated values of $\int \sqrt{\omega^T S \omega}$ | 188 |

Chapter 1

Introduction: efficient rendering with level-of-detail representations

More than a hundred and fifty thousand final frames need to be created for an average feature-length film. This number doubles for 3D stereoscopic films. In both animated features and films with special effects based on 3D computer graphics, producing these frames requires a tremendous amount of human and computer work. The total number of frames computed during the production is actually much larger than the number of final frames if we take into account all the provisional frames and test images computed to iteratively improve the film.

Many artists must be involved in the process of producing believable computer-generated worlds. In particular:

- Some artists must create believable and detailed 3D geometry.
- Some artists must paint detailed textures and design realistic materials using these textures.
- Believable animations for characters are either created manually by 3D animators or with the help of motion capture techniques. Other elements such as hair, cloth, fluids and destructions are often animated based on physical simulations with artistic control.
- Lighting artists must design and place virtual light sources such that rendering algorithms can generate a frame by computing the scattering of the light in the virtual scene.

Huge amounts of detail in both the geometry of a scene and its textures are required for reaching the visual complexity of the real world. These details often result in hundreds of gigabytes of data for rendering a single frame. This is a major problem in every studio because large geometry and texture files lead to paralyzing loading time in the software used for designing and animating a scene as well as very large rendering time. Artists must sometimes wait minutes before obtaining very noisy previews of a frame, and hours to obtain a final frame even with powerful render farms.

In this thesis, we address the problem of rendering extremely complex scenes. Our approach to this problem is *appearance-preserving prefiltering*: we design algorithms whose input is some high-resolution data (typically geometry, textures and reflectance models) and that generate data at a lower resolution, with the same distant appearance, that is, such that no difference can be seen between the input data and low-resolution data when it is viewed from a distance. An important part of our work and of prefiltering methods in general is the choice of appropriate *representations* for the prefiltered data, that is, the choice of appropriate types of rendering primitives and appropriate reflectance models. Prefiltering methods are also called *level-of-detail* or *LOD methods*, because they typically generate low-resolution renderable data at various prefiltering levels. If LODs are accurate, they can be used in place of sub-pixel or hidden details without lowering the quality of the final image.

In this chapter, we discuss the scope and context of our thesis. We first give a brief overview of rendering algorithms, and we provide details about the goals and trends of prefiltering techniques for high-quality rendering.

1.1 Rendering algorithms

Real-time and off-line rendering. Rendering is traditionally divided in two categories based on the available time budget for computing one frame. *Real-time* rendering techniques aim at generating at least 30 frames per second. It is mostly achieved with graphics processing units (GPUs), and used in interactive applications such as videos games, simulators or interactive scientific visualization. *Off-line rendering* designates high-end rendering applications where quality standards are very high, and whose computation time is often expressed in hours or tens of hours for final frames, that is a million times more than the budget of real-time rendering. In this thesis, we focus on such off-line applications that compute accurately light scattering in very complex virtual scenes.

Interactive rendering. *Interactive rendering* designates cases where the computation is not necessarily real-time but fast enough (for example expressed in seconds) for the user to move the camera and change some settings while having a visual feedback. Interactive rendering is extremely important for 3D artists when they design the animations, the materials and the lighting of a scene. Art is an iterative process and the final quality often depends on the number of iterations.

A current challenge in the film industry is designing rendering applications that support both interactive and off-line rendering so that artists can freely balance the accuracy and the rendering efficiency. This is very challenging because off-line and interactive rendering algorithms typically have very different architectures, and because the complexity of the geometry and the materials in production scenes cannot be processed interactively without simplification. Level-of-detail representations can bring large benefits in this context.

Pixel values as integrals. In this thesis, we focus on off-line rendering applications that accurately simulate light scattering in a scene from the light sources to the virtual camera. Pixels of a frame are defined as an integral of the light energy scattered by the scene and reaching a pixel. Moreover, the amount of light scattered at a point in the scene towards a pixel of the camera is itself defined as an integral of the incident light energy at this point times a scattering function that characterizes the surface of the medium at this point. The formalization of rendering as a recursive integral equation has been introduced in computer graphics by Kajiya [Kaj86] under the name *rendering equation*. This inspired all the photo-realistic rendering algorithms currently used in the film industry.

Monte Carlo rendering. Rendering algorithms used in the special effects and 3D animation industries estimate the integrals of the rendering equation by finding *light paths* between the virtual light sources and the virtual camera, based on geometrical optics. These paths contain one or more scattering events, such specular or diffuse reflections and refractions on surfaces of the 3D scene. Pixels values are estimated using Monte Carlo approaches: many paths must be found between each pixel and the lights based on random sampling techniques, such that the average the contribution of the paths provides an estimator of the amount of light scattered towards the pixel. Many algorithms have been proposed based on the seminal work of Kajiya, including many algorithms for computing volumetric light scattering in participating media such as the skin, wax, marble, clouds or milk. These algorithms can compute very precise images but are prone to noise, and computing random light paths until the convergence of each pixel value can be very long. In this thesis, we do not propose new rendering algorithms, we instead propose ways to prefilter the appearance of high-resolution rendering primitives and to represent it using memory efficient and accurate models, in order to decrease the rendering time for the same image quality. The methods introduced in this manuscript are rather independent of rendering algorithms.

1.2 Rendering large amounts of detail

Amounts of detail and image resolution. The lack of detail in a 3D scene is immediately perceived and it limits the realism of rendered pictures. In most cases, small details must be seen at the scale of pixels, for example the small irregularities of skin, the fibers of clothes or the small leaves of a plant. Ideally, the amount of detail required for rendering one frame should be approximately linear to the resolution of the frame: each of the 2 or 8 millions of pixels should approximately depend on a few details that cover approximately one pixel in screen space, since smaller details cannot be seen individually. In practice, the amount of detail at rendering is non-linear to the image resolution, for several reasons:

- It is not convenient to model 3D content at an arbitrary resolution. For example, for modeling a tree that will appear in the background, it is much easier to model the geometry at the scale of the leaves, even if leaves will be sub-pixel in the frame. Similarly, it is much easier to model a distant fur with explicit fibers than using lower-resolution primitives that would have exactly the same distant appearance. Painters know how to paint the distant appearance of foliage without painting individual leaves, but this is hard to achieve for realistic 3D rendering, especially because the position of the camera and the lighting of the scene are not known precisely at the time the data is created by 3D artists. The 3D assets must have a correct appearance when rendered with arbitrary lighting conditions and in all directions, and they are modeled using many and sometimes unnecessary details.
- A 3D object can be seen at various scales in a shot (a group of contiguous frames). For this reason, all the assets must be modeled with enough detail such that they have a sufficient resolution for close views, and they will have too many details for other frames.
- Off-line rendering algorithms compute the accurate scattering of the light in the scene. Some important details are only seen indirectly: for example, they cast shadows on the directly visible parts of the scene, or they appear in some specular reflections. Rendering algorithms often load and render entire virtual scenes even if only a small part of them is directly visible from the camera, and many of the off-screen details could often be safely prefiltered because their high frequencies have little effect on pixel values.

For all these reasons, scenes are often rendered with lots of unnecessary details. As discussed in the next section, prefiltering details without changing pixels values is not trivial in general.

The cost of rendering detailed scenes. Scenes with large amounts of detail raise important production problems:

- Large amounts of data must be loaded and processed for rendering a frame, before the actual rendering step begins. Loading can take minutes or even hours for massive scenes.
- The geometric complexity increases the cost of finding the intersections of light rays with the scene. The computation of ray intersections is a fundamental element of all off-line rendering algorithms, and it is a significant part of the total rendering cost.
- Production path tracing algorithms try to maximize the coherence of memory access to the geometry and textures. Large scenes deteriorate cache coherence and rendering efficiency.
- The evaluation and sampling of complex materials designed with many high-resolution texture layers can be very time consuming. This cost is called the *shading cost* and it is sometimes the bottleneck of production renderers.
- Because pixels are evaluated using Monte Carlo approaches, high frequencies in a scene result in noisier images, which is addressed by computing more light paths. Therefore, the cost of details is double: not only the cost increases with the complexity of the scene because computing light paths is more expensive, but more paths must be computed to avoid noise in pixel values.



(a) Photo of a piece of velvet.

(b) Photo of pine trees seen at various scales.

Figure 1.1: *Examples of complex appearances emerging from sub-resolution details. (a): The characteristic appearance of velvet emerges from its well-organized shiny fibers. Many other fabrics have such anisotropic reflections and complex appearance at grazing angles. (b): Pine trees seen at various scales. Even when pine needles are sub-pixel, the appearance of pines is complex and emerges from the scattering properties of the assemblies of bright and translucent needles.*

Out-of-core rendering and unrenderable scenes. Very detailed scenes do not fit into the main memory of computers, and *out-of-core rendering* refers to techniques for handling such scenes, by loading only parts of a scene on demand. Unfortunately, accessing the disk memory is at least one or two orders of magnitude slower than accessing the main memory^{1,2}. Moreover, out-of-core rendering requires additional memory for storing temporary data until the missing parts of a scene are loaded. Out-of-core rendering is not a mature technology, and very large scenes remain unrenderable in many studios.

1.3 Rendering with level-of-detail representations

For addressing problems raised by very complex scenes, two complementary approaches can be used. First, rendering algorithms must be optimized, for example using efficient acceleration structures such as bounding volume hierarchies, ray ordering for improving the coherency of memory access, or cache systems for reusing data from previous render tasks. Secondly, memory usage can be reduced using some type of data compression in a preprocessing step. Loss-less data compression allows for reducing loading times but it does not address the problem of noise due to high-frequency details nor scenes whose traceable geometry does not fit in the main memory. Detail prefiltering consists in simplifying rendering primitives while being aware of the effect of the simplification on pixel values. It can be thought as a form of lossy compression but it relies on hypotheses about what governs the appearance of 3D assets at a given scale, so that high spatial frequencies can be replaced by lower frequency data that have the same macroscopic appearance. This is the approach adopted in this thesis.

Appearance-preserving prefiltering is difficult. In the next chapters, we will show that prefiltering arbitrary 3D models is a difficult problem. For example, the appearance of complex objects such as fabrics (Fig. 1.1a) or trees (Fig. 1.1b) emerges from the scattering properties of the details and from masking and shadowing effects at small scales. Predicting and representing efficiently such objects at arbitrary resolutions is very challenging, and naïve prefiltering methods often fail in preserving the correct appearance at all scales. We will see that designing specific prefiltering methods for particular types of 3D assets is possible using appropriate assumptions, and that prefiltering robustly and automatically arbitrary production assets is a much more difficult problem.

LOD methods must avoid all kinds of temporal artifacts. When the resolution of an asset must be

¹<https://surana.wordpress.com/2009/01/01/numbers-everyone-should-know/>

²<https://gist.github.com/jboner/2841832>

changed during a shot, for example because the camera is moving backward, the appearance of the asset must remain consistent, and the transition between the levels of detail must be imperceptible and should not introduce flickering or popping artifacts. This is an important and challenging quality constraint for high-quality off-line rendering.

Current prefiltering techniques for off-line rendering. In the next chapters, we review techniques commonly used in production: geometric LODs, such as mesh simplification or adaptive subdivision (Chap. 3), textured-based prefiltering such as the well known mip mapping (Chap. 4) and techniques dedicated to render efficiently aggregate detail such as foliage or hair (Chap. 5). We will also discuss the validity domain of each method and show that all-scale prefiltering of complex assets such as trees remains an open problem that we address in Part II.

Drawbacks of the prefiltering approach. LODs have a rather bad reputation because many prefiltering methods are not robust and efficient enough for production. LODs are often a source of bugs and artifacts in rendered images, and 3D artists and directors do not like 3D assets being automatically simplified unless prefiltering algorithms are proven to be robust enough. Studios generally avoid using aggressive LODs when possible, although simplification techniques are mandatory for rendering very detailed scenes that could not be rendered otherwise. Moreover, we will see in chapter 6 that LOD methods raise several technical problems in production pipelines. For example, some LOD methods have a significant precomputation cost, meaning that LODs cannot be computed per frame at render time, and must but precomputed at the best place in the pipeline.

Instancing is another way of reducing memory usage in some complex scenes. It consists in defining multiple instances of a geometry in a scene while storing the geometry only once in the main memory. This allows great savings in scenes that have many buildings or trees with the same geometry. With instancing, the level of detail must be chosen according to the instance that is the closest to the camera, leading to sub-optimal resolutions for instances in the background. Using additional LODs would increase memory usage instead of decreasing it.

The future of prefiltering methods. The progressive increase of available memory in computers allows the rendering of extremely complex scenes compared to what was technically possible a few years ago. Many problems are actually solved years after they are studied simply because computers are more powerful, and approximate solutions get replaced by simpler high-quality methods. A huge amount of work has been done on generating level-of-detail representations of rendering primitives, and a large part of this work is now outdated, although inspiring. For instance, Reeves [RB85] proposed a volume approximation for shading its particle-based foliage, and stated that ‘it would be a formidable task to shade each particle exactly, calculating whether it is in shadow and determining if it should be highlighted. In fact, unless the camera points directly at the sun from within the tree, it is almost impossible to tell whether or not any one leaf should be in shadow’. It is now possible to compute unbiased shadowing and multiple scattering in trees with thousands of leaves.

One could wonder what will be the place of LOD methods in future production pipelines. On the one hand, as computers become more and more powerful, brute force approaches become affordable and allow for overcoming problems raised by previous approximate solutions. On the other hand, rendering pipelines will always be highly sub-optimal without LOD techniques and resources will be wasted because of unnecessary details. Further research on efficient and robust prefiltering can bring huge benefits for both interactive visualization and final frames.

1.4 Overview of the thesis

Our goal is to investigate new prefiltering techniques for improving the rendering efficiency in the case of very detailed scenes. As mentioned above, we focus on high-end rendering applications and we target high-quality rendering.

The first three chapters of this manuscript present previous work on geometric simplification (Chap. 3), surface-space prefiltering (Chap. 4) and aggregate detail (Chap. 5).

In Part II, we present our contributions on all-scale prefiltering of complex assets such as trees. Inspired by previous work on both surface-based prefiltering and volumetric approximations of aggregate detail, we revisit several techniques and combine them to achieve robust and seamless prefiltering. We show that we obtain high-quality results for various assets such as trees which are not supported by previous methods. This work has been published and presented at the conference Eurographics 2017 [LN17].

In Part III, we address the problem of downsampling high-resolution voxel grids representing heterogeneous participating media, while preserving their appearance at all scales. We discuss the limitations of naïve prefiltering methods and we review the related work of Zhao, Wu et al. [ZWDR16], in which volumes are downsampled based on optimizations of scattering parameters with image-space errors. We introduce an alternative method that is more general, that generates accurate results and that is significantly faster. This work has been published and presented at the conference Eurographics 2018 [LN18].

1.5 Summary of our contributions

In Part II, we present several contributions for the prefiltering of complex assets at arbitrary scales:

- We analyze the problem of prefiltering assets and the limitations of current methods. We propose a new hybrid LOD approach that combines the strengths of surface-based and volumetric prefiltering methods (Chap. 6).
- We introduce an algorithm that analyzes meshes and detects macrosurfaces at a given scale (Chap. 7).
- We propose a new vertex placement strategy for simplifying meshes with edge collapse operations, based on a two-sided geometric error metric (Sec. 8.2).
- We propose a technique for prefiltering albedos and microfacet distributions in the edge collapse framework (Sec. 8.3).
- We propose an alternative technique for macrosurface prefiltering in the case of structured quad meshes, which are the standard representation in many studios (Sec. 8.4).
- We introduce a voxelization algorithm that estimates occlusion probabilities of sub-resolution intricate geometry and that avoids aliasing (Chap. 9).

In Part III of the thesis, we present the following contributions:

- Based on our analysis of the problem of prefiltering high-resolution heterogeneous volumes (Chap. 11), we propose a new participating medium model based on the microflake model [JAM*10a] that is adapted to volume downsampling (Chap. 12).
- We show that choosing appropriate functions in our model allows for deriving closed-form expressions which are required for its efficient implementation. We provide all these derivations (Chap. 13).
- Based on our model, we introduce a new downsampling approach and we demonstrate its benefits compared to previous work (Chap. 14).

1.6 Industrial collaborations

During this PhD, I worked as an intern in the studios of Weta Digital in Wellington, New Zealand (July-August 2015), and at the Walt Disney Animation Studios in Burbank, CA, USA (October-December 2017). I studied various aspects of LOD methods for production. Although confidentiality clauses prevent

us from providing information about the pipelines of these studios in this document, discussions about our motivations and general rendering problems are largely inspired by these internships and by our discussions with rendering researchers and engineers.

Chapter 2

Introduction : rendu efficace avec des niveaux de détail

Plus de cent cinquante mille images doivent être produites en moyenne pour former un long métrage. Que ce soit pour des films d'animation 3D ou pour des films demandant beaucoup d'effets spéciaux calculés sur ordinateur, produire ces images demande une quantité de travail gigantesque et nécessite la collaboration de nombreux artistes spécialistes de l'informatique graphique 3D. Pour créer des mondes virtuels convaincants, ces artistes doivent produire :

- de la géométrie en 3D convaincante et suffisamment détaillée,
- des textures et des matériaux réalistes pour habiller cette géométrie,
- des animations crédibles, soit manuellement, soit en s'aidant de procédés de capture de mouvements, soit en utilisant des simulations physiques pour animer les cheveux, les fluides ou les vêtements,
- des éclairages virtuels à la fois plausibles et esthétiques qui sont utilisés par un logiciel de rendu qui va calculer la diffusion de la lumière dans la scène virtuelle pour produire une image du film.

Une énorme quantité de détails est nécessaire pour reproduire virtuellement la complexité du monde réel. Ces détails sont en grande partie géométriques, mais des textures très détaillées sont également nécessaires. La somme de ces détails conduit à l'élaboration de scènes virtuelles dont la taille mémoire peut dépasser des centaines de gigaoctets pour le calcul d'une seule image. Gérer de telles quantités de données est un défi majeur pour tous les studios d'effets spéciaux : les temps de chargement sont extrêmement longs et ils paralysent les artistes qui doivent attendre plusieurs minutes voire des heures pour obtenir ne serait-ce qu'un aperçu bruité de l'image finale, et ce même avec des ordinateurs puissants ou d'immenses serveurs de calcul.

Dans cette thèse, nous étudions les problèmes qu'affrontent les studios lorsque des scènes très détaillées doivent être produites et rendues. Notre but est de concevoir des algorithmes qui permettent de réduire les temps de calcul sans porter atteinte à la qualité des images finales. Notre approche du problème est une approche de *préfiltrage* : en partant de données de haute résolution, nos algorithmes génèrent des données de plus basse résolution afin que les ressources des ordinateurs ne soient pas gaspillées pour des détails qui ne sont pas visibles dans l'image parce qu'ils sont masqués, hors-champ ou sous-pixel. Ces données préfiltrées sont appelées des *LODs* (*levels of detail* en anglais).

Dans ce chapitre d'introduction, nous délimitons le périmètre de nos recherches, nous donnons un aperçu des méthodes de rendu et nous présentons notre approche et ses avantages pour augmenter l'efficacité du rendu d'images de haute qualité.

2.1 Les algorithmes de rendu

Rendu temps réel, rendu off-line. Le domaine du rendu est traditionnellement divisé en deux catégories en fonction du budget temps disponible pour le rendu d'une image. Le *rendu temps réel* désigne un ensemble de techniques dont le but est de générer plus de 30 images par seconde pour des applications interactives comme des simulateurs, de la visualisation scientifique ou des jeux vidéos. Le rendu temps réel repose principalement sur les cartes graphiques pour atteindre la performance requise. A l'opposé, le *rendu off-line* est la méthode de rendu utilisée pour les images prédictives en architecture, la publicité, les effets spéciaux et les films d'animation qui utilisent l'informatique graphique 3D. Dans les applications de rendu off-line, c'est la qualité de l'image qui prime et non l'efficacité du rendu. Il n'est pas rare d'attendre des heures ou des dizaines d'heures pour le rendu d'une image, ce qui permet à l'algorithme de calculer précisément la diffusion de la lumière dans la scène et de générer des images convaincantes voire indiscernables d'une photographie. La dichotomie entre ces méthodes de rendu est grande : de quelques millisecondes à une dizaine d'heures, le budget temps est multiplié par un million. Dans cette thèse, nous étudions des algorithmes pour le rendu off-line, avec une attention particulière pour les applications liées au cinéma qui demandent le calcul de très nombreuses séquences d'images.

Rendu interactif. Le *rendu interactif* est une troisième famille d'algorithmes qui vient se placer entre le rendu temps réel et le rendu off-line. Son but est de générer des images de manière suffisamment rapide pour permettre une interaction de l'utilisateur avec la scène virtuelle, sans pour autant imposer une limite stricte de plusieurs dizaines d'images par seconde. Avec le rendu interactif, l'utilisateur peut déplacer une caméra virtuelle, changer le matériau d'un objet ou créer une animation tout en ayant à la fois un bon aperçu de l'image finale et en gardant une approche d'essai-erreur avec des retours visuels rapides. Le rendu interactif est très important dans les domaines du design de scènes virtuelles.

Ce type de rendu est un défi actuel pour l'industrie des effets spéciaux. En effet, il est difficile de concevoir des logiciels de rendu qui peuvent à la fois produire des images de manière interactive et calculer les images finales efficacement car les architectures logicielles de ces algorithmes sont très différentes. Dans le cas des scènes complexes, il est parfois impossible d'obtenir une prévisualisation interactive tant la quantité de détails engendre de longs temps de chargement et de calcul. Les algorithmes de niveaux de détail sont alors d'une grande utilité.

Les valeurs de pixels définies comme des intégrales. Dans cette thèse, nous nous concentrons sur les algorithmes de rendu précis qui calculent entièrement la diffusion de la lumière dans une scène, c'est à dire les multiples réflexions, réfractions et diffusions qui mènent les rayons lumineux issus des sources de lumière virtuelles jusqu'à la caméra. Ces calculs reposent sur des modèles physiques de réflectance des surfaces des objets virtuels. Pour calculer la valeur d'un pixel, il faut calculer l'intégrale de toute la lumière qui l'atteint. De plus, la quantité de lumière réfléchiée par les surfaces dans la direction du pixel est elle-même définie comme une intégrale de la lumière incidente à cette surface fois une fonction de réflectance. Cette formalisation du rendu par des intégrales récursives est étudiée en informatique graphique depuis les travaux de Kajiya [Kaj86] et c'est cette approche qui est utilisée pour calculer des images réalistes dans l'industrie des effets spéciaux.

Le rendu par des techniques de Monte Carlo Les algorithmes modernes de rendu de haute qualité estiment les intégrales en trouvant beaucoup de chemins de lumière entre les sources et la caméra de la scène virtuelle, en s'appuyant sur les lois de l'optique géométrique. Ces chemins peuvent comporter plusieurs interactions avec les objets de la scène, par exemple plusieurs réflexions spéculaires. Comme les intégrales n'ont presque jamais de solutions analytiques, les valeurs des pixels sont calculées avec des approches de Monte Carlo, c'est à dire numériquement en échantillonnant aléatoirement le domaine d'intégration. Pour cela, beaucoup de chemins lumineux sont calculés et leur contribution moyenne est un estimateur de l'intégrale du pixel. Cette méthode a été l'objet d'un grand nombre de travaux et beaucoup d'algorithmes de rendu ont vu le jour, y compris beaucoup de méthodes calculant la diffusion de la lumière dans les milieux participatifs translucides comme le marbre, le lait, les nuages, ou la cire. Tous ces algorithmes calculent des images très précises, si tant est qu'on leur en donne le temps : les images sont victimes du bruit caractéristique des méthodes de Monte Carlo et la convergence est parfois

très longue, voire tellement longue dans certains cas pathologiques que plusieurs mois de calculs ne suffiraient pas. Dans cette thèse, nous ne proposons pas de nouveaux algorithmes de rendu. Notre but est de simplifier les données qui sont rendues afin que tous ces algorithmes soient plus rapides, le tout sans amoindrir la qualité des images calculées. Nos travaux sont donc relativement indépendants des algorithmes de rendu, bien qu'il faille savoir comme ils fonctionnent pour les rendre plus efficaces.

2.2 Le rendu de scènes très détaillées

Quantité de détails et résolution d'image. Une quantité trop faible de détails dans une scène est très perceptible et ruine le réalisme d'une image. Idéalement, des détails doivent être modélisés et rendus jusqu'à l'échelle des pixels, en créant par exemple de petites irrégularités sur les visages, des fibres sur un tissu ou de petites feuilles sur une branche. En théorie, la quantité de détail nécessaire pour le rendu d'une image pourrait être linéaire du nombre de pixels de l'image, puisque seuls les détails qui peuvent être discernés dans l'image sont indispensables. En pratique, ce n'est jamais le cas, et ce pour plusieurs raisons :

- Les artistes créent les objets 3D à une résolution qui leur convient, et pas à des résolutions arbitrairement basses pour les besoins d'une image. Par exemple, créer un modèle d'arbre 3D réaliste est plus facile en modélisant la géométrie à l'échelle des feuilles, même si ces feuilles sont sous-pixel dans l'image finale. Si les peintres savent créer directement l'apparence d'un feuillage distant sur une toile, cette approche est bien plus complexe dans le cadre d'un rendu réaliste, où le feuillage doit avoir une apparence plausible sous tous les points de vues et pour tous les éclairages possibles. De même, il est bien plus facile de créer une fourrure réaliste en modélisant un ensemble de poils qu'en utilisant d'autres modèles de plus basse résolution dont l'apparence distante est celle d'une fourrure.
- Le point de vue de la caméra n'est pas connu précisément lors de la modélisation, il est donc difficile de modéliser un objet en s'arrêtant exactement à l'échelle nécessaire. De plus, si la caméra ou l'objet bougent, un objet 3D peut être vu à plusieurs échelles, et il faut qu'il ait suffisamment de détails pour chacune des images du film. Les objets 3D sont donc modélisés à de hautes résolutions au cas où ces détails soient nécessaires pour une scène du film. A cause de cela, la quantité de détails est souvent beaucoup plus grande que nécessaire dans une image donnée.
- Les algorithmes de rendu off-line calculent précisément la diffusion de la lumière. Les chemins lumineux parcourent toute la scène virtuelle et pas seulement la partie de la scène directement visible à l'écran. Par exemple, les objets 3D peuvent projeter des ombres complexes ou être vus dans des reflets : il est donc nécessaire de garder des détails pour les objets même si ils sont hors-champ. Néanmoins, une grande partie de ces détails indirects sont inutiles en pratique car malgré leur contribution indirecte aux pixels, ils ne sont pas discernables individuellement.

En résumé, les scènes sont toujours rendues avec des détails qui contribuent peu aux valeurs des pixels. Ou plus exactement, c'est l'ensemble des détails dans un voisinage qui contribue au transport de la lumière, mais les hautes fréquences de ces détails ne contribuent pas à l'image. Nous verrons dans la suite de ce chapitre que préfiltrer ces détails sans modifier l'image finale est un problème difficile dans bien des cas.

Le coût du rendu des scènes détaillées. Les scènes virtuelles très détaillées peuvent engendrer d'important surcoûts :

- Les scènes complexes doivent être chargées en mémoire pour le rendu d'une image. Plusieurs étapes de préparation des données pour l'algorithme de rendu pèsent également sur l'efficacité du rendu. Ces étapes représentent parfois plusieurs minutes ou plusieurs heures de chargement et de calculs, avant même que le rendu à proprement parlé commence.
- La complexité géométrique d'une scène augmente le coût du calcul des chemins lumineux, qui est une des briques fondamentales des algorithmes de rendu off-line.

- Du fait de la grande taille des scènes en mémoire, les opérations d'accès mémoire peuvent être un frein majeur à l'efficacité d'un algorithme. Les algorithmes de rendu ont des stratégies pour maximiser la cohérence de l'accès aux données pour réduire ces coûts, mais plus la scène est complexe, plus c'est difficile.
- Évaluer et échantillonner les matériaux de la scène peut également être très coûteux, notamment lorsque les textures sont de très haute résolution et que beaucoup de textures sont utilisées pour le même objet. Ce coût est parfois le principal goulot d'étranglement des algorithmes de rendu.
- Les méthodes de Monte Carlo donnent des valeurs de pixels bruitées si l'échantillonnage du domaine d'intégration est trop faible. De manière générale, plus les fonctions à intégrer ont des hautes fréquences, plus le bruit est important. Une scène détaillée engendre donc un double surcoût : non seulement chaque chemin lumineux est plus cher à calculer, mais il faut aussi calculer plus de chemins pour que l'image atteigne un niveau de bruit acceptable.

Rendu out-of-core et scènes trop complexes pour être rendues. Les scènes extrêmement détaillées ne peuvent être chargées entièrement en mémoire, faute de place. Les algorithmes de rendu *out of core* sont les algorithmes qui permettent de calculer le rendu sans avoir à charger toute la scène en même temps : un système de cache permet de charger sur demande les parties de la scène nécessaires à un moment donné du calcul, et de libérer de la mémoire lorsque certaines données ont déjà servi. Ces algorithmes ne sont pas optimaux car l'accès à la mémoire de stockage (disques dur, SSD) est bien plus long que l'accès à la mémoire vive, d'au moins un ou deux ordres de grandeur^{1,2}. De plus, les algorithmes *out of core* ont des surcoûts en terme de mémoire pour stocker des informations sur l'état du rendu (typiquement des chemins lumineux incomplets) en attendant que les données manquantes soient chargées, ce qui contribue aussi à la diminution des performances. Le rendu out-of-core n'est pas considéré comme un technologie mature et beaucoup de studios n'en disposent pas. Pour eux, les scènes trop complexes ne peuvent tout simplement pas être rendues.

2.3 Le rendu efficace avec des niveaux de détail

Pour affronter les problèmes soulevés par les scènes complexes, deux approches complémentaires peuvent être envisagées. La première consiste à optimiser les algorithmes de rendu, par exemple en utilisant des structures accélératrices comme les *bounding volume hierarchies*, en triant les rayons pour faire en sorte que les accès mémoire soit cohérents, ou en stockant certaines informations dans des caches pour qu'elles puissent être rentabilisées pour le calcul de plusieurs chemins lumineux. Une seconde stratégie consiste à réduire la quantité de données nécessaires pour le rendu, avant l'étape de rendu elle-même. La compression sans perte des données peut réduire les temps de chargement mais ne résout pas le problème du bruit causé par les détails de hautes fréquences, ni le problème des scènes massives qui ne tiennent pas en mémoire une fois décompressées. Le préfiltrage de détails consiste à réduire la quantité de données tout en ayant conscience de l'effet du filtrage sur l'apparence des objets 3D. Les techniques de préfiltrage peuvent être vues comme des techniques de compression avec pertes adaptées au rendu, et basées sur des hypothèses de ce qui constitue l'apparence d'un objet 3D à une échelle donnée. C'est l'approche que nous adoptons dans cette thèse.

Le problème du préfiltrage est un problème difficile. Dans les prochains chapitres, nous montrerons que préfiltrer automatiquement des modèles 3D arbitraires est un problème difficile. Pour citer deux exemples, l'apparence distante des tissus (Fig. 2.1a) et des feuillages (Fig. 2.1b) provient des propriétés optiques complexes des détails sous-résolution (les fibres, les feuilles) et de leur agencement qui engendre des effets de masquage et d'ombrage particuliers. Prédire et représenter efficacement l'apparence de tels objets à des échelles arbitraires est un important problème de recherche, et les méthodes naïves ne parviennent pas à préserver une apparence cohérente à toutes les échelles. Nous verrons que mettre au point des méthodes de filtrage pour des types particuliers d'objets 3D est possible en s'aidant

¹<https://surana.wordpress.com/2009/01/01/numbers-everyone-should-know/>

²<https://gist.github.com/jboner/2841832>



(a) Photographie d'une pièce de velours.



(b) Photographie de pins vus à plusieurs échelles.

Figure 2.1: Exemples d'apparences complexes émergeant de détails sous-résolution. (a): L'apparence caractéristique du velours provient de ses fibres brillantes et bien organisées. De nombreux autres tissus ont aussi des apparences anisotropes et des reflets particuliers aux angles rasants. (b): Des pins vus à plusieurs échelles. Même lorsque les aiguilles de pin sont sous-pixel, l'apparence du feuillage est complexe et provient des propriétés des ensembles d'aiguilles brillantes et translucides.

d'hypothèses bien choisies, et que le préfiltrage automatique et fiable d'objets arbitraires est un problème bien plus difficile.

Les méthodes de préfiltrage doivent éviter toute forme d'incohérence temporelle. Quand la résolution d'un objet 3D doit être changée au cours d'une séquence d'image, par exemple lorsque la caméra virtuelle se déplace vers l'arrière, l'apparence de l'objet doit rester cohérente et la transition entre les niveaux de détail doit être imperceptible. Cette contrainte ajoute de la difficulté au problème du préfiltrage.

Les méthodes actuelles de préfiltrage pour le rendu off-line. Dans les prochains chapitres, nous passerons en revue des techniques de préfiltrage existantes : le préfiltrage de géométrie, et en particulier la simplification de maillage et les méthodes de subdivision (Chap. 3), les méthodes de filtrage de textures comme le célèbre *mip mapping* (Chap. 4) et les techniques dédiées au préfiltrage des géométries complexes comme les feuillages ou les cheveux (Chap. 5). Nous parlerons également du domaine de validité de chaque méthode et nous montrerons que préfiltrer précisément certains objets tels que les arbres est un problème non résolu par les méthodes existantes. Nous proposons des solutions dans la partie II.

Les inconvénients de l'approche de préfiltrage Les LODs pâtissent d'une réputation plutôt mauvaise dans les studios de production parce que beaucoup de méthodes de préfiltrage ne sont pas suffisamment fiables et précises pour les besoins de l'industrie. Les LODs sont parfois source de problèmes et de défauts dans les images rendues, et ni les artistes ni les réalisateurs n'aiment l'idée que les objets 3D soient simplifiés automatiquement avant le rendu avec des risques de perte de qualité. De plus, nous verrons dans le chapitre 6 que les méthodes de préfiltrage posent plusieurs problèmes pratiques lorsqu'il s'agit de les utiliser dans de grands studios de production. Par exemple, du fait que le préfiltrage a un coût non négligeable, il ne peut avoir lieu au moment du rendu, il doit plutôt être précalculé en amont et rentabilisé pour le calcul de plusieurs images. L'*instancing* est une autre stratégie pour réduire la taille des scènes très détaillées. Cette méthode consiste à créer plusieurs instances d'une géométrie dans une scène tout en ne stockant la géométrie qu'en un seul exemplaire en mémoire. Cela permet de réduire de beaucoup la taille d'une scène si elle comporte beaucoup d'arbres ou de bâtiments similaires. Avec l'*instancing*, le niveau de détail utilisé pour le rendu doit être choisi en fonction de l'instance qui a le plus besoin de détail, typiquement celle qui est la plus proche de la caméra. Cela veut dire que toutes les autres instances, qui partagent la même géométrie, auront trop de détails pour le calcul de cette image. Dans cette situation, utiliser des LODs augmente la taille mémoire de la scène au lieu de la diminuer. Pour toutes ces raisons, les studios évitent quand ils le peuvent de recourir à des méthodes de préfiltrage.

Néanmoins, ils sont obligés d'utiliser des méthodes de simplification lorsque les scènes sont trop détaillées, sans quoi elles ne pourraient pas être rendues.

L'avenir des méthodes de préfiltrage. L'accroissement progressif de la mémoire vive des ordinateurs permet le rendu de scènes complexes qui étaient considérées comme impossible à traiter auparavant. Beaucoup de problèmes sont en fait résolus quelques années après avoir été identifiés, simplement parce les ordinateurs sont devenus plus performants. Les méthodes qui utilisent des approximations sont alors remplacées par des méthodes plus coûteuses mais plus précises. Beaucoup de recherches ont été menées sur la génération de niveaux de détail et une grande partie des solutions proposées n'ont plus de raisons d'être utilisées bien qu'elles puissent encore inspirer des travaux actuels. Pour citer un exemple, Reeves [RB85] a proposé un modèle volumique pour calculer l'ombrage approximatif dans les feuillages de ses arbres constitués de particules. Dans son article, il déclare que « déterminer de manière exacte l'illumination et l'ombrage de chaque particule demanderait des calculs considérables. En réalité, à moins que la caméra soit tournée exactement vers le soleil, il est à peu près impossible de déterminer si une feuille est ou n'est pas dans l'ombre. » Actuellement, il est possible et même courant de calculer en quelques minutes et de manière très précise la diffusion de la lumière dans des arbres virtuels faits de milliers de feuilles.

On peut se demander quelle sera la place des méthodes de préfiltrage dans quelques années dans les studios d'effets spéciaux. Des ordinateurs de plus en plus puissants permettent de résoudre des problèmes par des approches précises et directes et de s'affranchir des problèmes posés par les méthodes approximatives. D'un autre côté, les algorithmes de rendu seront toujours sous-optimaux sans préfiltrage et les ressources seront gaspillées à cause de détails inutiles. Des avancées dans le domaine du préfiltrage précis et fiable pourraient être extrêmement utiles pour le rendu interactif ou final des images.

2.4 Aperçu de la thèse

Notre but est de proposer de nouvelles techniques de préfiltrage pour accélérer le rendu des scènes très détaillées. Nous nous plaçons dans le cadre du rendu off-line de haute qualité.

Les trois premiers chapitres de cette thèse (Partie I) présentent les travaux existants sur la simplification géométrique (Chap. 3), sur le préfiltrage de l'apparence des surfaces (Chap. 4), et les approximations des grands ensembles de détails (Chap. 5).

Dans la Partie II, nous présentons nos contributions pour le filtrage des objets 3D complexes comme les arbres. Inspirés par les méthodes existantes sur le filtrage de surfaces et de géométrie complexe, nous revisitons plusieurs techniques et nous les combinons de manière à préfiltrer des apparences complexes à toutes les échelles, avec des transitions imperceptibles entre les niveaux de détail. Nous montrons que notre méthode produit des LODs précis pour des types d'apparence mal supportés par les méthodes existantes. Ce travail a été publié et présenté à la conférence Eurographics 2017 [LN17].

Dans la Partie III, nous nous intéressons au problème de la simplification des grilles de voxels de haute résolution représentant des milieux participatifs hétérogènes, tout en préservant leur apparence à toutes les échelles. Nous présentons les travaux existants et leurs limitations, et en particulier le travail de Zhao, Wu et al. [ZWDR16] dans lequel ils proposent une solution au problème. Nous montrons que notre approche est plus générale et plus rapide tout en étant précise. Nos résultats ont été publiés et présentés à la conférence Eurographics 2018 [LN18].

2.5 Résumé de nos contributions

Dans la Partie II, nous présentons plusieurs contributions pour le préfiltrage d'objets complexes à n'importe quelle échelle :

- Nous proposons une analyse du problème du préfiltrage pour les géométries complexes et nous

expliquons les limitations des méthodes existantes. En partant de ces constats, nous proposons une nouvelle approche de préfiltrage hybride qui combine les atouts des représentations surfaciques et volumiques (Chap. 6).

- Nous proposons un algorithme qui analyse des maillages et détecte leurs macrosurfaces à une échelle donnée (Chap. 7).
- Nous proposons une nouvelle stratégie de placement de vertex dans le cadre de la simplification de maillage par edge collapse, en optimisant une erreur géométrique dans l'esprit de la distance de Hausdorff entre deux surfaces (Sec. 8.2).
- Nous proposons une technique de préfiltrage des albedos et des distributions de microfacettes dans le cadre de la simplification de maillage par edge collapse (Sec. 8.3).
- Nous proposons une deuxième technique de préfiltrage de macrosurfaces, plus adaptée au cas des maillages réguliers de quadrilatères qui sont très utilisés en production (Sec. 8.4).
- Nous proposons un algorithme de voxélisation qui calcule les probabilités d'occlusion locale dans la géométrie sous-résolution, tout en évitant les problèmes d'aliasing (Chap. 9).

Dans la Partie III, nous présentons les contributions suivantes :

- En partant d'une analyse du problème de préfiltrage des volumes hétérogènes de haute résolution (Chap. 11), nous proposons un nouveau modèle de milieu participatif issu du modèle de microflakes [JAM*10a]. Ce nouveau modèle est plus adapté au problème de la simplification volumique (Chap. 12).
- Nous montrons qu'en choisissant soigneusement certaines fonctions, il est possible d'obtenir des formes closes pour notre modèle, qui sont nécessaires pour son implémentation efficace (Chap. 13).
- Grâce à notre modèle, nous proposons une nouvelle méthode de simplification volumique et nous montrons ses avantages par rapport aux méthodes existantes (Chap. 14).

2.6 Collaborations industrielles

Pendant la préparation de cette thèse, j'ai eu la chance de travailler dans les studios de Weta Digital à Wellington, en Nouvelle Zélande (juillet et août 2015), et dans les studios d'animation de Walt Disney à Burbank, en Californie (de octobre à décembre 2017). J'y ai étudié plusieurs aspects des problèmes de préfiltrage et des méthodes de rendu en production. Des clauses de confidentialité nous empêchent de donner des détails sur certains points techniques et de montrer les résultats obtenus. En revanche, dans les différents chapitres de ce manuscrit, la description des problèmes, des contraintes de production et de nos motivations est très largement inspirée de nos discussions avec des chercheurs et des ingénieurs dans ces studios.

Part I

Level-of-detail techniques for efficient rendering

From the beginning of computer graphics, rendering very detailed scenes has been identified as a major long-term challenge. Starting from the 70s', several researchers introduced the main concepts of level-of-detail representations, which allow for adapting the complexity of a scene for a particular rendering task in order to avoid aliasing and the waste of computational resources due to useless object-space details. For instance, Catmull described an algorithm that 'subdivides a patch into successively smaller subpatches until a subpatch is as small as a raster-element, at which time it can be displayed' [Cat74]. Clark proposed a hierarchy of models that provides a way to 'vary the amount of detail in a scene both according to the screen area occupied by the objects in the scene and according to the speed with which an object or the camera is moving', as well as the idea of ensuring that computation time 'grows linearly with the visible complexity of a scene rather than as a worse than linear function of the object-space complexity' [Cla76]. Kajiya [Kaj85] proposed the idea of the multi-scale modeling of surface appearance using either explicit geometry, texture mapping or BRDF models depending on the scale.

A huge amount of work has been done on rendering more and more complex scenes with reasonable rendering time. As mentioned in the introduction chapter, many level-of-detail methods are now outdated, because they were designed for primitives and rendering algorithms that are not used any more, or because our computers can now compute more accurate solutions quite easily. On the other hand, level-of-detail techniques such as mip mapping [Wil83] or mesh simplification [LWC*02] are still used today and continue to inspire new methods. In this part of the dissertation, we review previous work on prefiltering geometry (Chap. 3), surface reflectance (Chap. 4) and aggregate detail (Chap. 5). We focus on techniques that inspired our work and on the recent research related to our thesis.

Chapter 3

Previous work on geometric level-of-detail representations

Following the formalization of level-of-detail approaches for efficient rendering in the 70s' [Cla76], the automatic simplification of high-resolution geometry has been addressed by many authors and became an entire discipline starting from the early 90s'. Such algorithms and in particular triangle mesh simplification algorithms [LWC*02] are now considered a mature technology [TPC*10]. Besides polygonal meshes, other level-of-detail representations of 3D geometry have been explored, such as surfels [PZvBG00], point set surfaces [ABCO*01, AA04, AK04] and voxel-based representations [GM05, YLM06, LK10a, HN12]. We also review them in this chapter.

In rendering applications, the appearance of a 3D object is due to both its geometry and its surface reflectance models. Geometric prefiltering and reflectance prefiltering are closely related but often addressed separately by authors. This chapter mainly focuses on the former, and reflectance prefiltering is discussed in greater detail in chapter 4.

All the methods presented in this chapter share a common limitation: they cannot produce accurate LODs of complex geometry such as trees at coarse resolutions. This is because they explicitly or implicitly assume that the simplified geometry is large, simple or smooth enough at the prefiltering scale. Alternative methods that specifically aim at prefiltering intricate geometry are discussed in chapter 5.

3.1 Mesh simplification algorithms in computer graphics

Mesh simplification received the greatest attention because meshes are the standard primitive not only for rendering surfaces but also in many fields of computer graphics such as animation, shape modeling and physical simulations. The book *Level of detail for 3D graphics* [LWC*02] has been written by some of the most influential researchers in the domain of mesh simplification and it summarizes previous work done in the 90s' and early 00s'. We refer the reader to this book for a comprehensive description of triangle mesh simplification algorithms and quality metrics. Some of these algorithms are still commonly used nowadays, for instance in modeling software and geometry acquisition pipelines. In this section, we give an overview of the topic, mention some recent publications and discuss the aspects that are important for our thesis such as seamless transitions between LODs. In chapter 8, we rely on this previous work for triangle and quad mesh prefiltering.

3.1.1 Overview of mesh simplification algorithms

Manifold meshes. Polygonal meshes are collections of vertices and polygons that connect these vertices. Some definitions include the case of lines, *i.e.* edges that are not adjacent to any faces, although

meshes are more often considered as collections of triangles, quadrilaterals (*quads*) or other polygons. Meshes in computer graphics are mostly used for representing the boundaries of opaque objects or the interface between two media (they are sometimes called *B-rep* representations). For this reason, most meshes model 2-manifold surfaces, that is surfaces that are locally similar (or *homeomorphic*) to a disc. Manifold surfaces are either closed or with boundaries, in which case the geometry at the boundaries is homeomorphic to half-discs (Fig. 3.1).

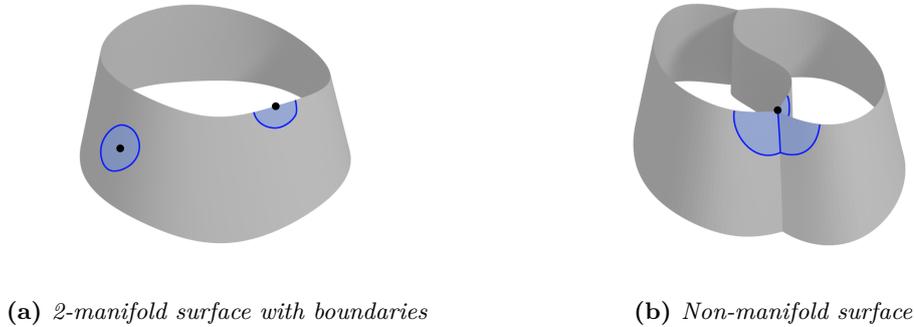


Figure 3.1: Example of a 2-manifold surface with boundaries (a) and a non-manifold surface (b). The geometry around points of a manifold surface is homeomorphic to discs or half-discs.

Discrete, continuous and seamless LODs. Mesh simplification algorithms can generate multiple versions of a geometry with different levels of detail. These LODs are often called *discrete LODs*. In rendering applications such as games, LODs are chosen depending on various criteria such as their distance to the camera, and the sudden change of LOD often leads to visible popping artifacts when the camera moves. Some authors proposed *continuous LOD* (or *CLOD*) frameworks [LKR*96, Hop96], in which a list of coarsening operations describes the transformation from the high- to the low-resolution mesh. This approach is in fact equivalent to a large set of discrete LODs, and does not necessarily support perfectly seamless (invisible) transitions between the LODs. There are techniques for morphing high-resolution surfaces into low-resolution surfaces and avoiding popping artifacts. This is called either *LOD interpolation* [Tur92, BRS96], *geomorphs* [Hop96] or *continuous LODs* [TB94], which can be ambiguous. All these methods rely on mappings between vertices of high- and low-resolution surfaces. In this dissertation, we will refer to it as either *LOD interpolation* or *seamless transitions between LODs*.

Budget-based and fidelity-based LODs. Most mesh simplification algorithms are characterized by the local coarsening operations that are applied iteratively on a mesh. The user often asks the algorithm to stop when the complexity of the mesh has been reduced to a given vertex or face count. In this case, the algorithm must maximize the geometric fidelity without exceeding the specified budget. This is called either *budget-based simplification* [LWC*02], *min- ϵ simplification* [CMS98] or *best-effort simplification* [WLC*03]. In some applications, it is more convenient for the user to provide a minimum quality criterion, for instance a level of geometric accuracy, and to ask the algorithm to simplify the mesh as much as possible without violating the quality constraints. This is called either *fidelity-based simplification* [LWC*02] or *min-# simplification* [CMS98].

Quality metrics. Budget-based and fidelity-based simplification both require some quality metrics for identifying the coarsening operations that are the less aggressive. These coarsening operations are usually applied first, because the order in which these operations are performed has a large impact on the quality of the resulting surface [COM98]. We will see that many metrics have been proposed, including various aspects such as geometric accuracy, attribute deviation (for example in the case of meshes with per-vertex colors [Hop96]), texture deviation [COM98], or view-dependent criteria that are mandatory for the simplification of large terrains [Hop97, LE97, XESV97].

Related algorithms. Mesh simplification is related to other mesh algorithms such as *remeshing*, which consists in generating a new mesh *from scratch* that approximates well the input mesh, without relying on coarsening operations. The goal of remeshing algorithms is usually to generate meshes with special properties, for example with nicely shaped and evenly sized polygons, or using only quadrilateral faces [BLP*12]. *Surface reconstruction* algorithms from points clouds [HDD*92,KBH06] are also related in the sense that they could be used for decreasing the complexity of a mesh, although it is a rather uncommon approach because surface reconstruction is more difficult than mesh simplification.

3.1.2 Coarsening operations for triangle meshes

The majority of work on mesh simplification has focused on the case of triangle meshes because such meshes are the most general representation (other polygons can be triangulated). Moreover, triangles are convenient because they are always flat and convex. Well known coarsening operations for triangle meshes include:

- The vertex removal operation, which was one of the first proposed coarsening operation. It consists in removing a vertex and all its adjacent faces, and to fill the hole (*re-triangulation*) [SZL92, Tur92, BRS96].
- The vertex clustering operation, or *cell collapse*, consists in merging a set of neighboring vertices, and finding a good representative vertex for each cluster. This operation was first proposed for a regular grid of clustering cubes [RB93] and it has been improved by centering the cells around important vertices [SS95].
- The edge collapse operation (Fig. 3.2) can preserve the topology of the mesh and it allows for generating accurate decimated surfaces by optimizing vertex positions at each collapse [HDD*93, GH97, LT98].
- The vertex contraction is similar to the edge collapse but it can be applied to non adjacent vertices [GH97].

Most mesh simplification algorithms rely on the edge collapse operation [LT98, Hop99], and other operations are more rarely used.

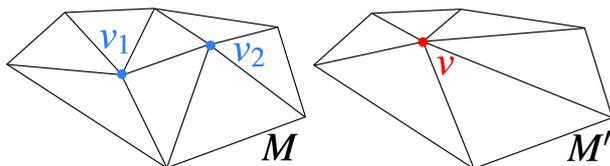


Figure 3.2: *The edge collapse coarsening operation. Various methods have been proposed for placing the vertex v such that the geometric error is minimized [HDD*93, GH97, LT98].*

Note that some mesh simplification algorithms rely on local operation that do not coarsen the mesh, for instance the edge split and the edge swap operations [HDD*93]. It has been observed that these operations bring little benefits in general and that coarsening operations such as the edge collapse are sufficient [Hop96].

3.1.3 Quad mesh simplification

More recently, some authors addressed the problem of simplifying quad meshes (*i.e.* meshes made of quadrilateral faces) by using only quads in low-resolution geometry. Quad mesh simplification is more difficult than triangle mesh simplification, because it is rather difficult to find local coarsening operations that generate “good quads”, *i.e.* almost flat and nicely shaped quads with vertices of valence 4, without introducing triangles in the neighborhood. Various quad coarsening operations have been proposed and combined in previous work, including:

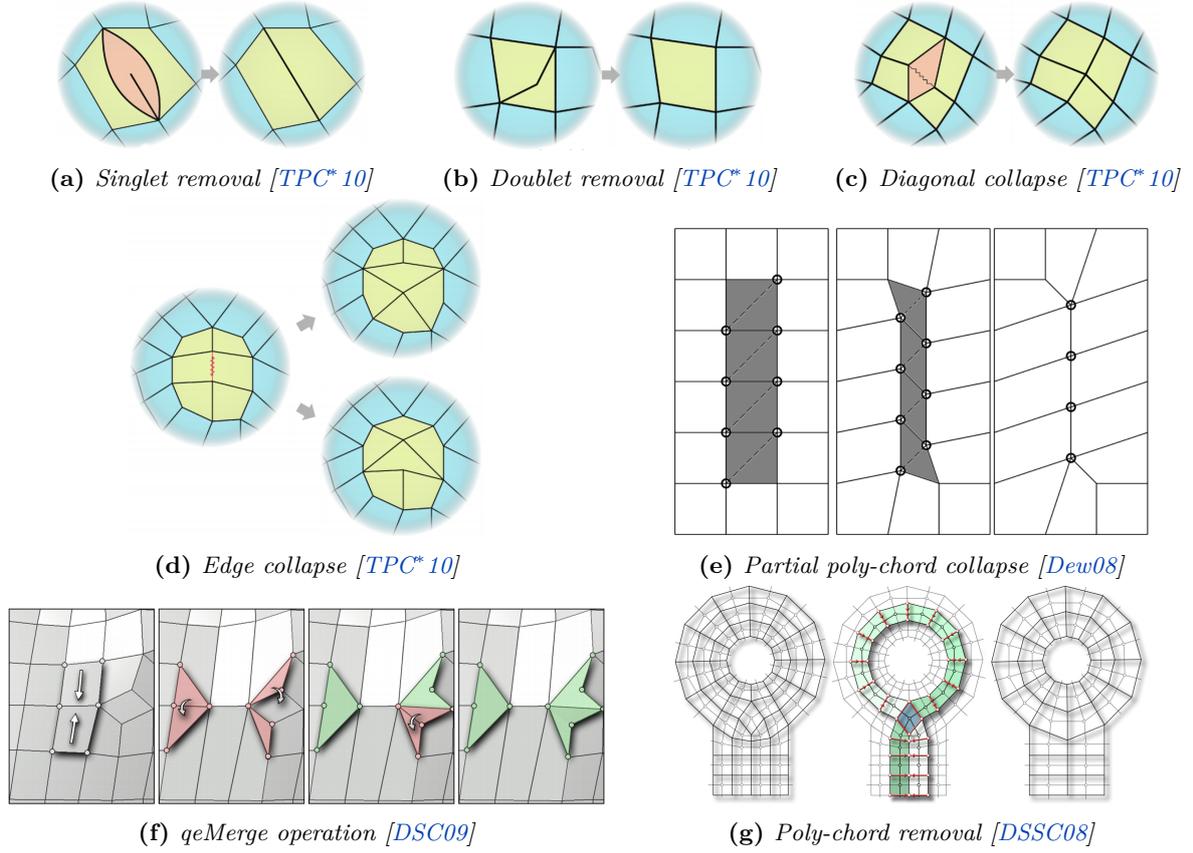


Figure 3.3: *Examples of coarsening operations for quad mesh simplification.*

- Local coarsening operations that only affect a few quads:
 - The diagonal collapse [TPC*10], also known as quad-close [Kin97], quad-collapse [DSSC08], quadvertex merge [DSC09], diamond or face close operation [Dew08] (Fig. 3.3c).
 - The singlet and doublet removal [TPC*10] (Fig. 3.3a and 3.3b).
 - The edge collapse for quads [TPC*10] (Fig. 3.3d).
 - The *qeMerge* operation [DSC09] (Fig. 3.3f).
- Less local coarsening operations:
 - The ring collapse [tBPHJ02], later generalized as poly-chord collapse [DSSC08,SBS08] (Fig. 3.3g).
 - The partial ring collapse [Dew08], (Fig. 3.3e).

Quad mesh simplification is discussed with more detail in section 8.4, in which we rely on a type of poly-chord collapse operation for prefiltering quad meshes and their materials.

3.1.4 Quantifying and minimizing geometric errors

Several authors proposed quality metrics for quantifying the geometric error of mesh simplification algorithms. These metrics are used to optimize the quality of the mesh in budget-based methods, and to control and stop the simplification in fidelity-based methods.

Maximum distances. A simple metric of the quality of a mesh consists in defining a distance from a surface M to a surface M' as the maximum of the distance between a point of M and the surface M' .

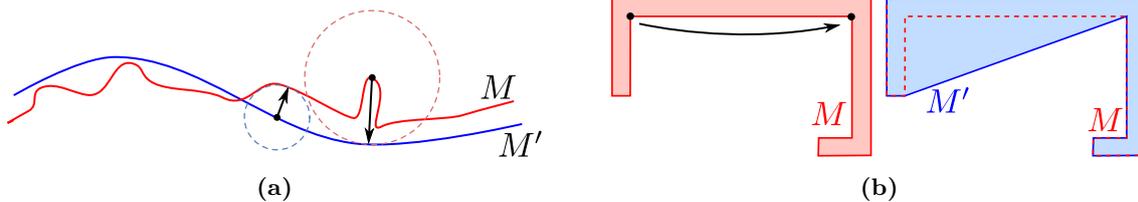


Figure 3.4: The surface-to-surface distance defined as $d_1(M, M') = \max_{x \in M} d(x, M')$ is not symmetric: $d_1(M, M') \neq d_1(M', M)$ in general, as shown in (a). Using this distance is insufficient for mesh simplification. For instance, in example (b), $d_1(M, M') = 0$ even if M' introduces a geometric error. It is often more accurate to rely on a two-sided distance, for example a Hausdorff distance defined as $d_h(M, M') = \max(d(M, M'), d(M', M))$.

This *one-sided* surface-to-surface distance writes

$$d_1(M, M') = \max_{x \in M} d(x, M')$$

with

$$d(x, M') = \min_{y \in M'} \|x - y\|_2.$$

where $\|\cdot\|_2$ denotes the usual Euclidean norm. Using this definition, the distances $d_1(M, M')$ and $d_1(M', M)$ are not equal, as illustrated in Fig. 3.4: even if all the points of M are close to the surface M' , some points of M' could be at a larger distance from M . In a mesh simplification context, it is generally important that both $d_1(M, M')$ and $d_1(M', M)$ remain small. Therefore, a two-sided distance such as the *Hausdorff distance* provides a better information about the quality of a simplification [KLS96, Ros97, ASCE02, LWC*02]:

$$d_h(M, M') = \max(d_1(M, M'), d_1(M', M)).$$

Various algorithms have been designed based on such one-sided [CVM*96] or two-sided distances [Gué99]. Bounding the surface-to-surface distance in object space allows for bounding the geometric error in screen space, for example ensuring that the simplified surface does not deviate of more than one pixel.

Mean distances. Cignoni et al. introduced the Metro tool [CRS98] for evaluating the geometric accuracy of LODs, based on the integral of the point-to-surface distance on the surface:

$$E(M, M') = \frac{1}{|M|} \iint_{p \in M} d(p, M') ds$$

where $|M|$ denotes the area of M . For mesh optimization, for example for optimizing vertex positions in the edge collapse framework [HDD*93], is it convenient to have similar metrics based on some kind of mean distance between two surfaces, so that the minimization problem can be solved locally with a few iterations [HDD*93]. Hoppe et al. [HDD*93] proposed a one-sided error metric based on square distances between points on the high-resolution mesh and the simplified mesh:

$$E(X, M') = \sum_n d(x_i, M')^2$$

with $X \in (\mathbb{R}^3)^n$ a set of 3D points $x_i \in M$ and M' the simplified surface. Aspert et al. [ASCE02] write a similar *root mean square error* with an integral:

$$E(M, M') = \sqrt{\frac{1}{|M|} \iint_{p \in M} d(p, M')^2 ds}.$$

Two-sided versions of these error metrics have also been used [GH97, OVBP11]. We rely on similar metrics for optimizing vertex positions in section 8.2.

Quadric error metrics. Quadric error metrics (QEM) were introduced by Garland et al. [GH97] and used for ordering collapse operations and optimizing vertex positions. The error associated with a vertex position is a weighted sum of square distances between this point and the planes that contain the neighboring triangles before the collapse operation. This error can be written $E(v) = v^t Q v$ with $v = (x, y, z, 1)^t$ the vertex position and Q a 4×4 matrix. Indeed, the distance between a point (x, y, z) and a plane with equation $ax + by + cz + d = 0$ (with $a^2 + b^2 + c^2 = 1$) writes $v^t P = ax + by + cz + d$ with $P = (a, b, c, d)^t$. The square distance writes

$$(ax + by + cz + d)^2 = (v^t P)^2 = v^t P P^t v = v^t Q_p v \quad \text{with} \quad Q_p = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}.$$

Then the sum of square distances to several planes writes $v^t Q v$ with Q a sum of matrices Q_p . Given a matrix Q , optimizing v consists in finding the position v that minimizes $v^t Q v$, which is a linear problem [GH97]. The quadric error metric must be seen as an approximation of the geometric error. Indeed, what really matters is the distance between v and the input geometry, which is only approximated by distances between v and a set of planes. Quadric error metrics have been used in many applications because they are memory efficient, they can be implemented easily, they are fast and they improve the quality compared to naïve methods.

It has been observed that placing vertices in order to preserve the volume of the geometry increases the accuracy of mesh simplification methods [LT98], compared to the original QEM and naïve placement strategies. Several authors proposed volume-preserving algorithms [Gué99, LT98, Hop99]. We found that volume-preserving strategies lack robustness when simplifying very thin geometry, as discussed in section 8.2. We argue that volume preservation is not the proper requirement for most applications (including ours) but a convenient metric for approximating two-sided geometric errors. In section 8.2, we show how to minimize directly a two-sided geometric error, which is more robust than volume preserving methods in some cases.

Memoryless simplification. The geometric error that we want to minimize is the error between the input surface and the simplified surface. Some of the first geometric simplification algorithms such as the one proposed by Hoppe [Hop96] approximate such errors during the optimization process. Lindstrom and Turk [LT98] have shown that it is sufficient to minimize the geometric *incremental error* locally after each collapse, between the newly created geometry and the geometry before the collapse operation. This strategy has been called *memoryless simplification* because the input high-resolution surface is not used for computing errors. Hoppe later proposed a new version of the QEM and provides some insight about the benefits of memoryless simplification [Hop99].

Using screen-space errors. Lindstrom and Turk [LT00] relied on QEM for placing vertices but they evaluate the cost of collapse operations in image space, by rendering the surface before and after the collapse from different points of view. This metric is appropriate for applications where only the visual fidelity matters but mesh simplification with this approach is more difficult and light-dependent.

3.1.5 Taking into account reflectance attributes in quality metrics

The first mesh simplification algorithms only addressed the problem of simplifying the geometry. In the following years, several authors addressed the case of simplifying meshes with attributes while preserving their appearance. The attributes were mostly per-vertex or per-corner colors and normals [Hop96], and textured meshes with per-corner uv coordinates. Some quality metrics were proposed for taking into account both the geometric error and the attribute deviation during the simplification [BRS96, GH98, Hop96, EM99, Hop99], while Cohen et al. [CMO97, COM98] proposed a metric for the texture deviation.

These metrics are either used for restricting the simplification to non-destructive operations [BRS96], but also for estimating the best positions and attributes for the vertices after each edge collapse operations, in order to minimize both the geometric error and the attribute errors [GH98, Hop96, Hop99]. Optimizing

vertex attributes, for example using quadric error metrics [GH98,Hop99], can be seen as a form of material prefiltering. In the case of per-vertex attributes, for example, the color of a new vertex is obtained from the colors of the neighboring vertices. However, this method has limited accuracy for normals [Hop99] and has not been used for prefiltering normal distributions. We propose an alternative prefiltering technique for edge collapse operations in section 8.3 and we will show how to prefilter microfacet distributions in this context.

3.2 Other geometric level-of-detail representations

Meshes are the most used representation for rendering surfaces but many authors proposed alternative representations that have better properties for some applications. In particular, various alternative surface representations, such as surfels, point set surfaces and distance fields have several benefits for prefiltering high-frequency geometric details. In this section, we describe these representations and discuss their strengths and weaknesses for LOD rendering.

3.2.1 Surfels as LODs for global illumination

The word *surfel* is an abbreviation of *surface element*. Surfels have been initially used by Pfister et al. [PZvBG00] as an alternative rendering primitive to polygon meshes for interactive rendering. Each surfel is a sample of a surface, that is a 3D position, a normal, and some shading parameters. Pfister et al. use them for efficient interactive rendering of relatively complex surfaces, although the authors state that “surfels are not well suited to represent flat surfaces, such as walls or scene backgrounds, where large, textured polygons provide better image quality at lower rendering cost”. Surfels are related to point based rendering [AGP*04,Gro09], whose main goal is to render point sets without surface reconstruction.

Surfels were initially rendered using *splatting* techniques in interactive applications [PZvBG00]. In ray tracing applications, surfels rendered as discs lead to various artifacts such as holes in the surface and false self-shadowing [WS05]. Surfels have been used in production for rendering distant crowds for which LOD techniques were mandatory due to the large number of agents [Kan14].

Interestingly, surfels have received much more attention when physically-based global illumination computations became affordable for the animation and special effects industries [PH08]. Indeed, surfels often lack accuracy for the surfaces viewed directly, but they have been used for representing approximate versions of 3D scenes for computing indirect lighting [Bun05,PH08,Chr10,KTO11]. This technique is often referred to as *point-based global illumination*, or PBGI. Groups of surfels are approximated using spherical harmonics [PH08] or wavelets [WMB15], and the unstructured nature of surfels allows fast access to appropriate LODs of the geometry seen from each shading point.

The original PBGI algorithm is limited to diffuse global illumination, or reflections from rough materials at the cost of increased memory usage [WMB15]. Despite the interest of this noise-free and fast global illumination technique, the film industry progressively moved to path-traced global illumination [KFF*15].

3.2.2 Point set surfaces

Point set surfaces are implicit surfaces defined from unstructured point sets, *e.g.* points obtained from 3D scanning devices. They were introduced to computer graphics by Alexa et al. [ABCO*01] and improved in various other publications [AA04,AK04,GG07,Gro09]. They aim at defining smooth, manifold surfaces from a set of 3D points (often with their normals) without relying on surface reconstruction algorithms and mesh representations.

Point set surfaces can be used for rendering complex surfaces from little information (points without connectivity), as shown in Fig. 3.5. Moreover, high-frequency details can be very easily filtered by changing a parameter used in the projection operator that defines the surface [ABCO*01]. However, point set surfaces have not been used for LOD rendering, because ray tracing such surfaces is much less efficient

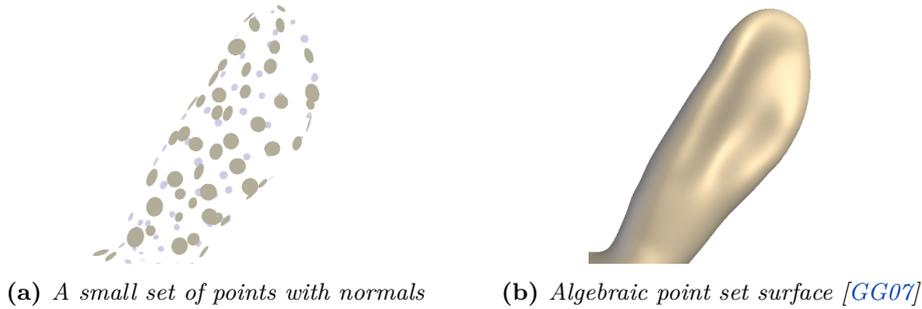


Figure 3.5: Example of point set surface defined from a small set of points with their normals, from the work of Guennebaud and Gross [GG07].

than rendering polygons [AA03, WS05], and because they are less flexible than polygons for representing complex geometry with a lot of sharp features. Conversion to meshes has been studied [SFS05] but point set surfaces as intermediary LOD representation bring few benefits if the high-resolution asset is a mesh.

3.2.3 Representing surfaces with volumetric data

Volume representations are convenient for LODs thanks to their structured and hierarchical nature, allowing for prefiltering 3D data in the spirit of texture mip mapping. In particular, volumes can be used for prefiltering high-frequency geometry into semi-transparent voxels (this is what we propose in chapter 9). In this chapter, we focus on volume representations used for representing simple enough surfaces.

Gobbetti et al. [GM05] proposed to store some geometry in a volume hierarchy, and to render the sub-resolution geometry of low-resolution voxel using simple OpenGL point primitives. A view-dependent shading is estimated for each point by casting rays on the input geometry in a preprocess step. Yoon et al. [YLM06] proposed to approximate sub-resolution geometry in low-resolution bounding boxes by a plane with material attributes. This strategy is actually related to surfels (Sec. 3.2.1) and it has the same limitations: visible holes on curved surfaces, false self-shadowing and popping artifacts at LOD transitions. This approach was later revisited and improved by Laine et al. [LK10a, LK10b]. Instead of using a single plane in their sparse voxel octrees, they store a pair of parallel planes (or *contour planes*) per voxel that envelop the geometry contained in the voxel. The orientation of the planes matches the orientation the approximated surface. At rendering, the slab defined by the two planes is intersected with the slabs of the parent voxels for improving the accuracy, as shown in Fig. 3.6. This representation is memory efficient, it is not subject to holes and it can be rendered very efficiently. However, the authors did not address the problem of seamless transitions, and this representation is not appropriate when the geometry inside a voxel is too complex.

Sparse voxel octrees were initially designed for efficient real time rendering of complex scenes but they have also been used for precomputing LODs of production assets [PMA14, PES15]. Prus et al. [PES15] built on the contour planes of Laine et al. and they prefilter materials relying on Burley’s BRDF [Bur12] whose parameters have a near linear impact on the visual aspect of the material, allowing for linear parameter prefiltering. Prus et al. also addressed the problem of normal prefiltering in Laine’s framework.

Heitz and Neyret [HN12] proposed an alternative representation to contour planes. Like Crassin et al. [CNS*11], they rely on cone octree traversal. Cones from the camera are divided into small cone elements, approximated by cylinders with equal diameter and length. Macrosurfaces are encoded in the volume representation using a signed distance and a normal per voxel. In each cone element volume data is interpolated for obtaining a prefiltered normal and a signed distance that define a plane which approximate the sub-voxel geometry. This representation brings several benefits compared to contour planes: it produces better results under magnification (that is, when voxels are larger than pixels), and it supports seamless LOD transitions with quadrilinear interpolation. One important assumption of their work is that the sub-resolution geometry in voxels is always a macrosurface, with a low macroscopic

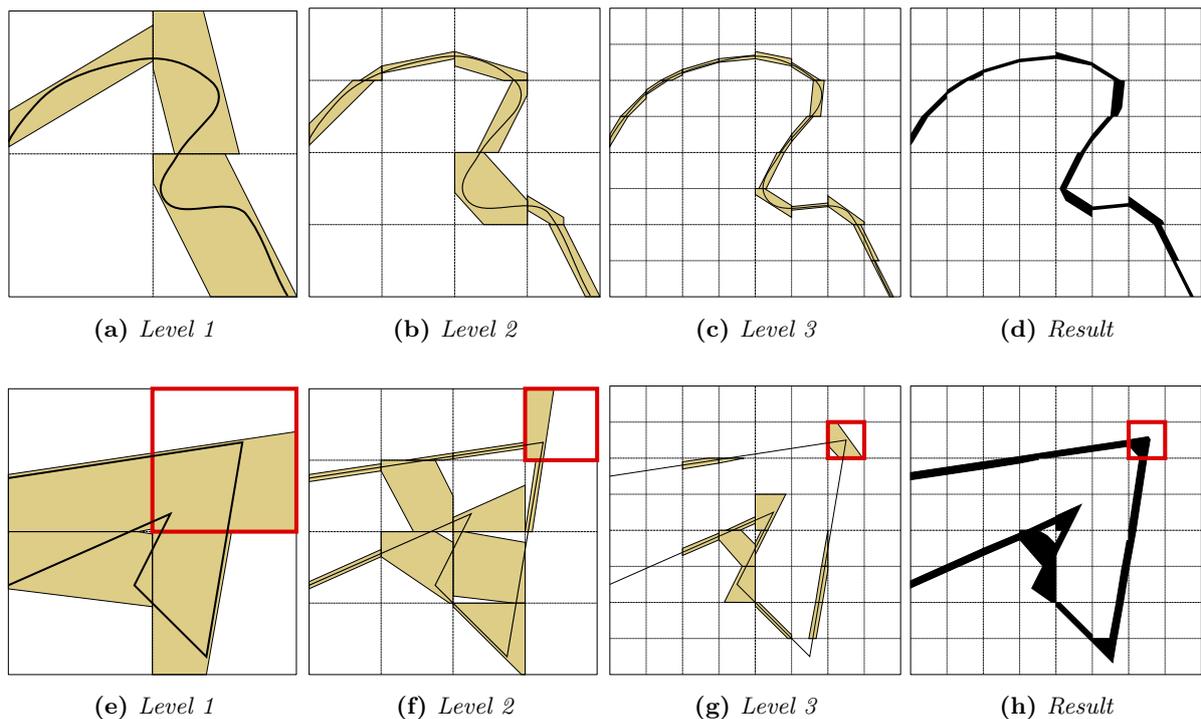


Figure 3.6: *Laine et al. [LK10a, LK10b] store two parallel planes per voxel for approximating the geometry at different scales. At rendering, the slabs of parent voxels are taken into account for improving the accuracy of the representation (d,h).*

curvature and belonging to a thick object (hypotheses H6 and H7 in Sec. 5.1 in [HN12]). The authors did not use their grid-based signed distance field in a ray tracing application although it could be rendered like other implicit surfaces with iterations to find ray intersections [Har96, WS05, Jam10b].

3.3 Subdivision, tessellation, displacement

Subdivision surfaces and displacement mapping are standard tools in the industry for modeling complex surfaces with a few base polygons and texture-based geometric details. These techniques support LODs by nature, although achieving watertight, artifact-free LODs with seamless transitions is not straightforward. We review some previous work that are useful for understanding the problem of prefiltering production assets (Chap. 6).

3.3.1 Subdivision surfaces

Parametric patches (Bézier patches, B-spline patches) are mappings from a 2D domain into 3D space, used in modeling for creating smooth surfaces from a small set of control points. Subdivision surfaces were introduced in the publications of Catmull and Clark [CC78] and Doo and Sabin [DS78] at a time when researchers were looking for ways to generalize spline modeling to arbitrary topology [DKT98]. Starting from a polygon mesh, a subdivision scheme is applied iteratively to refine the mesh, and the geometry converges to limit smooth surface (Fig. 3.7). Catmull-Clark subdivision surfaces can also be seen as parametric surfaces since they can be evaluated directly using Stam’s method [Sta98], instead of based on iterative subdivision.

Subdivision surfaces have huge benefits for modeling and rendering surfaces:

- They allow for modeling smooth shapes (the limit surface has C^2 continuity almost everywhere)

using a few control points. Control points only affect the geometry locally (this property is also called *compact support of local definition*). This facilitates authoring and computations.

- One can manipulate the coefficients of subdivision to achieve effects such as sharp creases or control the behavior of the boundary curves (we will discuss this in chapter 6).
- Subdivision surfaces can be efficiently animated by animating the base mesh.
- They naturally support levels of detail. Base meshes are typically subdivided until each polygon projects to approximately one pixel. The base meshes are often considered as good coarse LODs when base polygons themselves are sub-pixel. However, the base mesh is not the best approximation of the limit surface in general, since the geometry tends to shrink significantly during the first subdivisions.

Various subdivision schemes have been proposed, including Doo-Sabin [DS78], Catmull-Clark [CC78], Loop [Loo87], Butterfly [DLG90, ZSS97], Kobbelt [Kob96] and Midedge [PR97]. Catmull-Clark subdivision surfaces are the most used in production with quadrilateral base meshes. The limit surface defined by Catmull-Clark subdivision can be described by a collection of bicubic B-spline patches, with an infinity of such patches around extraordinary vertices [SRK*15].

Liktor and al. [LPD14] addressed the problem of adaptive subdivision of parametric surfaces, so that large and close patches are more subdivided than small or distant patches. Adaptive subdivision raises the problem of cracks between neighboring patches that do not have the same subdivision level. Liktor and al. addressed this problem based on the fractional tessellation introduced by Moreton [Mor01], and their method supports seamless transitions. They have used their method in the case of Catmull-Clark subdivision, relying on the approximations proposed by Loop et al. around irregular vertices [LSNCn09].

Once subdivided, the vertices can be further displaced using displacement maps or procedural functions to add high-frequency details to the surface. This is very commonly used for modeling organic shapes that have a smooth macroscopic appearance with small irregularities (skin, bark, leaves, etc.) in the animation and special effects industries. Efficient subdivision surfaces for real-time applications on the GPU is a more recent research problem [LS08, LSNCn09, NLMD12, SRK*15, BFK*16] and subdivision surfaces are not widely used currently in the game industry.

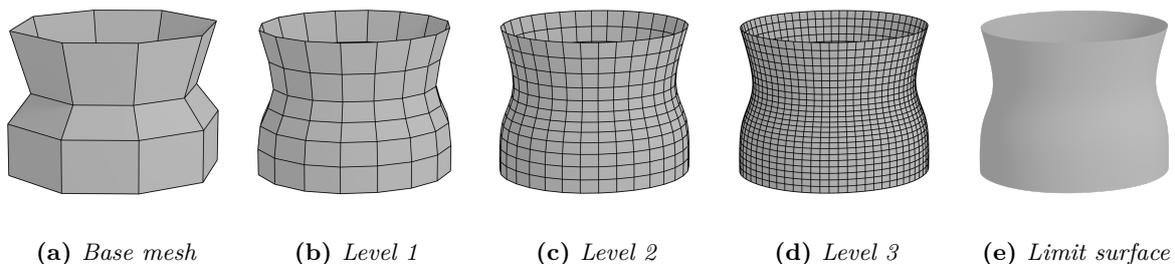


Figure 3.7: *Catmull-Clark subdivision surfaces* [CC78]

3.3.2 Displacement mapping

In 1978, Blinn proposed to perturb surface normals using a wrinkle function [Bli78] to mimic the appearance of a high-resolution surface without increasing the geometric complexity. This led Cook [Coo84] to develop displacement mapping to render more realistic silhouettes. Displacement mapping is a standard tool in the animation and special effect industry because it produces accurate silhouettes but also correct masking-shadowing effects a small scales. Base polygons are tessellated using either a simple flat tessellation algorithm of subdivision surfaces, and the resulting vertices are displaced in the direction of the normal based on scalar displacement maps, or displaced in arbitrary directions if vectorial displacement maps are used.

The high-frequency geometry described by displacement maps can be prefiltered in the parametric space, which is simpler than mesh-based geometry prefiltering. For example, normal distributions of displaced geometry can be efficiently prefiltered in texture space [DHI*13], given that the amplitude of the displacement is small enough with respect to the macroscopic curvature.

Seamless LODs for adaptively tessellated surfaces with displacement mapping has been recently studied when hardware tessellation became available on GPUs. On-the-fly hardware tessellation allows for rendering very detailed meshes much faster than traditional high-resolution meshes with similar complexity [NKF*16], and it is now widely used in computer games, as well as in interactive previews in the film industry.

Problems appear when displacement mapping is used in continuous LOD frameworks. The hardware tessellator generates vertices with their positions and texture coordinates in the parametric space, and fractional tessellation introduces swimming artifacts during transitions because the generated vertices are progressively sliding in texture space. Several authors recently addressed this problem for reducing or avoiding such artifacts [NL13, SPM*13, LBG16].

3.4 Summary of the chapter

In this chapter, we have presented previous work on:

- Mesh simplification methods based on small coarsening operations, driven by quality metrics that include geometric error, attribute and texture deviations, and screen space errors.
- Other surface representations (surfels, point set surfaces and voxel-based representations such as distance fields and plane contours) that can be used in LOD framework.
- Technique for modeling and prefiltering the geometry of highly detailed surfaces through adaptive subdivision and displacement.

All these methods are intended to prefilter the geometry at scales where it can be approximated locally by a piece of surface (polygon, surfels or smooth implicit surface). This assumption does not hold for assets with sub-resolution intricate geometry and the surface-based methods presented in this chapter do not address the problem of representing accurately semi-transparent appearances with correct occlusion. In the next chapters, we discuss existing techniques for prefiltering materials in texture space (Chap. 4) as well as methods for prefiltering sub-resolution intricate geometry when surface-based methods fail in preserving a correct macroscopic appearance (Chap. 5).

3.5 Résumé du chapitre

Dans ce chapitre, nous avons présenté plusieurs méthodes existantes :

- Des méthodes de simplification de maillage qui reposent sur de petites opérations locales de simplification, contrôlées par des métriques qui prennent en compte l'erreur géométrique introduite, la déviation d'attribut ou de coordonnées de texture, ou encore en prenant en compte des erreurs en espace image.
- Des représentations surfaciques alternatives aux maillages (surfels, point set surfaces et représentations par des grilles de voxels) qui peuvent être utilisées pour du rendu efficace avec niveaux de détail.
- Des techniques de modélisation et de filtrage de surfaces très détaillées, et en particulier les techniques de subdivision adaptative et de déplacement.

Toutes ces méthodes ont pour but de préfiltrer la géométrie à des échelles où elle peut être approximée localement par un morceau de surface. Ces approximations sont incorrectes dans le cas d'une géométrie complexe à l'échelle du préfiltrage, et les méthodes présentées dans ce chapitre ne permettent pas de traiter ce cas important. Dans les prochains chapitres, nous présentons les techniques existantes pour le préfiltrage des matériaux en espace texture (Chap. 4) ainsi que les méthodes de préfiltrage pour la géométrie complexe (Chap. 5).

Chapter 4

Previous work on surface-space prefiltering

In physically-based rendering techniques, the pixels of virtual camera models measure light intensity emitted and scattered in the scene. Each pixel value is defined by 4D integral of the incident light over a 2D spatial kernel around the pixel center and over a set of directions in the 2D spherical domain depending on the aperture of the camera model. Such integrals are expensive to compute. In Monte Carlo rendering, pixel integrals are estimated by casting rays that sample the 4D domain, leading to noise if the integrand has high frequencies (meaning that pixels “see” complex things).

In this chapter, we focus on the case where pixels only see a piece of surface, and we will show that with some assumptions it is possible to rewrite the pixel integral as a surface integral over the pixel footprint. Then, we can derive prefiltering methods in surface space that allow for prefiltering spatial frequencies without changing the pixel integral. In other words, we can simplify the appearance of surfaces so that less data needs to be loaded, stored and accessed at rendering, while decreasing the noise of the rendered image and without introducing visible errors. The need for surface prefiltering has been soon analysed and addressed in the 80s’ with the introduction of texture mip mapping [Wil83] and the idea of BRDF models as prefiltered representations of the distant appearance of complex surfaces [Kaj85, BM93]. Prefiltering in surface space is still an ongoing research problem, in particular for real-time applications where anti-aliasing methods are mandatory [OB10, DHI*13, KHPL16, XWZB17].

We first recall the derivation of prefiltering integrals from scene space to surface space based on previous work [BN12] and we recall the underlying assumptions of any surface prefiltering methods. We start with the simple case of pixel footprints covering flat surfaces, and then we describe the case of bumped surfaces with more complex self-shadowing and view-dependent visibility effects. Then, we present previous work on linear and non-linear surface prefiltering.

4.1 Physical units and notations

Physically-based rendering relies on radiometric units such as *irradiance* and *radiance* on which rely surface reflectance models. We briefly recall the main physical units used in computer graphics and we introduce some notations.

4.1.1 Directions and solid angles

Directions are fundamental in physically-based rendering. They are either defined as normalized 3D vectors, that is, points on the unit sphere (noted S^2), or using two angles in spherical coordinates. A *solid angle* is an area on the unit sphere that quantifies how much a set of directions occupies the spherical

domain. The unit of a solid angle is the steradian, noted *sr*. Steradians are to directions what square meters are to 2D points. The set of all the directions measures 4π sr (the area of the unit sphere), the hemisphere measures 2π sr. In our notations, we use ω for directions and $d\omega$ for infinitesimal solid angles around direction ω for integrals in the spherical domain.

4.1.2 Radiant flux

Light is an electromagnetic radiation that carries energy. An “amount of radiation” is called a *radiant flux* and its SI unit is the watt, that is a radiant energy (in joule) per second. In physically-based rendering, pixels values for each wavelength can be thought as measures of a radiant flux.

4.1.3 Irradiance

Irradiance is a radiometric unit that measures a radiant flux received by a surface, per unit area. Its SI unit is the watt per square meter ($\text{W}\cdot\text{m}^{-2}$), that is an energy per second and per square meter. Integrating an irradiance over a surface gives a radiant flux. *Radiosity* has the same unit and is used for measuring a flux *leaving* a surface.

4.1.4 Radiance

Radiance is the most used radiometric units in computer graphics theory. Unlike irradiance, it quantifies the light emitted, scattered or received by a surface in a particular direction ω . It is defined as the radiant flux per solid angle around the direction ω , and per square meter *orthogonally* to ω . We will note $L(p, \omega)$ or simply $L(\omega)$ the radiance in direction ω at a point p on a surface, and its unit is the watt per steradian per square meter ($\text{W}\cdot\text{sr}^{-1}\cdot\text{m}^{-2}$). Radiance can be integrated over the hemisphere to obtain an irradiance:

$$E(p) = \int L(p, \omega_i) \langle n \cdot \omega_i \rangle d\omega_i \quad (4.1.1)$$

where $\langle \cdot \rangle$ is the clamped dot product. The term $\langle n \cdot \omega_i \rangle$ comes from the fact that the radiance $L(p, \omega_i)$ is defined per square meter orthogonally to ω_i . The cosine of the angle between the surface normal n and the direction ω_i must be taken into account to obtain the received energy per square meter on the surface.

4.1.5 Bidirectional reflectance distribution function (BRDF)

A bidirectional reflectance distribution function (BRDF) characterizes how light is scattered at a point on a surface, more precisely how much incident light from direction ω_i is scattered in direction ω_o . We note this function $f(\omega_i, \omega_o)$ and it is defined as

$$f(\omega_i, \omega_o) = \frac{dL(\omega_o)}{L(\omega_i) \cos(\theta_i) d\omega_i}$$

with $L(\omega_i)$ the incident radiance in direction ω_i , and θ the angle between ω_i and the surface normal. The term $\cos(\theta_i)$ has the same meaning as in Eq. 4.1.1: it scales the incident radiance $L(\omega_i)$ to obtain an energy per square meter on the surface. The unit of BRDFs is sr^{-1} .

This definition of the BRDF may be unintuitive because of the derivative. The reason for this is that we want $f(\omega_i, \omega_o)$ to measure how much the incident radiance from direction ω_i is reflected in direction ω_o : the ratio of $L(\omega_o)$ and $L(\omega_i)$ would be insufficient since $L(\omega_o)$ takes into account the radiance scattered in direction ω_o from arbitrary directions, and not only from ω_i . The derivative avoids this problem: the BRDF measures a change of $L(\omega_o)$ due to a change of incident radiance $L(\omega_i)$.

Physically-based BRDFs in computer graphics have the following properties:

1. The BRDF must be positive: $f(\omega_i, \omega_o) \geq 0$. An incident radiance $L(\omega_i)$ cannot remove energy in scattering direction ω_o .
2. The BRDF must satisfy $f(\omega_i, \omega_o) = f(\omega_o, \omega_i)$: a surface that scatters light from direction ω_i to direction ω_o scatters light from ω_o to ω_i . This property is sometimes called the Helmholtz reciprocity principle (more generally, inverting a source and a receiver does not affect the light transport between them).
3. The scattered energy at the surface is not larger than the incident radiance. This writes: $\int_{S^2} f(\omega_o, \omega_i) \langle n \cdot \omega_i \rangle d\omega_i \leq 1$, with S^2 the unit sphere.

The refraction and diffuse transmission of light through a surface, for example at a glass-air interface or through a thin sheet of paper, can be characterized by a bidirectional transmittance distribution function (BTDF) over the lower hemisphere, the same way the BRDF is defined on the upper hemisphere. The union of the BRDF and the BTDF is called a bidirectional scattering distribution function (BSDF). We refer the reader to the work of Walter et al. [WMLT07] for more information about commonly used physically-based BSDF models based on microfacets.

4.1.6 The local illumination equation

The BRDF allows for writing a radiance of scattered light as an integral of the BRDF and the incident radiance:

$$L(\omega_o) = \int_{S^2} f(\omega_i, \omega_o) L(\omega_i) \langle \omega_i \cdot n \rangle d\omega_i. \quad (4.1.2)$$

This equation has been introduced to computer graphics in 1986 [ICG86, Kaj86] and is sometimes referred to as the *rendering equation*, although modern renderers simulate other kinds of scattering events such as volumetric scattering. This equation is also called the *local illumination equation* [BN12] and it is an important equation for deriving surface prefiltering methods.

4.2 Prefiltering flat surfaces

In this section, we consider the case of a locally flat surface seen by a pixel of a virtual camera, and we recall the hypotheses under which the pixel integral can be rewritten as an integral of radiance over the *pixel footprint* on the surface. The case of flat surfaces allows for understanding the validity domain of all surface prefiltering methods.

4.2.1 Problem statement

Let's consider a locally flat surface P with normal n , rendered as an opaque surface mapped with a spatially varying BRDF. This surface is seen by a virtual camera, and we are interested in the value of a pixel that only receives radiance from this surface (there are no other occluders in the scene nor volumetric scattering).

Flat surface seen by a pinhole camera. For this derivation, we focus on the case of a pinhole camera, that is a virtual camera with infinitely small aperture, so that every point in the image plane only receives radiance from a single direction. The 4D pixel integral then reduces to a 2D spatial integral of the radiance on a spatial kernel around the pixel center. This kernel for integrating the radiance $L(x, \omega)$ is illustrated in Fig. 4.1 in the virtual image plane. It can be a simple box filter or a smoother filter with better anti-aliasing properties, for instance a Gaussian filter or a Mitchell-Netravali filter [MN88].

For each point x on the virtual image plane, the only radiance that contributes to the image is the radiance in direction ω from x to the pinhole. Therefore, pixel values can be written either as integrals

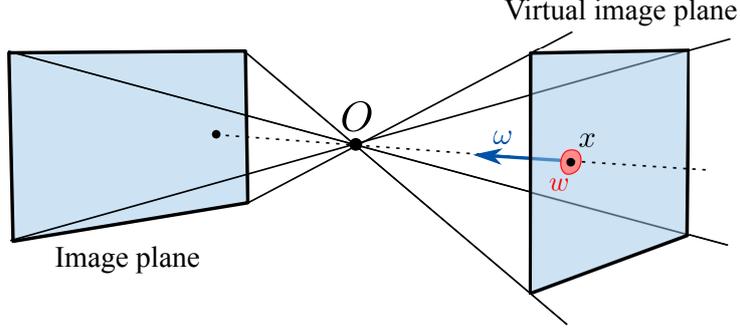


Figure 4.1: A pinhole camera model. Each pixel is defined as an integral of the radiance $L(x, \omega)$ on a kernel w , either in the image plane or in the virtual image plane.

in screen space or in the spherical domain:

$$I = \int_{S^2} w(\omega) L(O, \omega) d\omega \quad (4.2.1)$$

with $L(O, \omega)$ the radiance at the pinhole in direction ω , I a mean radiance with unit $\text{W}\cdot\text{m}^{-2}$, and w the pixel filter in the space of directions such that

$$\int_{S^2} w(\omega) d\omega = 1.$$

Because the pixel sees directly the flat surface P , the pixel integral (Eq. 4.2.1) can be written as an integral of the radiance reflected or emitted from the surface, in the pixel footprint on the surface. This is illustrated in Fig. 4.2.

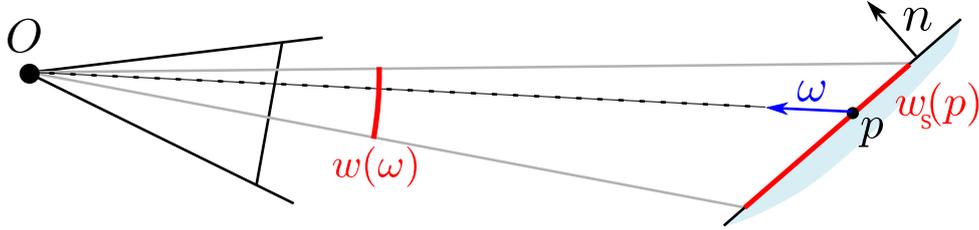


Figure 4.2: A pixel of a pinhole camera, characterized by an angular kernel w , seeing a flat surface with normal n . The angular kernel can be replaced by a spatial kernel $w_S(p)$ of the pixel footprint on the surface.

Each direction w corresponds to a point p in the pixel footprint on the surface, and we have the relation $d\omega = \frac{\omega \cdot n}{r^2} dp$ [BN12], with r the distance between the surface and the pinhole $\|O - p\|$. We can rewrite the pixel integral (Eq. 4.2.1) as follows:

$$I = \int_{S^2} w(\omega) L(O, \omega) d\omega = \int_P w_S(p) L(p, \omega) dp$$

with $w_S(p) = w(\omega) \frac{\omega \cdot n}{r^2}$ the spatial kernel on the surface. We have

$$\int_P w_S(p) dp = \int_P w(\omega) \frac{\omega \cdot n}{r^2} dp = \int_{S^2} w(\omega) d\omega = 1.$$

Local illumination equation. Using the local illumination equation (Eq. 4.1.2) in the pixel integral, I becomes a double integral over the spatially varying BRDF in the pixel footprint and the incident

illumination in the spherical domain:

$$I = \int_P w_S(p) \int_{S^2} f(p, \omega, \omega_i) L(p, \omega_i) \langle \omega_i \cdot n \rangle d\omega_i dp \quad (4.2.2)$$

Surface prefiltering. A common assumption at this stage is to assume that the incident radiance $L(p, \omega_i)$ is uniform over the pixel footprint (this is sometimes called the *far-field approximation*). This assumption is always used in prefiltering methods, and this is why it is always possible to claim the inaccuracy of LODs in the case of laser-like light sources, which are out of the validity domain of prefiltering methods. Using the far-field assumption, $L(p, \omega_i)$ becomes $L(\omega_i)$ in the equation. Moreover, the direction ω towards the pinhole can be considered constant in the pixel footprint, given that the camera is far enough, and that the pixel angular footprint is small. Using these two assumptions, we can rewrite I as:

$$I = \int_{S^2} \left(\int_P w_S(p) f(p, \omega, \omega_i) dp \right) L(\omega_i) \langle \omega_i \cdot n \rangle d\omega_i.$$

This is the *surface prefiltering equation*: it means that we can prefilter the spatially varying BRDF over the pixel footprint without changing the pixel value. We can define a new BRDF f_F as follows:

$$f_F(\omega, \omega_i) = \int_{S^2} w_S(p) f(p, \omega, \omega_i) dp \quad (4.2.3)$$

so that the pixel integral becomes

$$I = \int_{S^2} f_F(\omega, \omega_i) L(\omega_i) \langle \omega_i \cdot n \rangle d\omega_i.$$

As discussed in the introduction, prefiltering the surface reflectance functions brings several benefits:

- Evaluating the prefiltered BRDF f_F can be much cheaper than integrating all the BRDFs f : the 4D integral in Eq. 4.2.2 reduces to a 2D integral (since the 2D spatial integral is precomputed). In practice, this means less noise and faster rendering.
- The prefiltered BRDF f_F can sometimes be represented accurately with much less data than the set of input BRDFs, because the appearance that emerges from sub-resolution details often have a lower complexity than the details themselves (at least the spatial information can be removed). This means that rendering the prefiltered BRDF requires less memory and allows fast access to surface parameters at shading time.

Domain of validity. This derivation has been written for the case of locally flat surfaces seen by a pinhole camera with the far-field approximation. In practice, the prefiltering method described above is also valid (although approximate) for surfaces with low curvature, with other virtual camera models and additional non-pathological occluders in the scene. Moreover, in surface prefiltering techniques, the integral over the pixel footprint is always an approximation of the real footprint.

4.2.2 Prefiltering strategies

General approaches vs BRDF models. Given Eq. 4.2.3, the main goal of prefiltering methods is to find a good way to represent f_F in order to reduce the memory footprint while ensuring efficient evaluation. A very general strategy is to represent f_F using a multidimensional array. Indeed, BRDFs are functions over a 4D domain and they can be stored as 4D arrays of values. The dimension can be reduced to 3 for isotropic BRDFs (that is, BRDFs whose lobes do not depend on the azimuthal angle of the incident direction). When functions f_F are themselves spatially varying, this adds two dimensions to the arrays. Such 5D or 6D arrays are called Bidirectional Texture Functions (BTF) [DvGNK99], and they are equivalent to a set of photos of the surface for various view and light directions. Because BTFs directly represent reflectance values, they can be prefiltered very easily using simple linear combinations. Despite its generality, this strategy is not the most used for prefiltering surfaces because of its huge

memory requirements, in particular for high frequency BRDFs that require a fine sampling of the 4D or 3D angular domain. Another very general way of representing BRDFs is to compute their projections onto any basis of the 4D domain, for example using wavelets or spherical harmonics. This strategy has also significant memory requirements if BRDFs have high angular frequencies.

BRDFs are more efficiently represented using *parametric BRDF models* with a few reflectance parameters such as albedo (color), roughness, direction of anisotropy or index of refraction. Once a model is chosen for representing the prefiltered BRDF f_F , the prefiltering problem reduces to estimating the best set of parameters X_F such that

$$f_F(X_F, \omega, \omega_i) \simeq \int_P w_t(p) f(p, \omega, \omega_i) dp.$$

A common method is to use the same BRDF model for f_F and the input BRDFs f :

$$f_F(X_F, \omega, \omega_i) = f(X_F, \omega, \omega_i) \simeq \int_P w_S(p) f(X(p), \omega, \omega_i) dp$$

so that the parameters X_F can be estimated directly from the parameters of the input BRDFs $X(p)$.

Linear prefiltering of BRDF parameters. Some BRDFs parameters can be *linearly prefiltered* if they have a linear relation with the surface reflectance. In this case, the prefiltering problem reduces to

$$X_F = \int_P w_S(p) X(p) dp$$

or, in the case of textured surfaces,

$$X_F = \sum_{t \in T} w_t X_t$$

with T the set of texels in the footprint, w_t a weight for each texel and X_t the BRDF parameters of this texel.

For example, when the surface is assigned a Lambertian BRDF with spatially varying color $c(p)$, the prefiltering integral writes:

$$f_F(\omega, \omega_i) = \int_P w_t(p) f(p, \omega, \omega_i) dp = \int_P w_S(p) \frac{c(p)}{\pi} dp = \frac{\int_P w_S(p) c(p) dp}{\pi}$$

which is another Lambertian BRDF.

Linear prefiltering of microfacet distributions. Another example is the case of microfacet normal distributions in microfacet-based BRDF models [WMLT07, Hei14b]:

$$f(\omega_i, \omega_o) = \int \left| \frac{\omega_i \cdot \omega_m}{\omega_i \cdot n} \right| f^m(\omega_i, \omega_o, \omega_m) \left| \frac{\omega_o \cdot \omega_m}{\omega_o \cdot n} \right| G(\omega_i, \omega_o, \omega_m) D(\omega_m) d\omega_m$$

where D is the microfacet normal distribution, G is a masking-shadowing term that depends on the distribution D , $f^m(\omega_i, \omega_o, \omega_m)$ is the micro-BRDF on a microflake with normal ω_m . Assuming that the micro-BRDF is the same for all the microfacets of the surface, and that only the microfacet distribution D is varying on the surface, the prefiltering equation rewrites:

$$\int_P w_S(p) f(p, \omega, \omega_i) dp = \int_P w_S(p) \int \left| \frac{\omega_i \cdot \omega_m}{\omega_i \cdot n} \right| f^m(\omega_i, \omega_o, \omega_m) \left| \frac{\omega_o \cdot \omega_m}{\omega_o \cdot n} \right| G(\omega_i, \omega_o, \omega_m) D(\omega_m) d\omega_m dp \quad (4.2.4)$$

$$= \int \left| \frac{\omega_i \cdot \omega_m}{\omega_i \cdot n} \right| f^m(\omega_i, \omega_o, \omega_m) \left| \frac{\omega_o \cdot \omega_m}{\omega_o \cdot n} \right| \left(\int_P w_S(p) G(\omega_i, \omega_o, \omega_m) D(\omega_m) dp \right) d\omega_m. \quad (4.2.5)$$

This means that it is valid to linearly prefilter the product $G(\omega_i, \omega_o, \omega_m) D(\omega_m)$ in the pixel footprint. In practice, the masking-shadowing term G is neglected (since it has essentially an effect at grazing angles) or

considered roughly independent of the microflake normal distribution D , so that the prefiltering equation reduces to the prefiltering of the microflake distribution alone:

$$D_F(\omega) = \int_P w_S(p) D(\omega_m) dp.$$

Many authors rely on this approximation or a similar one [BN12]. Dupuy, Heitz et al. [DHI*13, Hei14a] showed that this approximation is accurate in the case of Gaussian surfaces.

Non-linear parameters. In some cases, BRDFs parameters do not have a linear relation to the surface reflectance and cannot be prefiltered linearly. For example, the roughness parameter of the Blinn-Phong BRDF [Bli77] cannot be prefiltered linearly. For a more complex example, let's consider a Lambertian surface with a spatially varying scalar parameter $s(p)$, and such that the color of the surface at a point p is obtained through an arbitrary colormap $C(s(p))$ with $C: \mathbb{R} \mapsto \mathbb{R}^3$ [HNPN13]. Then the prefiltering equation writes

$$f_F(\omega, \omega_i) = \int_P w_S(p) \frac{C(s(p))}{\pi} dp = \frac{\int_P w_S(p) C(s(p)) dp}{\pi}.$$

Because

$$\int_P w_S(p) C(s(p)) dp \neq C\left(\int_P w_S(p) s(p) dp\right)$$

in general, it is not possible to prefilter linearly the parameters $s(p)$.

Bruneton and Neyret [BN12] listed various general strategies for obtaining parameters that can be prefiltered linearly from non-linear parameters. Let's consider a function $f(X(p), Y)$ with non-linear and spatially-varying parameters $(X(p))$, so that the prefiltered function $f_F(Y)$ cannot be obtained directly:

$$f_F(Y) = \int w(p) f(X(p), Y) dp \neq f\left(\int w(p) X(p) dp, Y\right).$$

In this situation, it is possible to obtain $f_F(Y)$ through linear prefiltering using one of the following strategy:

- When the function f is a probability density function, it is sometimes well characterized by some its first moments. For example, in the case $Y \in R$:

$$\left[\int f(X(p), Y) \quad \int Y f(X(p), Y) \quad \int Y^2 f(X(p), Y) \right]$$

or in the case of $Y = (y_1, y_2) \in R^2$:

$$\left[\begin{array}{ccc} \int f(X(p), Y) & \int y_1 f(X(p), Y) & \int y_1^2 f(X(p), Y) \\ \int y_2 f(X(p), Y) & & \int y_2^2 f(X(p), Y) \\ & \int y_1 y_2 f(X(p), Y) & \end{array} \right].$$

The important idea is that moments of $f_F(Y)$ can be obtained by prefiltering linearly the moments of the base functions $f(X(p), Y)$. For example:

$$\int Y f_F(Y) dY = \int Y \int w(p) f(X(p), Y) dp dY = \int w(p) \left(\int Y f(X(p), Y) dY \right) dp$$

Therefore, if $f_F(Y)$ is well characterized by its moments, a simple prefiltering can consist of linearly prefiltering the moments of the base functions $f(X(p), Y)$. This is the core idea of the LEAN mapping technique for prefiltering normal distribution [OB10], as discussed later. This technique is mostly used for prefiltering Gaussian distributions that are entirely characterized by the first and second order moments.

- Another technique consists in projecting the base function $f(X(p), Y)$ onto a basis of functions $B_i(Y)$:

$$f(X(p), Y) = \sum a_i(p) B_i(Y)$$

Then the coefficients $a_i(p)$ can be prefiltered:

$$\int w_S(p) f(X(p), Y) dp = \int w_S(p) \left(\sum a_i(p) B_i(Y) \right) dp = \sum \left(\int w_S(p) a_i(p) dp \right) B_i(Y).$$

- Similarly, it is possible to write $f(X(p), Y)$ with a spanning set of functions that do not form a basis. This is the strategy adopted by multi-lobe approaches, as discussed later (Sec. 4.4.2). Such methods usually require to store less coefficients but prefiltering is more difficult if the number of coefficients has to remain constant at all scales [Fou92a, TLQ*05, TLQ*08].
- It is possible to rewrite $f_F(Y)$ as an integral over a distribution of parameters $\phi(x)$:

$$f_F(Y) = \int w(p) f(X(p), Y) dp = \int_{\chi} f(x, Y) \phi(x) dx$$

with

$$\int_{\chi} \phi(x) dx = 1$$

and χ the space of the parameters. If it is possible to efficiently evaluate the integral of the product $f(x, Y)\phi(x)$, then a simple prefiltering method consists in linearly prefiltering the distribution of parameters. This strategy has been used by several authors, including Heitz et al. [HNPN13, HNPN14] for prefiltering color mapped textures and surfaces.

4.2.3 Correlation between BRDF parameters

In some cases, the base BRDFs depend on several parameters that have a linear relation to the appearance and that are correlated to each other. An example is given in Fig. 4.3c and 4.3d in the case of a surface with correlated microflake distributions and albedos (colors). In this case, it is incorrect to linearly prefilter each type of parameter independently of the others, because of the correlations (Fig. 4.3). One can find many other cases of correlated parameters.

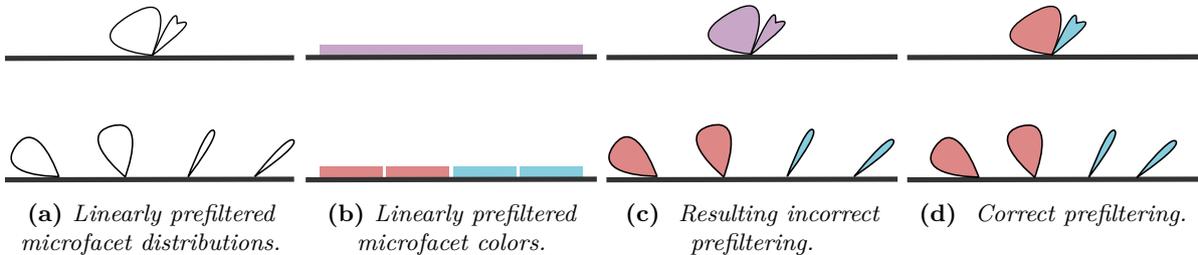


Figure 4.3: *Bottom:* Input surface with spatially-varying attributes. *Top:* Prefiltered surface. When surface attributes are correlated (here, microfacet distributions (a) and microfacet albedos (b)), it is incorrect to prefilter linearly each type of parameters independently of the others (c).

The absence of correlation is a common hypothesis in prefiltering methods [BN12], but some methods support some types of correlation as discussed in section 4.4.3.

4.3 Prefiltering rough surfaces

In the last section, we have shown that it is possible to prefilter the appearance of flat surfaces without changing pixel values. However, most rendered surfaces in computer graphics are not perfectly flat. In particular, surfaces are commonly tessellated and perturbed using displacement maps in order to add high-frequency details to a scene. In this section, we discuss the problems raised by the prefiltering of non-flat surfaces.

4.3.1 Problem statement

As in the previous section, we consider the case of a pinhole camera with a pixel footprint covering a surface, but we do not assume that the surface is flat. Because of surface perturbations, the normal is varying in the pixel footprint and some parts of the surface can be hidden. The pixel integral now rewrites as an integral over the *visible* part of the surface in the pixel footprint (Fig. 4.4).

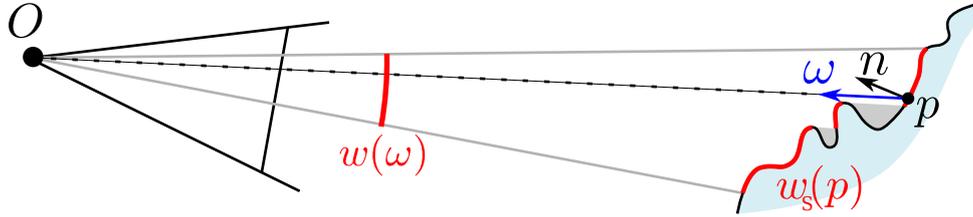


Figure 4.4: A pixel of a pinhole camera seeing a surface. Because the surface is not flat, the normal is varying in the pixel footprint and some parts can be hidden.

The pixel integral now writes

$$I = \int_P w_S(p) L(p, \omega) dp$$

with $w_S(p)$ taking into account the direct visibility of a point p from the camera, $V(p, \omega)$, and the spatially varying normal:

$$w_S(p) = w(\omega) V(p, \omega) \frac{n(p) \cdot \omega}{r^2}.$$

As before, and thanks to the visibility function, we have

$$\int_P w_S(p) dp = \int_P w(\omega) V(p, \omega) \frac{n(p) \cdot \omega}{r^2} dp = 1.$$

Using the local illumination equation, we obtain

$$I = \int_{\omega} \left(\int_S w'_S(p) f(p, \omega, \omega_i) \langle n(p) \cdot \omega_i \rangle dp \right) L(\omega_i) d\omega_i. \quad (4.3.1)$$

4.3.2 Useful hypotheses

At this stage, we can see from Eq. 4.3.1 that prefiltering arbitrary surfaces in surface space is not as simple as in the case of a flat surface: the integrand of the spatial integral contains a visibility term $V(p, \omega)$ and the term $n(p) \cdot \omega$ that are both view dependent. In other words, the contribution of points p to the radiance directly seen by the pixel is view-dependent, so that the surface must be prefiltered in a different way for each viewing direction, contrary to the case of the flat surface in which a single prefiltered BRDF can be used for all directions ω .

Nevertheless, with more assumptions, including the hypothesis of the absence of correlation between the visibility term and the BRDFs [BN12] or specific surface statistics [HNPN13, HNPN14], it is possible to derive approximate prefiltering equations.

4.4 Previous work

4.4.1 Mip mapping

Williams [Wil83] proposed to prefilter images by downsampling them by a factor 2 in each dimension using a box filter. The set of downsampled images is stored as a *pyramid* of prefiltered maps, called *mip*

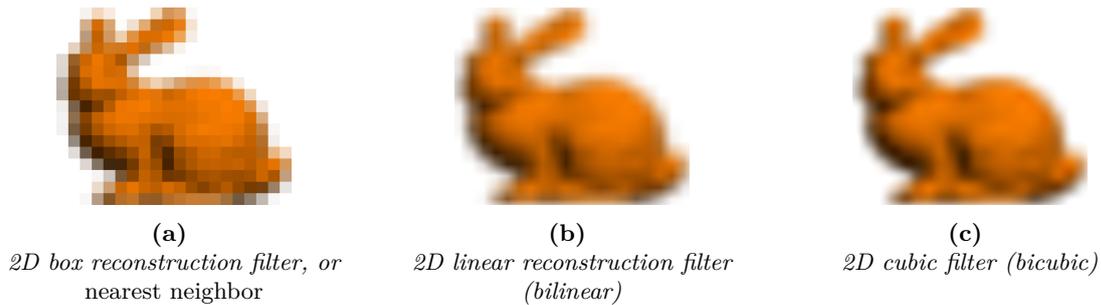


Figure 4.5: Common reconstruction methods for 2D images. Linear and cubic methods (b,c) offer higher quality than the box reconstruction filter (a).

maps. At render time, prefiltered values of the image are reconstructed using trilinear interpolation, *i.e.* a linear interpolation of two samples of two adjacent maps in the pyramid, themselves obtained with a bilinear interpolation of 4 values in the maps (Fig. 4.5). Therefore, trilinear interpolation requires the access to 8 texels of the prefiltered maps. Mip mapping or related methods are used in almost every rendering applications that have textured geometry.

The kernel used for generating the mip maps in the original algorithm was a square box filter. The data is further *blurred* by the trilinear interpolation when these maps are used for obtaining a sample. In the end, the prefiltering kernel of each sample is often quite different from the shape of the pixel footprint on the surface. Several authors introduced alternative techniques for a better matching between ellipsoid-like or trapezoid-like footprints and prefiltering kernels [Cro84, FF88, Hec89, SKS96, MPFJ99, BA05]. These techniques are often referred to as *anisotropic prefiltering techniques*.

As discussed in Sec. 4.2 and 4.3, mip mapping for color textures is only valid under some hypotheses: the underlying geometric surface should be flat enough or with statistical perturbations that are uncorrelated to colors, the effective mip mapping kernel should correspond to the pixel footprint on the surface and the colors must have a linear effect on the BRDFs. Mip mapping has later been used for prefiltering linearly other kinds of reflectance parameters, such as the moments of slope distributions [OB10, DHI*13].

4.4.2 Normal map prefiltering

Mip mapping relies on the linear prefiltering of texel values. Such linear operations must preserve the appearance at all scales: they must correspond to a valid prefiltering operation derived from the prefiltering equations (Eq. 4.2.3 and 4.3.1). With the success of bump mapping [Bli78], *i.e.* surfaces with perturbed shading normals for adding surface details without increasing the geometric complexity, prefiltering high-resolution bump maps became an important problem, in particular for real-time rendering. Contrary to color textures, bump maps (or the related normal maps) cannot be prefiltered by linearly averaging normals. Indeed, averaging and renormalizing normals results in a smooth macroscopic surface, which is a rather bad approximation of distant bumpy surfaces, which are better described by a distribution of normals or other kinds of roughness information. Various prefiltering methods for normal maps have been proposed for the multi-scale rendering of bumpy surfaces.

Normal map prefiltering as a convolution problem. Normal map prefiltering methods usually start from the case of a flat surface with a given *geometric* normal n (Sec. 4.2) and spatially varying *shading* normals $n_s(p)$, defined using a bump map, a normal map or procedural functions. The prefiltering equation reduces to the computation of a prefiltered BRDF f_F :

$$f_F(\omega_i, \omega_o) = \int w_S(p) f(p, R_{n_s}(\omega_i), R_{n_s}(\omega_o)) dp$$

where R_{n_s} is a rotation operator that rotates ω_i and ω_o in an appropriate shading frame aligned with the normal $n_s(p)$. In the case where the BRDF is not spatially varying, the prefiltered BRDF f_F

can be written as an integral of the base BRDF and a normalized normal distribution D in the pixel footprint [HSRG07]:

$$f_F(\omega_i, \omega_o) = \int f(R_\omega(\omega_i), R_\omega(\omega_o))D(\omega) d\omega. \quad (4.4.1)$$

Then this can be further simplified assuming that the base BRDF can be written as a function of ω_i ($f(\omega_i \cdot n)$ for Lambertian BRDFs) or ω_h , with

$$\omega_h = \frac{\omega_i + \omega_o}{|\omega_i + \omega_o|}$$

the half vector. With these particular BRDFs, equation 4.4.1 reduces to a convolution in the spherical domain. Note that BRDFs of the form $f(\omega_h, n)$ are related to microfacet-based BRDFs, because they are mainly characterized by the value of the microfacet distribution for the direction ω_h (although masking and shadowing terms also contribute, especially at grazing angles [WMLT07]). For this reason, prefiltering normal distributions or BRDFs in the space of the half-vector are very similar problems. Han et al. [HSRG07] compute this convolution in real time by projecting the BRDF onto zonal harmonics (a subset of spherical harmonics that are radially symmetric), and representing the normal distribution either using spherical harmonics or a set of von Mises-Fisher lobes (which are scalar function in the spherical domain that are symmetric about a central direction), inspired by the work of Fournier [Fou92b, Fou92a]. In the case where the base BRDF is a microfacet model, a microfacet normal distribution for f_F can be estimated by convolving the microfacet distribution of the base BRDF with the normal distribution. This idea can be found in the work of Toksvig [Tok05], where the BRDFs and normal distributions are written as gaussian functions in the 1D angular domain and convolved. Xu et al. [XWZB17] convolve the BRDF and the normal distribution, both approximated with von Mises-Fisher lobes.

Prefiltering the statistics of normal distributions. Convolution methods require to prefilter normal distributions. Han et al. [HSRG07] write the base normal distribution using spherical harmonics, so that the distribution can be prefiltered using a simple mip mapping of the spherical harmonics coefficients. They also propose an alternative multilobe representation that cannot be linearly prefiltered and that requires non-linear lobe fitting as discussed in the next paragraph. An alternative to spherical harmonics is the prefiltering of the statistics of normal distributions, which allows for prefiltering high-frequency distributions with very little data. Olano and North [ON97] proposed to represent normal distributions with 3D Gaussians restricted to a sphere. These distributions can then be prefiltered by averaging the centers of the 3D Gaussian distributions and estimating new standard deviations. As noted by Schilling [Sch97], this is related to the work of Neyret [Ney95, Ney98] in which normal distributions were encoded using ellipsoids that were combined into new ellipsoids for prefiltering. At the same period, Schilling [Sch97] proposed to encode normal distributions as 2D distributions of normal perturbations (which is equivalent to 2D Gaussian distributions in slope space). The distributions were characterized by the mean normal and a 2×2 covariance matrix. Schilling linearly combined the covariance matrices, which does not give exactly the correct statistics for the prefiltered distributions. A similar idea was reintroduced by Olano and Baker [OB10, Bak11] under the name *offcentered Beckmann distribution*. One of their main contribution is a new way to store the parameters of the distribution so that they can be prefiltered using linear mip mapping, and then used to reconstruct the covariance matrices. Dupuy, Heitz et al. [DHI*13] later revisited this representation. They used a microfacet-based BRDF model and they derived the missing masking-shadowing term for off-centered Beckmann distributions. All these methods are memory efficient but their accuracy is limited by the simplicity of the Gaussian distributions on which they rely.

Multilobe representations. The idea of *multilobe* techniques is to represent arbitrary BRDFs or normal distributions using a set of *base lobes*. Tan et al. [TLQ*05] use a set of 2D Gaussian functions for encoding normal distributions. Han et al. [HSRG07] used vMF lobes. The work of Fournier [Fou92a, Fou92b] is also related to multilobe representations of normal distributions since he represented prefiltered BRDFs using a set a Phong BRDFs. Similarly, Tan et al. [TLQ*08] rely on mixtures of gaussian and cosine BRDF lobes. Xu et al. [XWZB17] also use a set of vMF lobes for representing BRDFs in the half-vector space.

Multilobe methods allow for prefiltering more accurately complex normal distributions (compared to simple Gaussian distributions), but they raise two main challenges. First, approximating a set of

lobes with a smaller number of lobes requires non-linear and sometimes ill-conditioned optimization procedures [TLQ*05, HSRG07, TLQ*08]. Secondly, efficient seamless transitions are not trivial. Fournier [Fou92a, Fou92b] proposed to interpolate directly BRDF values, which requires to access and evaluate $8 \times N$ lobes for trilinear interpolation, N being the number of lobes in each texel. A less expensive method is to interpolate each lobe with the lobes in neighboring texels that have a similar orientation. Lobes must be ordered in each texel so that:

- each lobe i in a texel corresponds to (*i.e.* can safely be interpolated with) the lobes i in neighboring texels,
- the lobes i of 4 texels correspond to the lobe i of the corresponding low-res texel.

This is called lobe *alignment* in the publications. When lobes are aligned, the parameters of a lobe i can be trilinearly interpolated in order to reduce the shading cost compared to Fournier’s method. A *lobe alignment* constraint is added in the non-linear optimization step [TLQ*05, HSRG07, TLQ*08]. As noted by Bruneton et al. [BN12] and Dupuy et al. [DHI*13], lobe alignment can be ill-posed, and errors in alignment causes rendering artifacts. Xu et al [XWZB17] proposed a mutilobe method that overcomes these limitations. They divide the hemisphere of directions into several regions, and use one lobe per region, such that lobes are *aligned* by construction in neighbouring texels and in all the mip maps. Their method could be extended to support cases where color and normals are correlated. The authors reported some temporal noise and discussed the lack of accuracy for representing highly anisotropic normal distributions.

Rendering surfaces with tiny glints and scratches. Yan et al. [YHJ*14] addressed the problem of rendering specular surfaces with very high-resolution normal maps representing sub-pixel scratches and specular flakes. Unlike previous methods, they did not attempt to simplify the effective normal distribution of the pixel footprint since the high-frequencies of this normal distribution characterizes the appearance of their specular surfaces under high-frequency incident illumination (for instance a very small area light). Therefore, their method aims at evaluating more efficiently the effective normal distribution, rather than reducing the storage cost required by the high resolution normal map. Jakob et al. [JHY*14] addressed a very similar problem in the case of glints created procedurally, with a method for efficiently finding the facets in the spatial and angular domain. Their method is procedural and have low storage requirements, although it cannot be used for rendering structures whose size is larger than pixels. Yan et al. [YHMR16] later proposed an alternative version of their first method [YHJ*14], in which they represent the surface details as 4D Gaussian elements (two spatial dimensions and two angular dimensions) for improving rendering efficiency. In our work, we did not address specifically these types of surface appearance and we focused on prefiltering arbitrary surfaces while drastically reducing memory usage.

Filtering normals on curved surfaces. Kaplanyan et al. [KHPL16] addressed the problem of filtering tiny but important specular highlights on curved surfaces, in order to avoid aliasing in real-time rendering applications. Their method is related to normal map prefiltering since they filter the distribution of normals in a pixel footprint covering a curved surface. Given the camera and a point light, they estimate a set of half-vectors in slope space that contribute to light scattering from the light to the camera, in a pixel footprint around a point x on the surface. Then, they integrate the microfacet distribution over this set of half-vectors in slope space. The derivation is based on some approximations of the local geometry around x .

4.4.3 Prefiltering displaced surfaces

Methods for prefiltering color textures and normal map always consider that the surface is flat in the pixel footprint. As explained in section 4.3, prefiltering displaced surfaces is more challenging due to view- and light-dependent masking and shadowing effects, and potential correlations between visibility and surface attributes.

Bruneton and Neyret [BN12] discussed visibility prefiltering in their survey, and in particular horizon maps which encode visibility information as an elevation angle for each azimuthal direction, at each point on the surface. Heitz et al. [HNPN13, HNPN13] prefilter color mapped surfaces with small geometric details and complex visibility effects. They assume that the color of the surface can be decomposed into three functions:

- a term that is decorrelated to the surface geometry,
- a term that is correlated to surface height only,
- a term that is correlated to surface slopes only.

Under a Gaussian surface assumption [Smi67] (Gaussian distributions of heights and slopes) and the hypothesis of uncorrelated slopes and height, they derive reflectance models of surfaces with complex microscopic details. In this method, the relation between the micro-geometry and the colors must be known explicitly, which is not the case when prefiltering arbitrary textured surfaces. Dupuy et al. prefiltered displacement maps based on microfacet distribution prefiltering [DHI*13]. They derive masking-shadowing terms in order to obtain a physically-based BRDF, and they assume that the micro-geometry and the reflectance are uncorrelated.

4.5 Summary of the chapter

We have shown how pixel integrals can be rewritten as integrals in surface space in the case of flat surfaces, and how prefiltering methods can be derived from these integrals. We have seen that under some hypotheses, the surface appearance can be prefiltered without impacting pixel values, meaning that surface-space prefiltering sometimes allows for more efficient rendering without loss of quality.

We have seen that the case of non-flat surfaces introduces new challenges because of visibility effects at small scales: the pixel integral rewrites as an integral over the *visible* surface from the camera, which makes prefiltering equations more difficult to derive.

We reviewed some previous work on surface-space prefiltering, including the well known mip mapping technique, various normal prefiltering methods and non-linear methods for complex surfaces. Prefiltering non-flat surfaces is a difficult and rather understudied problem compared to normal distribution prefiltering.

4.6 Résumé du chapitre

Nous avons montré que les valeurs des pixels, définies comme des intégrales, peuvent dans certains cas être réécrites avec des intégrales sur l’empreinte du pixel sur la surface. Nous avons montré comment obtenir des équations de préfiltrage à partir de ces intégrales dans le cas des surfaces qui sont localement plates. Nous avons également montré qu’il était possible de préfiltrer les surfaces et donc d’accélérer le rendu sans changer la valeur des pixels.

Nous avons commenté le cas des surfaces non planes, et vu que le problème du préfiltrage est bien plus complexe pour ces surfaces, notamment à cause d’effets de masquage et d’ombrage à de petites échelles. Dans cette situation, l’intégrale d’un pixel peut être réécrite comme une intégrale sur la partie de la surface qui est visible par le pixel, ce qui complique le problème du préfiltrage puisque cette partie visible dépend du point de vue.

Nous avons présenté plusieurs méthodes existantes pour le filtrage de l’apparence des surfaces, et en particulier les techniques de mip mapping et le filtrage des distributions de normales. Le préfiltrage de surfaces non planes est un problème difficile et plutôt sous-étudié, contrairement au filtrage de normales.

Chapter 5

Previous work on prefiltering aggregate detail

In the previous chapter, we presented prefiltering methods that can be used in cases where pixel footprints cover entirely simple surfaces. However, these methods do not solve the problem of prefiltering collections of sub-resolution elements, such as complex buildings (Eiffel Tower, Golden Gate) seen from a distance, sailing ships and trees. The problem is related to the rendering of materials made of large amounts of small and similar elements, sometimes called *aggregate detail*. Such materials include hair, fabrics, sand, snow and foliage. The small elements must be rendered explicitly in close views at the price of large storage requirements and costly explicit path tracing, but they cannot be seen individually from a distance and must be prefiltered for efficient rendering. In this chapter, we review previous work on prefiltering sub-resolution elements in the context of off-line physically-based rendering.

5.1 The appearance of aggregate detail

We first provide some general information about the appearance of various common types of aggregate detail. This gives some intuition about how light is scattered in assemblies of elements and how to prefilter them accurately.

5.1.1 The appearance of hair and fur

Hair consists in dielectric fibers covered by microscopic scales forming the cuticle, with internal cylindrical structures (cortex, medulla). Light is reflected by hair fibers, but it is also refracted, internally reflected, scattered and absorbed in fibers. Hair fibers are usually aligned coherently, leading to a specific anisotropic appearance. Light is scattered many times in the hair, and multiple scattering contributes a lot to the appearance. In the case of high-albedo hair, for example human blond hair, multiple scattering is even predominant as shown in Fig. 5.1, partly because an important part of the incident light is scattered forward by hair fibers [MJC*03], meaning that light easily penetrates the hair. In computer graphics, hair fibers are modeled using dielectric tubes of elliptical sections [MJC*03,ZW07,dFH*11,dMH13,CBTB16], with parametric scattering models that take into account the effects of the tiny scales at the surface of the cuticle and additional internal structures in the case of non-human hair [YJR17].

5.1.2 The appearance of fabrics and clothes

Like hair and fur, fabrics are made of assemblies of fibers. They also have a complex anisotropic appearance (Fig. 1.1a) due to the fact that fibers are coherently organized in yarns, that are themselves woven

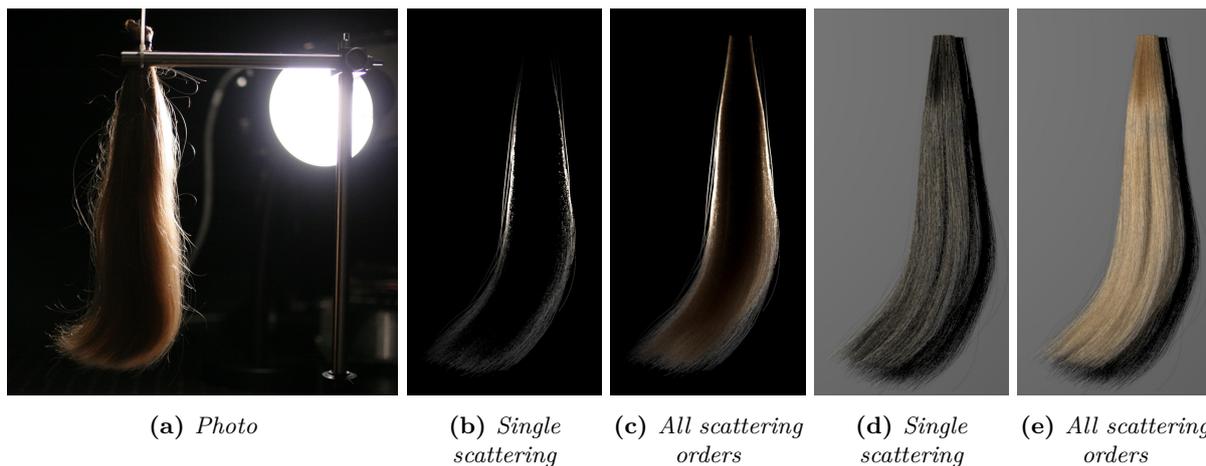


Figure 5.1: Hair photography (a) and rendering (b,c,d,e) from the work of Moon and Marschner [MM06]. Hair rendering allows for observing how much multiple scattering contributes to the appearance of aggregate materials with high albedo elements.

into regular patterns with well-defined orientations. Moreover, several types of yarns can be used in the same piece of cloth, sometimes leading to complex colored anisotropic reflections. In close views, fabrics are three dimensional, and the visibility of each type of yarn is view-dependent. For this reason, fabrics often show interesting reflectance properties at grazing angles where only a subset of the fibers contributes to the appearance. As for hair, light is scattered multiple times in fabrics, and multiple scattering may contribute a lot to the appearance.

5.1.3 The appearance of granular materials

Assemblies of grains such as snow, sand or sugar raise the same problems as hair and fabrics: large storage requirements for close views, costly rendering due to highly complex geometry, predominance of multiple scattering for high-albedo grains. Contrary to fibers, most grain assemblies do not have a strong anisotropic appearance, which simplifies the problem of prefiltering them. The appearance of some granular materials such as snow or sugar is characterized by many glints as shown in Fig. 5.2.

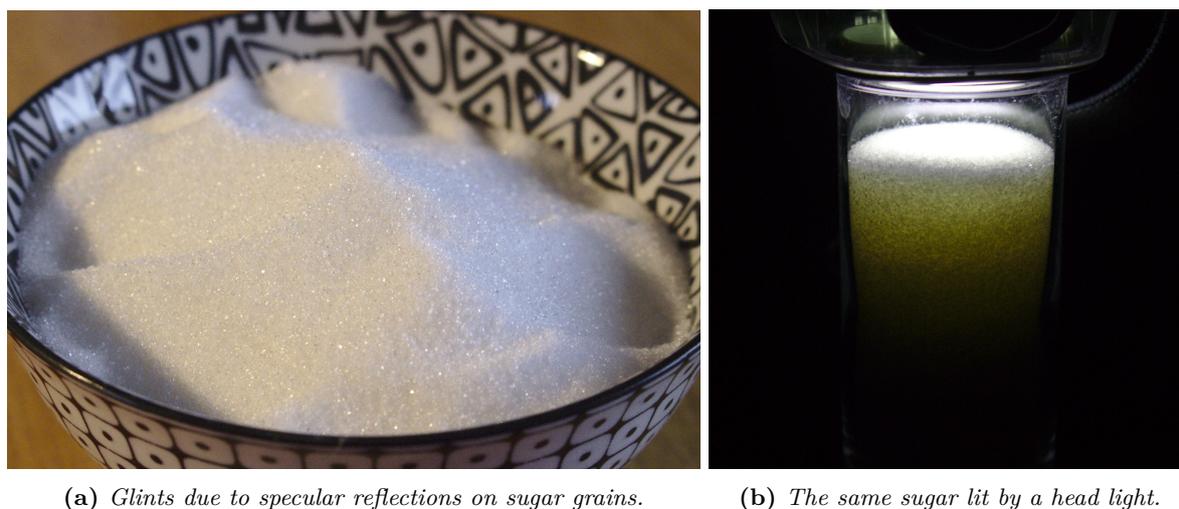
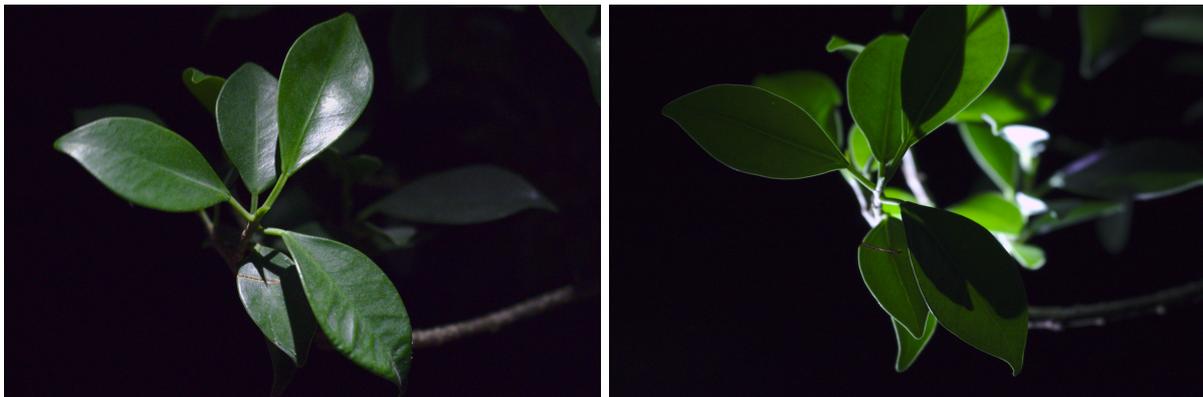


Figure 5.2: Various aspects of the appearance of sugar. Like in other granular materials, specular reflections on grains create glints (a). Light is scattered by these high-albedo grains and penetrates inside the material (b). Here, multiple scattering adds a saturated yellow tint to the material.

5.1.4 The appearance of foliage

Trees and other plants are typically observed at various scales: we can observe the individual leaves when we are close from them, but we also see thousands of distant trees at once when we contemplate landscapes. Light is reflected from leaves but also diffusely transmitted through them (Fig. 5.3). Bright leaves also result in glints when they are observed from a distance (Fig. 1.1b). Leaves are often coherently organized, leading to anisotropic occlusion and appearance. The front and the back of leaves typically have different scattering properties. Multiple scattering can be important in foliage although it is generally less critical than in high-albedo aggregate materials.



(a) Photo of leaves with a front light.

(b) The same leaves and a back light.

Figure 5.3: *The appearance of leaves is often characterized by diffuse and specular reflections (a) as well as translucency, that is, light diffusely transmitted through the leaves (b).*

5.1.5 Other complex appearances

Apart from the various types of aggregate detail presented above, other types of complex geometry exist and require to be rendered at various scales. For instance, complex buildings seen from a distance such as the Eiffel tower or the Golden Gate, or distant sailing ships with sub-resolution ropes. These appearances are most of the time less complex than aggregate detail, but remain challenging for artifact-free and all-scale prefiltering.

5.1.6 Relation to participating media.

All these examples have something in common: their semi-transparent distant appearance is analogous to light scattering in *participating media* such as wax, milk, clouds or skin. Indeed, light interacts with foliage, hair or snow with some probabilities that depend on the density of individual elements, and the macroscopic appearance results from their local scattering properties. From a distance, a statistical description of the position and the orientation of the elements is sufficient, from which can be derived occlusion probabilities and scattering functions. This is why participating media models have been used for approximating scattering in aggregate materials, as discussed in Sec. 5.4.

5.2 Stochastic simplification

Cook et al. [CHPR07] addressed the problem of simplifying aggregate detail for rendering massive production scenes. The landscape shown in their article is made of lots of small elements (‘a hundred million leaves’) and was ‘unrenderable’ without simplification.

In the same spirit as other stochastic methods [KKF*02,DCSD02], they randomly simplify complex assets by deleting some of the elements and changing the size and color of the remaining elements in order to approximately preserve the overall appearance of the asset. They propose methods for scaling elements for a given pruning level in order to reduce the variation of the projected area of the asset across scales. Unlike other prefiltering methods, they do not average the properties of the elements locally. Instead, they combine the colors of remaining elements with the average color of the entire asset to avoid artifacts (they call this ‘contrast correction’). This means that assets with a spatially-varying appearance have an homogeneous (and therefore incorrect) appearance at coarse resolutions. Despite its limitations, this strategy allows for simple and efficient simplification and it is probably used in many studios.

More importantly, stochastic simplification assumes that the appearance of their aggregate geometry can be prefiltered using a few large elements. This hypothesis does not hold at coarse resolutions: approximating a foliage with a few large leaves does not preserve the self-shadowing probabilities of intricate geometry nor other scattering statistics. Similarly, a hair strand does not have the same scattering characteristic as a single large hair fiber. Such problems have been observed at Weta Digital and at the Walt Disney Animation Studios. Larger leaves create unwanted glints unless normals are prefiltered, and LOD representations of foliage are always too bright unless the masking and shadowing probabilities in the input geometry are taken into account.

5.3 Background: participating media models in computer graphics

Participating media models are ubiquitous in off-line production rendering for smoke, clouds, skin, marble, scattering liquids or other volumetric effects. Sub-surface scattering is a special case in which light is only absorbed and scattered in the close vicinity of the surface. In this situation, it is possible to derive simplified models with analytical expressions based on diffusion approximations [Ish78,JMLH01,HCJ13], although unbiased volume path tracing is now affordable for some studios [CKB16,WVH17] and gives significantly more accurate results, in particular in the case of curved and relatively thin objects.

Despite the wide range of assets that are rendered with volumetric light scattering, the number of available volume scattering models is actually very small. We review them in this section.

5.3.1 The radiative transfer equation

Participating media models in computer graphics come from previous work on radiative transfer in physics [Cha50,Ish78]. The fundamental equation is the *radiative transfer equation* which expresses a radiance at a specific location in a participating medium depending on local attenuation, scattering and emission. We recall the main results here and we refer the reader to the course of Marschner [Mar12] for a more comprehensive discussion about the origin of this equation and a step by step derivation.

Attenuation in a medium. As light travels in a participating medium, some of it is absorbed or scattered, while some light does not interact and propagates in a straight line. This is modeled using an attenuation coefficient σ_t with unit m^{-1} , and the Beer-Lambert law gives the attenuation of light for a distance d in the media: $e^{-\sigma_t d}$. The attenuation σ_t can depend on the wavelength in the model. It can be spatially varying, in which case the attenuation along a distance d in the medium is given by an integral of the attenuation:

$$\exp\left(-\int_{s=0}^d \sigma_t(s) ds\right).$$

Scattering coefficient. Similarly, a scattering coefficient is used for characterizing the amount of scattered light per unit distance in the medium. It is noted σ_s in this manuscript, and its unit is the m^{-1} . It is also wavelength-dependent and can be spatially varying. Because attenuated light is either

scattered or absorbed, it is possible to define an absorption coefficient:

$$\sigma_a = \sigma_t - \sigma_s.$$

Phase functions. Light can be scattered isotropically or in specific directions when it interacts with a participating medium. This can be characterized by a probability distribution function in the spherical domain S^2 of the scattered energy for an incident direction ω_i . This distribution is called the phase function and it is noted $f(\omega_i, \omega_o)$. The name *phase function* comes from astronomy where the brightness of an reflecting body (for instance, the Moon) depends on its *phase*, that is the angle between the view direction and the light source. Most phase functions only depend on the angle between ω_i and ω_o and are noted $f(\omega_i \cdot \omega_o)$. Normalized phase functions satisfy

$$\int_{S^2} f(\omega_i \cdot \omega_o) d\omega_o = 1, \quad \forall \omega_i \in S^2.$$

The isotropic phase function is the simplest model. It corresponds to light scattered equally in all directions, and it writes

$$f(\omega_i \cdot \omega_o) = \frac{1}{4\pi}.$$

The Henyey-Greenstein phase function [HG41], often abbreviated *HG*, is widely used in computer graphics because of its simplicity, and its parameter $g \in [-1, 1]$ that allows for choosing between forward scattering (when $g > 0$) and backward scattering (when $g < 0$), which drastically change the appearance of participating media. It writes:

$$f(\omega_i \cdot \omega_o) = \frac{1}{4\pi} \frac{1 - g^2}{(1 + g^2 - 2g(\omega_i \cdot \omega_o))^{3/2}}.$$

Rayleigh and Mie scattering are other models used in the case of light scattered by small particles, in particular in the atmosphere and in clouds.

The radiative transfer equation (RTE). Using the functions introduced above, the radiative transfer equation at a point p in the volume writes:

$$(\omega \cdot \nabla)L(p, \omega) + \sigma_t L(p, \omega) = \sigma_s \int_{S^2} f(\omega \cdot \omega') L(p, \omega') d\omega' + Q(p, \omega)$$

with $L(p, \omega)$ is the radiance at a point p in direction ω and $Q(p, \omega)$ a source term characterizing light emission in the volume, in $\text{W}\cdot\text{m}^{-3}\cdot\text{sr}^{-1}$, that is an emitted radiance per meter. The term

$$(\omega \cdot \nabla)L(p, \omega) = \omega \cdot \begin{bmatrix} \frac{\partial L(p, \omega)}{\partial x} \\ \frac{\partial L(p, \omega)}{\partial y} \\ \frac{\partial L(p, \omega)}{\partial z} \end{bmatrix}$$

is a directional derivative of $L(p, \omega)$ at a point p along ω , that is how $L(p, \omega)$ varies for an infinitesimal spatial displacement from p in direction ω . Note that this equation is similar to the local illumination equation that we used for surfaces (Chap. 4): the integral of the BRDF and the incident radiance over the spherical domain is replaced by the integral of the phase function with the incident radiance.

5.3.2 Anisotropic RTE and microflake model

In the “classical” RTE presented above, the attenuation coefficient σ_t and the scattering coefficient σ_s do not depend on any direction. The classical RTE only models *isotropic media*, that is media whose

appearance does not change with the view direction. For example, for example, let's suppose that we are looking at a sphere of isotropic media, we could rotate it arbitrarily and it would have exactly the same appearance. This model is therefore limited for representing anisotropic media, for example media made of microscopic well-organized non-spherical elements. Jakob et al. [JAM*10a] have introduced an extension of the classical RTE physically-based model which allows for modeling such media. We will highly rely on this framework in our thesis and we recall their main results.

Anisotropic RTE

The anisotropic radiative transfer equation (RTE) proposed by Jakob et al. [JAM*10a] writes:

$$(\omega \cdot \nabla)L(\omega) + \sigma_t(\omega)L(\omega) = \sigma_s(\omega) \int_{S^2} f(\omega \rightarrow \omega')L(\omega') d\omega' + Q(\omega)$$

with $\sigma_t(\omega)$ the anisotropic attenuation coefficient, $\sigma_s(\omega)$ the anisotropic scattering coefficient (which is usually wavelength dependent), and f the anisotropic phase function (in the sense that it depends on ω and ω' , and not only on the angle between ω and ω'). In this manuscript, we use phase functions that are normalized over the second parameter:

$$\int_{S^2} f(\omega \rightarrow \omega') d\omega' = 1, \quad \forall \omega.$$

Helmholtz's reciprocity principle for anisotropic media

In general, phase functions in the anisotropic RTE framework do not satisfy $f(\omega \rightarrow \omega') = f(\omega' \rightarrow \omega)$. The Helmholtz's reciprocity principle, which states that the radiative transfer remains the same when interchanging sources and receivers, writes

$$\sigma_s(\omega)f(\omega \rightarrow \omega') = \sigma_s(\omega')f(\omega' \rightarrow \omega).$$

This is the reason why using arbitrary phase functions for a media with anisotropic attenuation coefficients often break reciprocity.

Microflake model

Based on their anisotropic RTE, Jakob et al. [JAM*10a] have studied the case of media made of clouds of microflakes, that is, microscopic flat surfaces with random positions and whose scattering properties are described by a micro-BRDF. The orientation of the microflakes in the medium is characterized by a microflake normal distribution D in the spherical domain. Jakob et al. derived attenuation coefficients and phase functions for such media and the Helmholtz's reciprocity principle is satisfied by construction. When the microflake micro-BRDF is perfectly specular, deriving evaluation and sampling procedures for the phase functions is simpler.

In this manuscript, we assume like Heitz et al. [HDCD15] that microflakes reflect on both side and we only consider normal distributions D that satisfy

$$D(\omega) = D(-\omega).$$

Given D , the attenuation coefficient is derived from the integral of the projected areas of the microflakes

$$\sigma_t(\omega) = \rho \int D(\omega_m) \langle \omega \cdot \omega_m \rangle d\omega_m \quad (5.3.1)$$

where ρ , often referred to as *density*, is the amount of microflake surface per unit volume, in m^{-1} . Assuming that the microflake color is not view-dependent, the scattering coefficient for a wavelength λ is given by

$$\sigma_s(\omega) = \rho \alpha(\lambda) \int D(\omega_m) \langle \omega \cdot \omega_m \rangle d\omega_m = \alpha(\lambda) \sigma_t(\omega) \quad (5.3.2)$$

with $\alpha(\lambda)$ the albedo of the microflakes. In the case of specular microflakes, the phase function writes

$$f(\omega \rightarrow \omega') = \frac{\rho\alpha(\lambda)D(\omega_h)}{4\sigma_s(\omega)} \quad (5.3.3)$$

with $\omega_h = \frac{\omega + \omega'}{\|\omega + \omega'\|}$ the half-vector.

5.4 Volumetric approximations of aggregate geometry

As mentioned in section 5.1, light scattering in aggregate geometry viewed from a distance is related to volume models used in physics. The precise position of each small element is not required, so that scattering and free-flight distances can be described statistically, which is exactly what volumetric models do. Volumetric models are memory efficient (they often have few parameters locally) and they allow huge savings in rendering time compared to rendering explicitly all the sub-resolution elements. Volumes have been used as approximations of complex geometry by many authors. We review their work in this section. In particular, we review several methods that rely on volumetric approximations for computing multiple scattering in aggregate geometry, since this is a type of prefiltering in which the statistics of light scattering in the volume must match the ones of the input explicit geometry.

The distant appearance of aggregate geometry emerges from the properties of the elements it is made of. It is often complex due to its semi-transparency, the visibility and shadowing effects between elements, the importance of multiple scattering for elements with high albedo, and sometimes the complex reflectance properties of individual scatterers. Therefore, most prefiltering methods aim at simplifying one type of complex geometry and have a limited validity domain.

5.4.1 Types of volumetric models

By *volumetric models*, we designate all models that rely on a statistical attenuation of light, based on attenuation coefficients or opacity values, and in which scattering is not governed by explicit geometry but by scattering coefficients and phase functions. Most models rely on the radiative transfer equation, but some authors proposed custom models that do not ensure Helmholtz reciprocity, and some proposed non-classical volume models with non-local scattering operations and non-exponential free-flight distance distributions.

5.4.2 Early volumetric approximations

Blinn [Bli82] introduced volumetric models in computer graphics. As other authors in the physics literature [Cha50], he proposed a model of single scattering in clouds of randomly positioned spherical scatterers, and used it for rendering the rings of Saturn made of particles of water ice. Blinn also presented various simple and widely used phase functions in other fields (isotropic, diffuse spheres, Henyey-Greenstein, Rayleigh, etc.). Kajiya et al. [KVH84] rendered heterogeneous volumes, and later proposed a path tracing algorithm [Kaj86] that has then been later generalized for rendering volumes. In another work [KK89], Kajiya and Kay used heterogeneous volumes for representing fur. Their medium had several parameters: a local projected area of microspheres (which gives an attenuation), a local fiber direction and a reflectance function. They introduced two fiber scattering models derived from Lambertian and specular cylinders. The volumetric fur was rendered with single scattering and volumetric shadows with ray tracing. The idea of using volumes instead of explicit cylinders has inspired many other authors as discussed below.

5.4.3 Prefiltering with hierarchical volume representations

Various forms of volume representations have been used for efficiently storing and rendering prefiltered versions of complex arbitrary geometry. Neyret [Ney95, Ney98] prefilter volumetric textures by representing the underlying geometry using ellipsoids, which can encode isotropic media as well as foliage-like and fur-like normal distributions. The same idea was later adapted to the physically-based microflake framework with the introduction of the SGGX microflake normal distribution [HDCD15], which is the normal distribution of an arbitrary ellipsoid. Lacewell et al. [LBBS08] prefilter complex geometry for reducing the noise in soft shadows. They use a bounding volume hierarchy and store directional occlusion information for each box. They show that the noise can be significantly reduced by prefiltering occlusion. They did not extend this approach with reflectance information for rendering directly distant foliage. As mentioned in their work, volume representations cannot be used for representing accurately opaque surfaces larger than the prefiltering scale. In their method, they chose manually a *correlation height*, that is a prefiltering scale from which occlusion can be prefiltered using their representation. Crassin et al. [CNS*11] also encoded geometric information using voxels for computing indirect illumination in real time. They used anisotropic voxels that encode view-dependent occlusion and appearance in the 6 main directions, and they proposed a simple algorithm for volume downsampling. Their method is less accurate for occlusion prefiltering than the one of Lacewell et al., but they encode reflectance information such as colors and normals. However, their representation has not been used nor evaluated for rendering directly distant assets.

5.4.4 Approximating hair and fur

The realistic rendering of hair is very challenging for several reasons:

- Light scattering in individual hair fibers is complex. Accurate, efficient and controllable hair shading models are still being investigated [dFH*11, dMH13, CBTB16, YJR17].
- The geometry of assemblies of hair fibers is complex and has large storage requirements. Computing ray intersections with these fibers is rather costly.
- Moreover, many intersections must be computed for rendering multiple scattering, making hair rendering even more costly.
- Hair rendering is subject to noise, because of the high spatial frequency and the complexity of light scattering.

For these reasons, various authors proposed ways to render hair more efficiently than brute-force approaches. Kajiya and Kay [KK89] used volumetric textures for rendering fur. Volumes were later used in other methods, for instance by Andersen et al. [AFFC16] for rendering the smooth appearance of sub-resolution undercoat hair. However, most methods rely on explicit hair fibers modeled using segments or other linear primitives. With the introduction in computer graphics of the first realistic scattering functions for human hair [MJC*03, ZSW04], it has been observed that multiple scattering plays an important role in the appearance of hair, and in particular in the case of high-albedo hair fibers (*e.g.* blond hair, white hair).

A seminal work on approximating light scattering in hair was done by Moon and Marschner [MM06]. Inspired by volume photon mapping algorithms [JC98], they proposed to trace particles in the explicit geometry during a first pass and to store their positions and directions in a 5D map. At rendering, the radiance $L(x, \omega)$ due to multiple scattering in the hair is estimated using the map and smooth 5D kernels, whereas low-order scattering is computed using the explicit geometry. This method does not exactly rely on a volumetric model but like other work on realistic hair rendering it uses the fact that multiple scattering creates a low-frequency radiance field that can be statistically estimated without explicitly computing every fiber-to-fiber scattering events. A similar idea has been proposed simultaneously by Zinke et al. [ZW06].

Moon et al. [MWM08] later proposed an alternative and more efficient solution. They voxelize the hair geometry by estimating in each voxel a main fiber direction and a standard deviation of fiber directions.

They also define an anisotropic attenuation coefficient that depend on the distribution of fiber directions. In a first pass, the light particles are scattered according to the volume representation, which is faster than light tracing in the explicit geometry [MM06]. The radiance information is stored using spherical harmonics in each voxel, and this information is filtered both in the angular domain (to avoid ringing) and spatially (to avoid noise). The SH representation allows for computing efficiently the integral of the product of the incident radiance and a fiber reflectance functions at rendering. The accuracy of their volumetric model has not be evaluated for the rendering of distant hair without explicit geometry.

5.4.5 Volumetric models of fabrics and clothes

As for hair, rendering fabrics using explicit fibers is realistic but inefficient, especially when multiple scattering must be computed. Schröder et al. [SKZ11] proposed to use volumes for rendering fabrics. They voxelize line segments at the scale of yarns, and show that the appearance of their volumetric representation can match accurately the images obtained by rendering the explicit geometry. In each voxel, they represent the fiber distribution using a Gaussian distribution, and they use Gaussian mixtures when voxels contain various yarns. Their volumetric model does not rely on the RTE. They use a precomputed visibility function for accounting for masking and shadowing at the scale of dense yarns, although they rely on a volumetric attenuation for global shadowing effects. They evaluated their method at multiple scales using various resolutions of voxel grids. Their low-resolution volumes show several artifacts because of some assumptions made at the level of yarns that do not hold at all scales. Moreover, their results show aliasing due to their voxelization algorithm that relies on 3D non-overlapping box filters: this indeed minimizes blur but introduces artifacts as discussed in Chap. 9, especially for fabrics in which the geometry is periodic with high-frequencies.

Jakob et al. introduced the anisotropic radiative transfer equation [JAM*10a] and applied it to the rendering of fabrics, using fiber-like microflake normal distributions in the microflake model. Zhao et al. [ZJMB11] built on this work and used the microflake model for rendering the micro-geometry of fabrics acquired from X-ray computed tomography. They introduced another fiber-like microflake distribution and estimate scattering parameters for their volumetric model from photographs of real fabrics. They show that volumetric models based on the anisotropic RTE can reproduce the complex appearance of fabrics such as velvet or satin.

In another work, Zhao et al. [ZHRB13] addressed the problem of rendering efficiently multiple scattering in very detailed microflake volumes, leveraging the fact that fabrics can be modeled with a few blocks of voxels tiled on a base surface. They precompute voxel-to-voxel, patch-to-patch, and patch-to-voxel light transfer and use this information for simulating efficiently high-order scattering. Heitz et al. [HDCD15] introduced a new parametric microflake distribution that can represent fiber-like media and can be prefiltered efficiently, although they do not address the problem of prefiltering density and scattering parameters. Zhao et al [ZWDR16] addressed this problem by optimizing scattering parameters in image space. Their work will be presented in more detail in section 11.3.3. We also contribute to this problem and we propose an alternative solution for accurate volume downsampling (Part III).

Khungurn et al. [KSZ*15] evaluated the accuracy of several volumetric representations compared to the rendering of explicit fibers for matching the appearance of real fabrics. They compared three representations: explicit geometry with a fiber scattering model (BCSDF), a volumetric representation with voxelized geometry and rendered with the microflake model, and also a volumetric representation rendered with phase functions derived from the fiber BCSDF. They have drawn the conclusion that all these methods can be used for rendering realistic fabrics, but that representations based on the fiber BCSDF are more accurate than the microflake model. Although they focused on fabrics, the same conclusions can probably be drawn for hair rendering: a phase function derived from dielectric fiber scattering models would be more accurate than a fiber-like microflake phase function.

5.4.6 Approximating granular materials

Multiple scattering in granular materials has also be studied and rendered using volumetric approximations. Moon et al. [MWM07] first proposed not to rely on the RTE, observing that light scattering

densely packed in grain assemblies do not follow the base assumption of independent scattering due to well-separated small scatterers from which the RTE is derived. They measured the scattering properties and distributions of free-flight distances in granular materials and showed the differences with the classical RTE. They introduced a *shell transport operator* that is precomputed for a particular granular materials and can be used for simulating various scattering events, leading to an efficient but specific rendering solution. They used a tabulated free-flight distribution function and do not rely on usual exponential distributions. However, the visual differences between their non-classical model and the classical RTE have not been demonstrated. They noted that rendering high-albedo granular materials without approximation is almost impossible, and their volumetric approximation greatly alleviate this problem. As for hair [MM06, MWM08], low-order scattering is computed with the explicit geometry.

Meng et al. [MPH*15] revisited this work and combined three rendering algorithms: explicit path tracing for low-order scattering when grains are seen individually, volume path tracing, as well as a diffusion approximation as previous work on sub-surface scattering [DI11]. They introduced another non-classical volume model, but derive parameters of the classical RTE (attenuation coefficient, scattering coefficient, phase functions) and rely on the classical volume path tracing at rendering. Their results show that the RTE framework provides a good approximation of densely packed grains, whereas the diffusion approximation introduces more bias but further improves the rendering efficiency.

Directly inspired by previous work [MWM07, MPH*15], Müller et al. [MPG*16] proposed a more general solution for rendering granular materials with large-scale spatial variations of grain properties. They also combine various strategies including volume approximations (*shell tracing*) and prefiltered grain scattering functions used for direct illumination when grains are sub-pixels, which is more efficient compared to rendering distant grains with their explicit geometry. As noted by the authors, rendering the explicit geometry remains the only way to preserve the glints of such materials. Ideally, the path tracing of the explicit geometry should not be necessary for distant grains, and a prefiltering method that preserves glints for such materials remains to be found.

5.4.7 Prefiltering foliage with volumes

Interestingly, foliage has received less attention in recent research on prefiltering complex geometry, despite its close relationship to other aggregate materials, maybe because leaves have lower albedos and the rendering of multiple scattering converges faster compared to blond hair or snow. Volume approximations have been used as well, for example by Max et al. [MKMW97], or by heitz et al. [HDCD15]. Leaves diffusely transmit light and often produce glints, and prefiltering accurately such effects has not been done yet to our knowledge. In Part II, we prefilter sub-resolution intricate geometry using the microflake model and we show how to estimate attenuation coefficients and normal distributions.

5.5 Summary of the chapter

In this chapter, we reviewed several accurate prefiltering methods for intricate geometry for which the surface-based methods seen in chapters 3 and 4 cannot be used. We have seen that:

- The stochastic simplification method [CHPR07] is efficient but has limited accuracy and validity domain. Because the local occlusion and scattering properties are not prefiltered, this method cannot be used for prefiltering assemblies of specular elements or spatially varying aggregate detail.
- We discussed the appearance of the main types of aggregate detail useful for computer graphics because of their ubiquity: hair, fur, fabrics, granular materials and foliage. We highlighted their similarity: their volumetric behavior at some scales, the importance of multiple scattering and the complex scattering in individual elements.
- We presented the previous work on volumetric approximations of such materials, and we discussed the fact that these approximations are memory efficient and highly improve the rendering efficiency in the case of fabrics [ZHRB13, KSZ*15], hair [MM06, MWM08] and granular materials [MWM07, MPH*15, MPG*16].

- We have presented the classical radiative transfer equation and its anisotropic extension [JAM*10a], on which we built our work in Parts II and III of this manuscript.

5.6 Résumé du chapitre

Dans ce chapitre, nous avons présenté et commenté plusieurs méthodes de préfiltrage pour des géométries complexes qui ne peuvent pas être traitées avec les méthodes surfaciques présentées dans les chapitres 3 et 4. Nous avons vu que :

- La suppression aléatoire d'éléments [CHPR07] est efficace mais elle est peu fiable et a un domaine de validité relativement restreint. Elle ne peut pas être utilisée dans le cas de géométries dont l'apparence varie spatialement parce que la transparence et les propriétés de réflexion locales ne sont pas préfiltrées.
- Nous avons passé en revue plusieurs types d'apparence complexe dont le rendu est utile en informatique graphique : les cheveux, la fourrure, les tissus, les ensembles de grains et le feuillage. Nous avons souligné certaines similarités entre ces matériaux, et notamment leur comportement volumique à certaines échelles, l'importance des diffusions multiples sur leur apparence et la complexité des réflexions à l'échelle des petits éléments.
- Nous avons présenté les méthodes existantes sur l'approximation de tels matériaux par des représentations volumiques, et nous avons vu que ces représentations sont efficaces du point de vue de la mémoire et qu'elles permettent d'améliorer grandement l'efficacité du rendu dans le cas des tissus [ZHRB13, KSZ*15], des cheveux [MM06, MWM08] et des matériaux granulaires [MWM07, MPH*15, MPG*16].
- Nous avons présenté les équations du modèle de transfert radiatif classique et de son extension anisotrope [JAM*10a], sur laquelle nous nous appuierons dans les parties II et III.

Part II

A new hybrid prefiltering method with meshes and volumes

In this part of the dissertation, we address the problem of generating appearance-preserving LODs of high-resolution textured meshes. We have seen that mesh simplification is a well studied problem in computer graphics (Chap. 3) and that several authors have addressed the problem of prefiltering the appearance of textured surfaces (Chap. 4). Despite this previous work, prefiltering complex assets such as buildings or trees at arbitrary scales remains understudied, because most methods rely on the assumption that assets have the appearance of simple surfaces at the prefiltering scale, which does not hold at scales where the geometry becomes intricate. Because of this *surface-like appearance* assumption, many methods are limited to simple surfaces or are only valid at intermediate prefiltering scales.

Some methods specifically address the problem of prefiltering complex geometry such as granular materials or fibers using volume representations or hybrid mesh-volume methods (Chap. 5). These methods lack generality and cannot be used for automatically prefiltering a wide range of assets. We introduce a new hybrid method for prefiltering assets at arbitrary scales that combines mesh analysis, mesh simplification and voxelization to overcome some limitations of previous methods.

This part of the dissertation begins with a problem analysis of accurate, seamless and general LOD generation from input textured meshes (Chap. 6). We will see that a hybrid approach can fit all the requirements of high-quality LOD generation. We then introduce new tools for prefiltering complex assets at arbitrary scales, including an algorithm for the automatic analysis of macrosurfaces in arbitrary meshes (Chap. 7), the joint simplification of geometry and materials for textured meshes (Chap. 8) and the prefiltering of intricate geometry (Chap. 9). In chapter 10, we discuss some implementation details and show examples of assets prefiltered using our method.

Chapter 6

Problem analysis: prefiltering production assets at arbitrary scales

One of our main motivations for this thesis is to explore new LOD methods for off-line production rendering, and in particular for the film industry in which extremely complex worlds have to be rendered with limited time, memory and CPU cores, and with substantial quality requirements. In this chapter, we provide information about production assets and we discuss the problem of simplifying them while preserving accurately their appearance at arbitrary scales. Based on this analysis, we propose a pipeline for generating LODs (Sec. 6.4) that combines mesh simplification and sub-resolution geometry prefiltering. The various elements of this pipeline are discussed in the next chapters.

6.1 Production assets

In this section, we discuss the properties of rendering primitives used in the industry at the time this dissertation is written. Because studios rarely provide detailed information about their production tools and workflows, the content of this section is mostly based on personal discussions and technical details learned during the internships at Weta Digital and at the Walt Disney Animation Studios.

6.1.1 The production pipeline

The production of special effects and computer-animated films is organized as a *pipeline* in which different departments reuse data from other departments and produce 3D content for a film, until final frames can be computed. This pipeline is complex [CCL14], and modifying it is difficult because changes affect many departments, production tools and artists. Having a global view of the pipeline and considering production constraints allows for identifying important requirements for LOD methods.

The pipeline varies from company to company, but departments such as modeling, texturing, lighting, rigging, animation, physical simulations, compositing, digital matte painting, stereo, layout, crowds, environments or rotoscoping can be found in many studios. The pipeline is ideally but rarely one-way, due to the interdependence of all the departments and the iterative nature of filmmaking. It is not uncommon that artists from a department ask for changes upstream in the pipeline. For instance, artists in the texture department may ask for some modifications of the topology of an asset to the modeling department, or lighting artists could ask for a modification of the material of an asset. A director can also ask for changes that affect various departments. Each department uses specific programs. Many are standard in the industry, such as Maya¹ (modeling, animation), Nuke² (compositing) or Houdini³ (simulation), but

¹Autodesk, <https://www.autodesk.eu/products/maya/overview>

²The Foundry, <https://www.foundry.com/products/nuke>

³SideFX, <https://www.sidefx.com/products/houdini-fx/>

studios also develop their own software and plugins when they need to have a complete control of the source code for their specific needs. For example, several major studios use their own renderers, such as Hyperion⁴ at the Walt Disney Animation Studios, Manuka⁵ at Weta Digital or RenderMan⁶ at Pixar Animation Studios.

Artists do not create entire scenes directly, they model smaller elements that are reviewed independently. Other artists assemble all the elements for a scene or a shot (a series of frames without cuts in a film, typically lasting a few seconds). These elements can be divided into two main groups:

- *Assets*: usually independent objects or elements of a scene. For example, a car, a tree, a fridge, a boulder, etc. Assets can be used in one or several scenes of a film. Large assets such as mountains, riverbeds or the interior of buildings for indoor scenes are sometimes called *environments* or *master sets*.
- *Characters*: they require a specific pipeline that include rigging and other tasks for hair and cloth.

The creation of rendering primitives usually involves three main steps: modeling the geometry, creating some textures, and finally setting the materials using the textures. This process can vary and include captured data from real-world objects, physical simulations, or data generated semi-automatically using procedural methods, for instance in the case of plants. The data processed by the renderers is rather standard in the industry, the most common representation being polygon meshes, subdivided using Catmull-Clark subdivisions [CC78] or similar algorithms (Sec. 3.3.1), displaced using displacement maps, and shaded with spatially-varying reflectance models often based on textures.

6.1.2 Geometry of base meshes

The geometry of assets such as trees and buildings consists of various disconnected meshes. For example, a tree will have distinct meshes for the trunk, the twigs and the leaves. Although renderers compute intersections between rays and triangles, the base meshes are usually quadrilateral meshes (or *quad meshes*) and are triangulated at render time. Quads are often more natural for modeling compared to triangles [DKT98], for example in the case of cylinder-like shapes using *extrusion modeling*⁷. Quads also facilitate texturing.

Production meshes are almost always manifold with boundaries, meaning that each point of the surface must have a neighborhood that is homeomorphic to a disk (for points inside the surfaces) or a half-disk (for points on boundary edges). For example, edges cannot be adjacent to more than 2 faces (Fig. 6.1a), and vertices cannot be adjacent to more than 2 boundary edges (Fig. 6.1b). However, self-intersections are allowed (Fig. 6.1d). These constraints come from several pipeline considerations. In particular, texture filtering is only well-posed for manifold surfaces. Modeling artist intend to produce semi-regular meshes (Fig. 6.2) with feature-aligned face loops.

6.1.3 Subdivision surfaces

Subdivision surface algorithms (Sec. 3.3.1, [CC78, Loo87, DLG90, ZSS97, Kob96]) are used for modeling smooth shapes, and the base mesh is seen as the control mesh of a smooth parametric surface. Smooth subdivision surfaces are sometimes called *limit surfaces* because they are defined by subdivision schemes that are applied iteratively such that the geometry converges towards a smooth surface. However, Catmull-Clark subdivision surfaces can be evaluated as other parametric surfaces using Stam's method [Sta98].

At rendering, a level of subdivision is chosen to approximate the limit surface. It is typically chosen depending on the position of the camera, and base meshes or subdivided meshes are considered as

⁴<https://www.disneyanimation.com/technology/innovations/hyperion>

⁵<https://www.wetafx.co.nz/research-and-tech/technology/manuka/>

⁶<https://renderman.pixar.com/>

⁷https://en.wikipedia.org/wiki/Polygonal_modeling

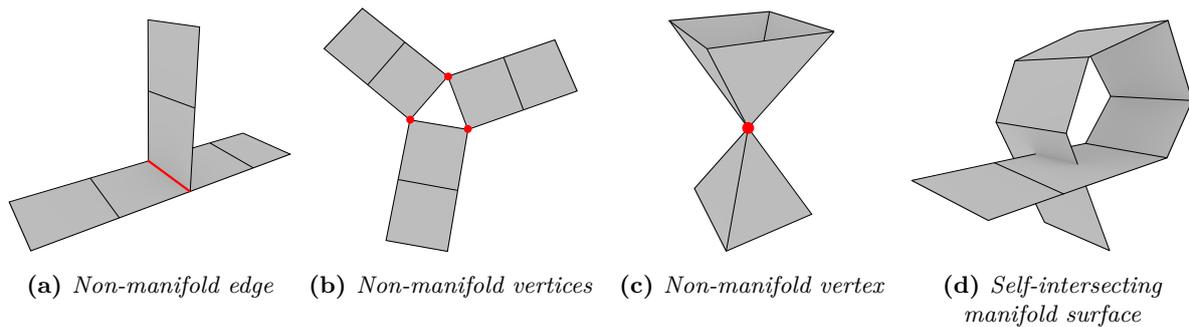


Figure 6.1: The geometry of production assets is usually manifold with boundary, i.e. for any point of the surface, there is a connected neighborhood of this point that is homeomorphic to a disk (for points inside the surface) or a half disk (for points on boundary edges). This means that self-intersections are allowed (d) but edges cannot have more than two adjacent faces (a) and vertices cannot have more than two adjacent boundary edges (b). Note that vertices can be non-manifold even if they are not adjacent to boundary edges (c).

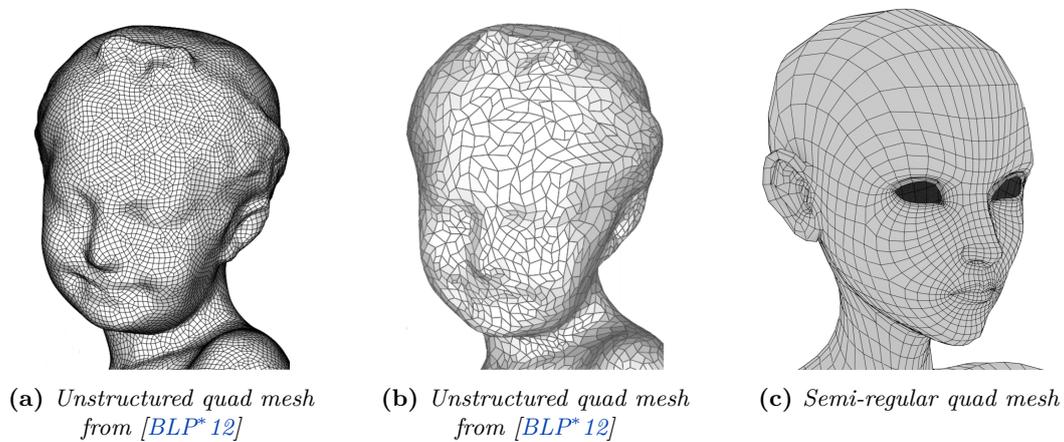


Figure 6.2: Quad meshes can be unstructured (a,b), but modeling artists usually produce semi-regular meshes (c) in which most vertices have 4 adjacent faces, most edges are feature-aligned. This facilitates authoring, texturing, and it avoids artifacts in subdivision surfaces. The mesh (c) comes from the Sintel project (<https://durian.blender.org/news/sintel-lite/>).

good approximations of the limit surface if each polygon projects to approximately one pixel. This means that in practice, the base mesh is rendered without subdivisions for assets in the background. Therefore, subdivision surfaces algorithms can be used as LOD techniques, although they cannot be used for prefiltering a geometry beyond the scale of the base polygons.

Subdivision surfaces tend to smoothen all the sharp features of base polygon meshes (Fig. 6.3a). Artists often need to model surfaces with both smooth areas and sharp features. They have two main ways of preserving sharp features with subdivision surfaces:

- Adding tiny faces around sharp features (Fig. 6.3b).
- Tagging some edges as “sharp” and modifying the subdivision scheme so that it can preserve sharp features. This is often called *creasing* or *creased edges and vertices* [HDD*94,DKT98], and a weight can be given to edges to chose a level of sharpness (Fig. 6.3c).

The subdivision level can be adaptive for a given surface: each polygon can be seen as a *patch* that can be subdivided independently of other polygons, so that large base polygons are more subdivided than small base polygons. However, naïve implementations lead to cracks between patches that have

a different level of subdivision, and various methods are used to avoid this problem, such as specific subdivision pattern [Mor01,LPD14].

In the case of off-line rendering algorithms that compute all orders of light scattering, it is not trivial to know the appropriate level of subdivision for each surface. Some geometry is invisible from the camera but indirectly visible through shadows, reflections or refractions. Various strategies can be used, including *tessellation oracles* for predicting automatically the required level of subdivision, or simpler strategies based on distances to the frustum.

Seamless changes of resolution are not always implemented in production rendering pipelines. In this case, artifacts may appear when the camera or an asset move during a shot. To avoid these problems, some pipelines support the use of *tessellation cameras* to control the tessellation for an entire shot, in the same spirit as *dicing cameras*⁸ in the REYES rendering pipeline [CCC87].

6.1.4 Heterogeneous or inappropriate resolution

The goal of modeling artists is to quickly produce visually good models for close views, because this is how assets are reviewed, and because the precise position of the asset is not necessarily known in advance. Therefore, assets are not modeled at the appropriate resolution for a particular scene or shot, as discussed in Sec. 1.2. Quads can have very heterogeneous sizes, and for a given frame, some of the base quads can be smaller than pixels while others cover several pixels. In particular, this happens when artists use tiny faces for controlling subdivision surfaces (Fig. 6.3c). Base polygon meshes are sometimes *over-modeled*, meaning that most of the base polygons are much smaller than pixels at rendering.

When base polygons are smaller than the desired spatial resolution, memory usage can often be saved by simplifying the base meshes. However, we will see that surface-based prefiltering is not always sufficient. For example, artists usually model trees at the scale of leaves (often using procedural tools), and prefiltering sub-pixel leaves cannot be done accurately with mesh simplification techniques.

6.1.5 Displacement

The subdivided geometry is very often *displaced* using displacement maps (Sec. 3.3.2). Such maps often contain scalar values, and vertices are displaced along the surface normal. *Vector displacement* is also used for displacing vertices in arbitrary directions. In production pipelines, displacement maps are used to add details of small amplitude with respect to the size of the base polygon. For example, small-scale irregularity on a skin, or small bumps on a tree bark. Displacement maps can be painted directly by artists or generated by virtual sculpting software such as Zbrush⁹ or MudBox¹⁰.

When displacement maps have a high resolution compared to the level of geometric subdivision, they must be prefiltered before displacing vertices to avoid aliasing problems, such as the *swimming artifact* [NL13,SPM*13,LBG16]. Large features such as ears or horns are almost never modeled using displacement maps, because this approach raises frequent problems for animation and collision detection, both generally computed using the base polygons. Displacement is important in appearance-preserving methods since small perturbations change the perceived roughness of an object from a distance and have to be taken into account in coarse levels of geometric subdivision for consistent appearance.

6.1.6 Textures and materials

Texture maps are used to store all kinds of scattering parameters, including albedos, roughness parameters, directions of anisotropy, and weights or masks for combining several reflectance models. Many studios rely on uv-mapping but others like the Walt Disney Animation Studios rely on an independent mapping for each base polygon (PTex [BL08]).

⁸https://renderman.pixar.com/resources/RenderMan_20/attributes.html#dicing-strategy-camera

⁹<http://pixologic.com/>

¹⁰<https://www.autodesk.com/products/mudbox/overview>

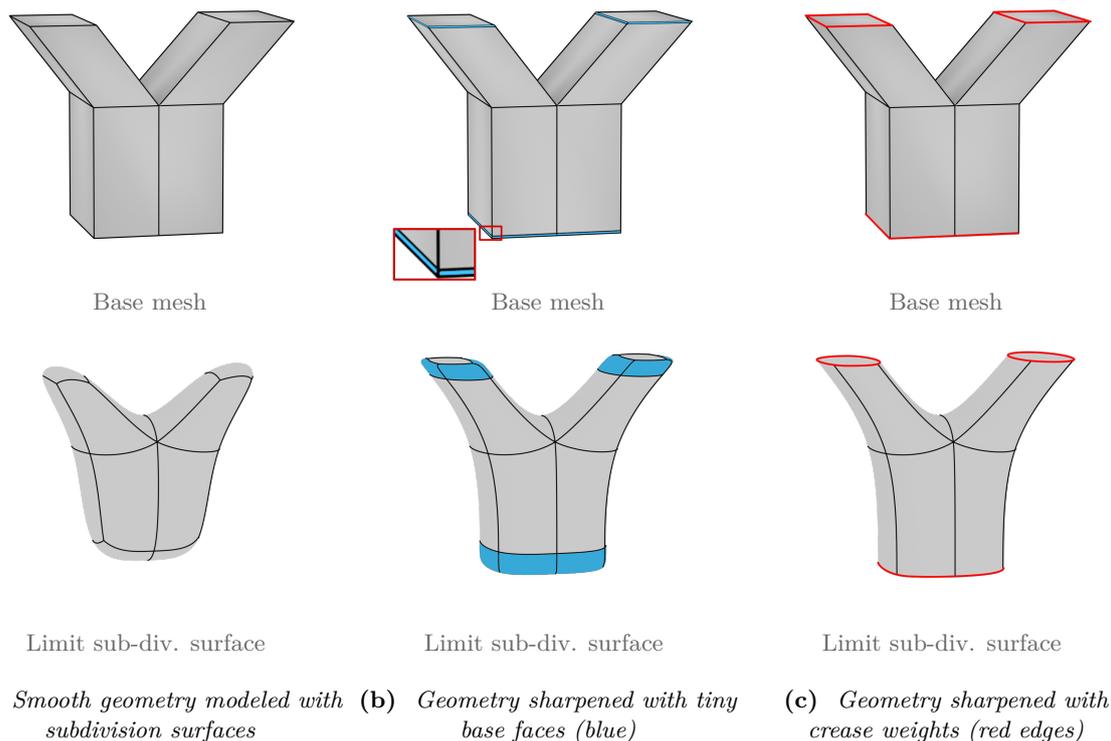


Figure 6.3: The limit surface of subdivision algorithm (here, Catmul-Clark) has no sharp features (a). Modeling artists use additional tiny face loops (b) or creasing (c) to preserve some sharp features.

Final materials for an asset are usually a combination of base materials. The most common base materials are the Lambertian reflectance model, smooth and rough dielectrics with microfacet models, smooth and rough conductors and emissive materials. Spatially varying materials are designed using textures but also procedural functions or other data such as distances to particles around the surface. Node-based graphical interfaces are very popular in the industry for the artistic authoring of materials. Examples of such interfaces can be found in the *Hypershade* material editor in Maya¹¹, in the Cycles renderer of Blender¹², in Nuke¹³ or in Mari¹⁴. Burley et al. introduced the principled BRDF [Bur12], an artist-friendly general material that only exposes to artists intuitive parameters for combining base materials, and reduces the problem of overly complex node graphs. This technique is currently used in Hyperion, the renderer of the Walt Disney Animation Studios, but also in other renderers such as RenderMan¹⁵.

6.1.7 Modeling practices

Assets and characters are made of lots of disconnected meshes that are manifold, with or without boundaries. Very often, each manifold surface is assigned an appearance based on the real-world object it models: wood, shiny metal, fabric, glass, plastic, etc. For example, a window will be modeled using a mesh for the frame with a wood appearance, and another mesh for the pane with a glass appearance, instead of using a single mesh with a glass material at the center and a wood material on the borders. Modeling mesh elements that have a single kind of appearance facilitates authoring, avoids some prefiltering problems and reduces the number of texture access at shading time.

¹¹<https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/Maya/files/GUID-EA5E2F6E-574A-4DBA-B3BE-01FDD5381FAD-htm.html>

¹²<https://docs.blender.org/manual/en/dev/render/cycles/introduction.html>

¹³<https://www.foxandcloud.com/products/nuke>

¹⁴<https://www.foxandcloud.com/products/mari>

¹⁵https://renderman.pixar.com/resources/RenderMan_20/PxrDisney.html

Disconnected meshes in assets and characters are almost always intersecting each other for avoiding light leaks at rendering. For instance, a building asset is typically made of many separated mesh elements for the walls, stairs, windows, ceilings and floors, and each element intersects some others. Similarly, the branches of a tree are often separated meshes that penetrate the mesh of the trunk, and hair primitives penetrate the geometry that models the skin.

6.1.8 Other rendering primitives: curves, volumes

Apart from textured meshes, off-line production renderers handle other rendering primitives such as segments or other 1D primitives, especially for hair modeling, or volumetric data, for example for rendering clouds, smoke and fire. In this part of the dissertation, we focus on prefiltering polygon meshes. We discussed previous work on hair prefiltering in Chap. 5, and volume downsampling is addressed in the last part of the dissertation (Chap. 11 to 14).

6.2 Using LODs in production pipelines

In the previous section, we gave an overview of the creation of textured meshes in the film industry. We now discuss practical problems raised by LODs in production pipelines and we list important requirements and useful properties.

6.2.1 Per asset, per shot and per frame precomputation

LODs methods can have a significant precomputation cost, especially when assets are very complex, because the data must be loaded and can be prefiltered with time consuming techniques, for example non-linear optimization (Sec. 4.4.2), random sampling for estimating integrals, or occlusion estimation [LBBS08]. Therefore, LODs must be precomputed at the right place in the pipeline, so that the precomputation cost is amortized for various rendering tasks and it does not put an additional burden on the pipeline.

For example, LODs can be precomputed per frame, per shot or per asset. Per-frame LOD computation is rather inefficient because a LOD can usually be used in many frames. Per asset precomputation is convenient because assets are often used in various shots. Moreover, in the modeling workflow, each asset is reviewed individually by directors or other artists with various lights and views, and the quality of LODs could be reviewed similarly (for various views, lights and scales). In the case of per asset LOD precomputation, the camera, that lights and the neighboring assets are not known. This context is known in per shot LOD precomputation, and it can be used for efficient prefiltering. For example, simplifying groups of neighboring assets can be more efficient than prefiltering each asset individually: in a dense forest with many overlapping plants, prefiltering a whole part of the forest can be more efficient than computing LODs for each plant.

6.2.2 LODs and instancing

Instancing consists in adding N instances of a mesh in a virtual scene without loading the geometry N times at rendering nor duplicating the geometry in memory. Each instance is described by its position in the scene and some transformations (scaling, rotations, deformations). The renderer computes ray intersections for all the instances using the same geometry. Instancing is very useful in scenes where lots of details are similar through the entire image: the leaves of a tree, the trees of a forest, the buildings in a city, etc. Instances may also have different reflectance attributes so that all instances look visually different even if the underlying ray-traced geometry is the same. For example, instanced leaves of a tree can have different materials based on their position in the foliage.

All the instances of a scene have the same geometry, that is the same resolution. This means that if

instances of a tree appear in both the foreground and the background, a high-resolution geometry must be used for all the instances. It is possible to create groups of instances depending on their positions with respect to the camera, and to use appropriate LODs for each group, but this reduces the benefits of instancing for reducing memory usage. However, creating groups of instances by level of detail has benefits in terms of noise. Therefore, there is a tradeoff between asset prefiltering and instancing, whose optimal solution depends on the scene, the complexity of the asset, the number of instances, and the bottleneck of the rendering algorithm.

It must be noted that LODs also tend to decrease the cost of ray intersections because they require less rendering primitives. However, these savings do not always bring significant speedup because the relationship between the cost of ray intersections and the geometric complexity is not linear. This is because rendering algorithms use highly optimized acceleration structures such as bounding volume hierarchies, and because ray intersections are not always the bottleneck compared to data loading and shading evaluations.

6.2.3 Useful properties of LOD methods for production pipelines

Seamless LOD transitions. Seamless transitions designate the possibility to change the resolution of an asset from the finest to the coarsest resolution without any brutal change of appearance (*popping artifacts*) in rendered images (Sec. 3.1.1). Seamless transitions are important when various resolutions of an asset are needed in the same shot, because popping artifacts are extremely perceptible. In particular, seamless transitions are required when assets are moving farther and farther from the camera, and when assets move and progressively go out of the camera frustum since popping artifacts could be visible in shadows or indirect reflections.

Automatic and robust. LOD methods should be as robust as possible. Artists' time is even more precious than computer resources, and LOD methods should ideally require no human intervention nor manual corrections.

Standard representations for the renderers. The representations used in LODs should be compatible with the existing rendering algorithms. For example, the primitives used in LODs should satisfy Helmholtz's reciprocity, especially for bidirectional rendering algorithms [LW93, Vea98].

Generality. Although strong assumptions on input assets allow for robust and efficient prefiltering, assets used in production are extremely diverse. Artists create trees with all kinds of branches, spatially-varying leaves, palms, flowers, fruits, lianas, etc. Buildings can have the appearance of medieval castles, Victorian houses or futuristic towers. Generality is very important and it is probably the most challenging requirement.

6.3 Types of appearance and efficient representations

One of the main questions in appearance prefiltering is the choice of representations used in the LODs, *i.e.* the rendering primitives such as polygons (Sec. 3.1), surfels (Sec. 3.2.1), implicit surfaces (Sec. 3.2.2), voxel-based geometry (Sec. 3.2.3) or volumes (Chap. 5), and the scattering models assigned to these primitives such as BSDFs (Sec. 4.1.5) and phase functions (Sec. 5.3). Some representations are more appropriate than others, because they approximate more accurately the appearance of the input data at some scales, because they have a better memory efficiency or because they support seamless transitions. In this section, we discuss the choice of the representations for prefiltering textured meshes at arbitrary scales.

6.3.1 Distant appearance of textured meshes

Textured meshes are used for representing all kinds of things, from simple surfaces to more complex appearance such as the foliage of pine trees. Depending on the distance between the asset and the camera, a pixel can either *see* (that is, capture the radiance from) a single polygon covering the pixel entirely (Fig. 6.4a), several connected polygons forming a simple surface (Fig. 6.4b), small polygons that do not cover entirely the pixel (Fig. 6.4c) or a large number mesh elements which we refer to as *sub-resolution geometry* (Fig. 6.4d). The appearance of simple surfaces is quite different from the appearance of a foliage viewed from a distance. The former can be characterized by a normal and a surface reflectance model, and the later does not have a well-defined normal and is better described by a probability of occlusion, *i.e.* how much the geometry attenuates light and covers pixels.

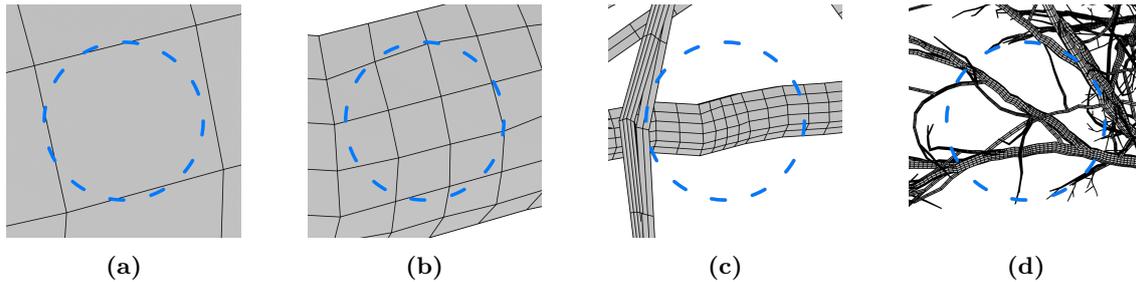


Figure 6.4: *The appearance of textured meshes depends on the scale at which they are observed, represented with blue circles. (a): In close views, the appearance is due to one or a few polygons. (b): At this scale, the appearance is due to a simple surface made of connected polygons. (c): For distant views, connected polygons do not cover pixels, leading to a semi-transparent appearance. (d): Semi-transparent appearance due to a lot of sub-resolution geometry.*

If we move progressively an asset from the foreground to the background, the appearance seen by the pixels changes continuously from a surface-like appearance to a semi-transparent appearance, until the entire asset becomes sub-pixel. This means that an all-scale prefiltering method should support various types of appearance with seamless transitions. Ideally, the input high-resolution textured mesh should be used as the finest of the LODs when the asset is very close to the camera.

In Chap. 3, we have seen that several LOD methods have a limited domain of validity because they assume that the appearance of the asset in each pixel is always a surface-like appearance. For example, mesh simplification methods or the contour plane method (Sec. 3.2.3, [LK10a]) cannot be used for prefiltering accurately foliage for distant views, because what pixels see is quite different from the appearance of simple opaque surfaces, and these methods do not preserve accurately the probability of occlusion. In other words, these methods should only be used as long as the asset is made of simple and large enough surfaces compared to the chosen prefiltering scale. In this part of the dissertation, we address the problem of all-scale prefiltering and we do not want to assume that pixels only see simple surfaces.

6.3.2 Identifying important characteristics of appearance

Before obtaining the results presented in the next chapters, we experimented various LOD methods using either volume-based representations or surfels as primitives for prefiltering all kinds of appearances, and we did not obtain satisfying results: volumes were inaccurate for representing large surfaces with accurate shading, and surfels lead to artifacts for both large surfaces and foliage-like geometry. We found that these rendering primitives are not appropriate for prefiltering all kinds of appearances, and that accurate prefiltering requires to understand what characterizes the distant appearance of an asset in order to preserve it using the best representation.

When an asset is moving from the foreground to the background, pixels see continuous changes of appearance. It is difficult to define precise and unambiguous types of appearance, and there are necessarily

transition scales. Nevertheless, from our tests, we have drawn the conclusion that the scattering of light at a given resolution can be roughly classified in two categories, each with specific prefiltering constraints:

- Assets are often made of large, simple enough, watertight surfaces (without *light leaks*) that cover several pixels. We found that these surfaces must be prefiltered using accurate and watertight surface representations, and with accurate surface reflectance models.
- In the case of sub-resolution geometry such as foliage seen from a distance, the amount of occlusion of the sub-resolution details is an important part of the perceived appearance, because it characterizes the shadows, how much the background is seen through the asset, and how deep light can penetrate inside the asset (for example in a foliage). Therefore, intricate details must be prefiltered with correct local occlusion, which is not the purpose of surface-based prefiltering methods.

6.3.3 Automatic analysis of complex geometry

Because the distant appearance of 3D assets can be classified in two categories with different prefiltering requirements, a solution is to prefilter them separately using appropriate methods and primitives for each one. Therefore, we need a tool for automatically analyzing input assets and classifying the geometry in two groups: the geometry that must be prefiltered using a watertight surface appearance, and the geometry that must be prefiltered with a correct local occlusion. This automatic analysis is actually a difficult problem, for two main reasons:

- As explained above, there is a continuity between surface appearance and semi-transparent appearance, and there are always ambiguous cases where it is unclear if the appearance should be prefiltered as a surface or as sub-resolution geometry.
- A surface-like appearance can be due to a large watertight mesh (Fig. 6.5a), but also to smaller disconnected elements, such as flat tiles on a roof (Fig. 6.5b). The former case is easier to detect automatically and we address this problem in the next chapter (Chapter 7). The second case is more difficult, because large scale watertightness is not something that can be detected with a local analysis of manifold geometry. This case remains an open problem, nor only for the detection but also for the robust prefiltering of such surfaces.

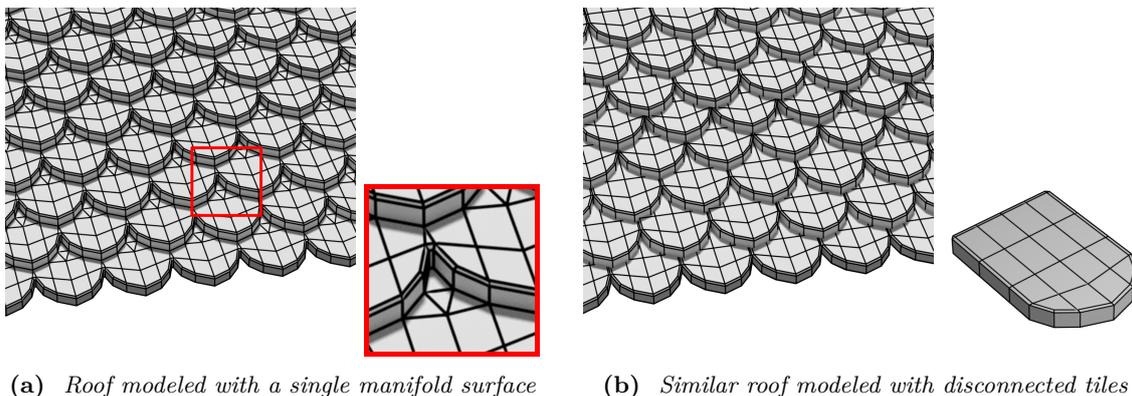


Figure 6.5: *Surface-like appearance from a distance can be due to either large-scale manifold surfaces (a) or assemblies of disconnected elements (b). Prefiltering manifold surfaces is much simpler, using mesh simplification and well-posed material prefiltering in surface space. Surface appearance due to disconnected geometry is more challenging for automatic detection and robust prefiltering.*

6.3.4 Prefiltering surface-like appearances

Once a surface-like appearance has been detected in an asset, at a given prefiltering scale, it must be prefiltered using appropriate rendering primitives and scattering functions. In the next chapters, we rely

on meshes for prefiltering surface-like appearance, because they fulfill all our requirements: they can model watertight surfaces without artifacts, they are memory efficient and they can be rendered very efficiently. Because most assets are modeled with large manifold meshes (Fig. 6.5a), prefiltering such surfaces using meshes reduces the prefiltering problem to a mesh simplification problem with surface material prefiltering. Moreover, mesh simplification is a well studied problem and this framework supports seamless transitions between discrete LODs (Sec. 3.1). The problem of prefiltering surface-like appearance modeled with disconnected elements (Fig. 6.5b) is much more challenging as it raises the problem of finding a good surface approximation, with a proper orientation and appropriate front and back materials, which are rather ill-posed problems with many ambiguous cases.

Meshes are not the only potential representation for modeling surface appearance. As we mentioned above, we experimented surfels in early tests and found that they suffer from several drawbacks compared to meshes:

- Surfels must be large enough and overlap for avoiding light leaks, which leads to rendering artifacts on curved surfaces (an example can be found in the work of Wald et al. [WS05]) but also on flat surfaces because surfels are perfectly superimposed, which raises ray casting issues (false shadowing between neighboring surfels and unstable ray intersections with surfels).
- With surfels, the knowledge of manifold surfaces is lost, and material prefiltering becomes ambiguous. For example, prefiltering front and back materials is well-posed for manifold surfaces but not for surfels. For this reason, prefiltering surfaces from the previous LOD would be difficult.
- Seamless transitions are more difficult to achieve using surfels and risks of light leaks appear when morphing large surfaces made of many surfels.

Among other surface representations, implicit surfaces based on point sets or distance fields (Sec. 3.2.2 and 3.2.3) have less flexibility than meshes for prefiltering. In various implicit surface representations, only a single surface can exist locally, and the smallest distance between two surfaces depends on the size of the reconstruction kernel for point set surfaces [ABCO*01] or the resolution of the distance field [HN12]. Representing geometry such as local intersections of large surfaces or close parallel surfaces would require various superimposed implicit surface definitions, which would increase storage and rendering costs. Although ensuring a single manifold surface locally seems to be a good property for LODs, it makes material prefiltering ill-posed, for the same reasons as for prefiltering disconnected elements using a single surface. In the case of large-scale parallel surfaces with, for instance, a glass material for the first surface and a diffuse material for the second one, prefiltering becomes difficult with a single surface in the LOD. To the contrary, preserving two surfaces allows for well-posed and robust prefiltering of each of the manifold surfaces, and a correct appearance of the LOD.

Heterogeneous participating media cannot be used for representing accurately a surface-like appearance at a large scale. Volumes are not watertight, unless density parameters in voxels are high, which in turn creates inaccurate thick silhouettes (Fig. 6.6). Moreover, it is difficult to preserve precisely the appearance of a surface using heterogeneous participating media because multiple scattering in the volume affects the perceived color and always scatters light diffusely. Volumes cannot represent several surfaces locally, and are not the most appropriate representation for view-dependent reflectance.

6.3.5 Prefiltering sub-resolution intricate geometry

As we mentioned above, prefiltering sub-resolution geometry requires preserving the local occlusion. This can be done by measuring the local occlusion of the input geometry, for example based on local ray casting as proposed in Chapter 9. Then, it is possible to use volumes, surfels or small polygons to preserve a correct local occlusion in the LOD, by estimating appropriate density parameters or surface areas for the prefiltered representations.

The microflake volume model introduced by Jakob et al. [JAM*10a] (Sec. 5.3.2) combined with the SGX distribution [HDCD15] allows for rendering foliage-like and fiber-like volumetric appearance with relatively low storage requirements, especially if sparse voxel grids are used. Microflakes volumes are

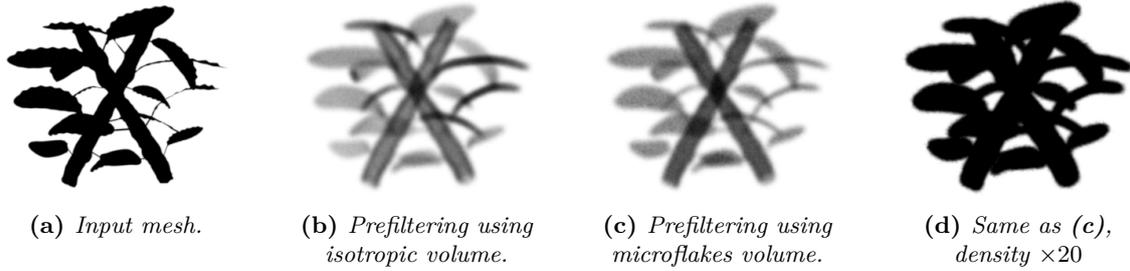


Figure 6.6: Several volumetric approximations (b,c,d) of a mesh containing large watertight surfaces with respect to the prefiltering scale (a). Volumes are rendered with approximately one voxel per pixel. From the point of view of the occlusion, heterogeneous participating media cannot be used for prefiltering such geometry because this representation fails to preserve both watertightness (b,c) and local occlusion (d). (a): Input high-resolution mesh. (b): Isotropic participating medium approximation with preservation of the mean local occlusion. (c): Using an anisotropic participating medium model (microflakes model). (d): Increasing density parameters preserves watertightness but leads to incorrect local occlusion and silhouettes. Rendered with volumetric path tracing in Mitsuba [Jak10].

characterized by microflake normal distributions and microflakes albedos, allowing for prefiltering directly sub-resolution polygons from their individual normals and albedos. On the other hand, from a pipeline point a view, polygons and surfels are more convenient in the case of overlapping or animated assets, compared to volume representations. In Chapter 9, we address the problem of estimating volume scattering parameters given some intricate geometry.

6.4 Our new framework for generating hybrid LODs

From the observations discussed in the previous section, we propose a pipeline for generating appearance-preserving LODs. As shown in Fig. 6.7, we propose a cascaded LOD generation based on an automatic surface appearance analysis, and separate prefiltering for large surfaces and sub-resolution intricate geometry. The cascade aims at enabling seamless transitions in the mesh simplification framework, as well as reducing the prefiltering cost by analyzing the mesh of the previous LOD instead of the input mesh.

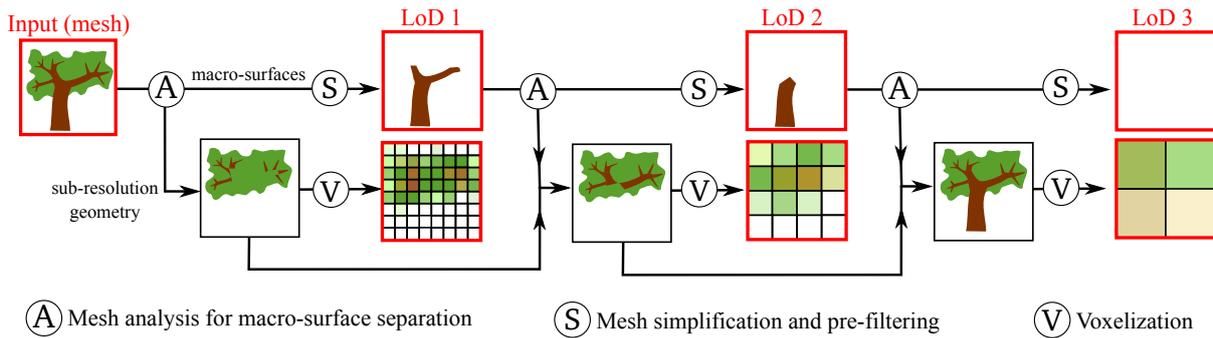


Figure 6.7: We propose a framework for generating all-scale appearance-preserving LODs. Starting from an input textured mesh, we analyze and separate large-scale surfaces (or macrosurfaces) from the rest of the geometry using our separation criterion (step A, Chap. 7). Macrosurfaces are prefiltered until they match the target resolution (step S, Chap. 8). The rest of the geometry, i.e. sub-resolution geometry is prefiltered independently of large surfaces (step V, Chap. 9). Each output LODs consists of the prefiltered macrosurfaces and prefiltered sub-resolution geometry (in this case, voxels). They have to be rendered jointly. Our cascaded LOD framework allows for seamless transitions.

6.5 Summary of the chapter

In this chapter, we introduced various ideas that led us to the hybrid prefiltering method whose parts are presented in the next chapters. Our main points were the following:

- Production assets can be assumed to have several properties that help prefiltering them. In particular, they are made of assemblies of clean manifold surfaces.
- On the other hand, production assets have very heterogeneous polygon sizes and can have both large surfaces and intricate details.
- The distant appearance of assets can be classified in two main categories: large-scale watertight surface-like appearance, and semi-transparent appearance due to intricate details.
- These kinds of appearance require different prefiltering methods: for the former, a watertight surface representation is needed with correct surface material, for the later the local occlusion must be preserved in the LOD.
- Among the existing surface representations for prefiltering and rendering large watertight surfaces, meshes are the most appropriate representation and allow us to rely on previous work on mesh simplification with seamless transitions.
- Prefiltering sub-resolution geometry can be done using various representations, including the microflakes volume model which can be used for rendering sub-pixel foliage-like and fiber-like appearances.
- We proposed a LOD pipeline with automatic analysis of large surfaces and sub-resolution geometry for all-scale appearance-preserving LODs.

In the next chapters, we address some of the problems identified in this chapter so that our prefiltering pipeline can be implemented: we address the automatic analysis of macrosurfaces in chapter 7, the prefiltering of macrosurfaces in chapter 8, and the prefiltering of intricate geometry in chapter 9. Results of our method can be found in chapter 10.

6.6 Résumé du chapitre

Dans ce chapitre, nous avons introduit plusieurs idées qui ont mené à la conception d'une méthode hybride pour le préfiltrage. Les différents éléments de cette méthode sont détaillés dans les prochains chapitres. Les idées principales de ce chapitre étaient les suivantes :

- Les objets 3D utilisés en production ont certaines propriétés spécifiques qui nous aident à les préfiltrer. En particulier, ils sont faits de surfaces de topologie simple et leurs polygones sont agencés de manière cohérente.
- D'un autre côté, les objets 3D de production peuvent avoir des tailles de polygones très hétérogènes, c'est à dire qu'on peut y trouver à la fois de grands polygones et de tout petits détails géométriques.
- Les apparences distantes de ces objets peuvent être classées en deux grandes catégories: les apparences des larges pans de surfaces étanches, et les apparences semi-transparentes dues à de la géométrie sous-résolution.
- Ces sortes d'apparence nécessitent des méthodes de préfiltrage différentes : pour les grands pans de surfaces, il faut utiliser une représentation surfacique étanche et des modèles de réflectance appropriés, tandis que pour la géométrie sous-résolution il faut surtout préserver la transparence locale dans les LODs.
- Parmi les représentations surfaciques existantes pour le préfiltrage et le rendu, les maillages sont les plus pratiques et ils nous permettent de nous appuyer sur de nombreux travaux de simplification de maillage et de transitions imperceptibles entre les niveaux de détail.

- Pour préfiltrer de la géométrie complexe sous-résolution, plusieurs solutions peuvent être envisagées, et en particulier les modèles volumiques de microflakes qui peuvent être utilisés pour le rendu de milieux anisotropes comme les ensembles de fibres ou les feuillages.
- Nous proposons une approche de LOD hybride qui repose sur une analyse automatique des macrosurfaces et de la géométrie sous-résolution pour le préfiltrage à des échelles arbitraires.

Dans les prochains chapitres, nous nous pencherons sur plusieurs problèmes identifiés dans ce chapitre, de manière à pouvoir implémenter entièrement notre approche hybride : nous traiterons le problème de l'analyse automatique de macrosurfaces dans le chapitre 7, le préfiltrage des macrosurfaces dans le chapitre 8, et le préfiltrage de la géométrie complexe dans le chapitre 9. Les résultats obtenus avec notre approche seront présentés dans le chapitre 10.

Chapter 7

A new algorithm for detecting macrosurfaces

In this chapter, we address the problem of analyzing the geometry of an asset for detecting automatically surface-like appearances and sub-resolution intricate geometry, in order to prefilter them accurately using appropriate algorithms and representations. We show in this chapter that detecting surface-like appearances due to large manifold surfaces (Fig. 6.5a) can be achieved with a local analysis of a mesh. As mentioned in the previous chapter, detecting surface-like appearances due to disconnected elements (Fig. 6.5b) is much more challenging. For example, it is difficult to robustly detect that an assembly of disconnected tiles forms a large watertight surface, using local geometric information or local occlusion measurements.

We only address the problem of detecting manifold surfaces, which are a lot more frequent in production assets than surface-like appearance made of disconnected elements. We use the term *macrosurface* for designating manifold surfaces that are large compared to a given prefiltering scale. In this chapter, we propose a definition of macrosurfaces for our hybrid LOD method and we propose practical solutions for implementing a macrosurface analysis algorithm.

7.1 Definition of macrosurfaces

7.1.1 Ambiguities and requirements

Our macrosurface definition aims at separating a given geometry into large-scale surfaces and sub-resolution geometry. Of course, ambiguities appear because there are necessarily transitions scales where surfaces are not large but not clearly sub-resolution either. Nevertheless, some cases are not ambiguous:

- Tiny cylinders (Fig. 7.1a) and ribbons (Fig. 7.1b) are clearly not macrosurfaces and should not be prefiltered using large opaque surfaces.
- Tiny disconnected elements should be robustly classified as sub-resolution geometry (Fig. 7.1c and 7.1d)
- To the contrary, large smooth surfaces are clearly macrosurfaces (Fig. 7.1e) and should be robustly identified as such, as well as large surfaces with small perturbations (Fig. 7.1f).

Ambiguous cases include:

- Surfaces with relatively large perturbations with respect to the prefiltering scale. Their appearance is similar to the one of sub-resolution cylinders (Fig. 7.1g).

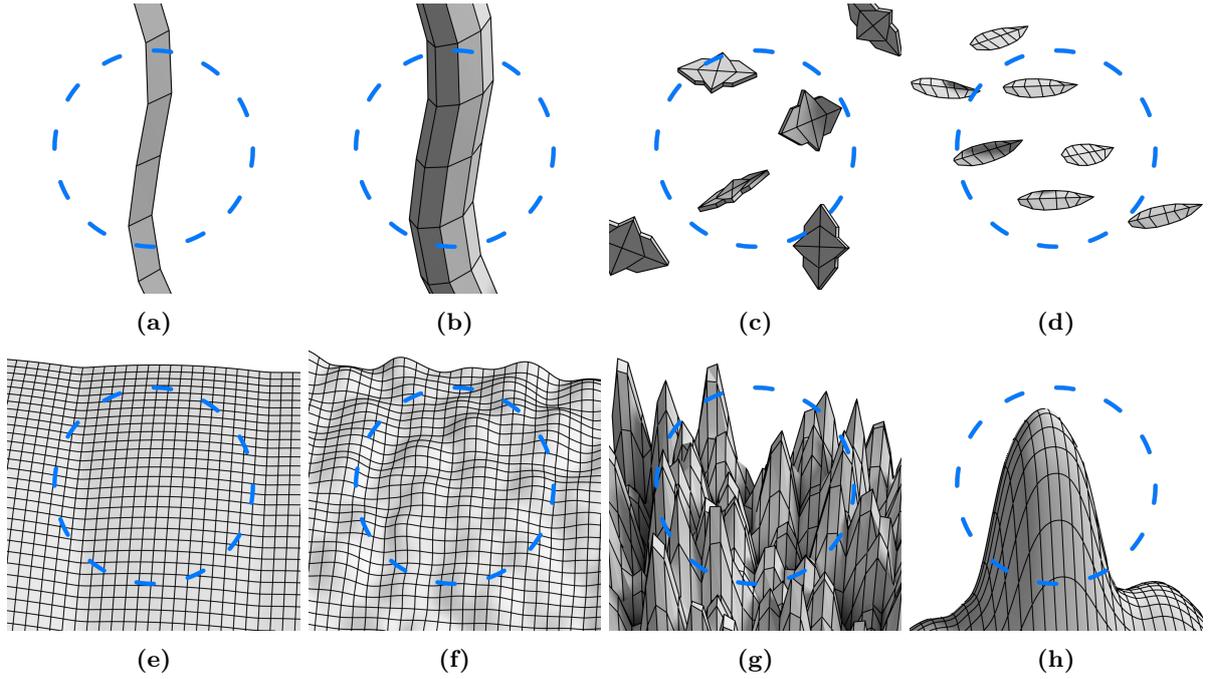


Figure 7.1: (a,b,c,d): Non-ambiguous cases of sub-resolution geometry that do not have the appearance of macro-surfaces at the prefiltering scale (blue circles). (e,f): Non-ambiguous macro-surfaces at the prefiltering scale. (g,h): Ambiguous cases with highly curved surfaces that are between simple surfaces and cylinder-like sub-resolution geometry.

- Highly curved surfaces that do not cover pixels (Fig. 7.1h).

The goal of our definition is not to overcome ambiguities but to robustly detect cases that are not ambiguous. In our implementation (Sec. 7.2), we rely on a cleaning step from improving the quality of the analysis.

7.1.2 Our definition of macro-surfaces

We propose a definition that classifies points on a manifold surface into two groups, depending on the connected neighborhood of each point: points belonging to macro-surfaces, and points belonging to sub-resolution geometry.

Let x be a point on the mesh M and r the target spatial resolution, which gives the scale of the macro-surface analysis. Let S be the sphere of radius r centered in x , and B the closed ball bounded by S (Fig. 7.2a). We call I the set of points $p \in (M \cap S)$ that are connected to x in $M \cap B$. We say that x belongs to a macro-surface if I has *one single connected component*.

7.1.3 Examples

To have more intuitions about the behavior of our macro-surface criterion, we can look at what happens in two simple cases:

- I is empty for all points on any disconnected geometry that is small compared to S (Fig. 7.2d). Our criterion states that these points on such small elements do not belong to a macro-surface.
- For all points on cylinders whose radius is small compared to S , I has two connected components, as shown in Fig. 7.2c, meaning that such cylinders are analyzed as sub-resolution geometry.

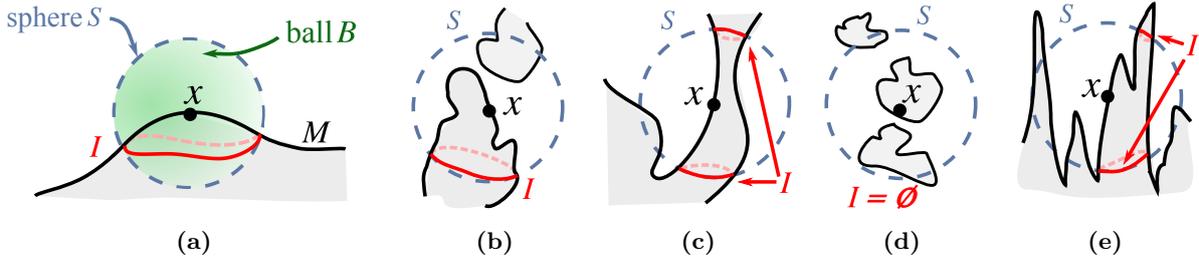


Figure 7.2: Several geometries including non-ambiguous macrosurfaces (a), sub-resolution geometry (c,d) and ambiguous cases (b,e). Our macrosurface analysis at a point x on a surface M depends on the number of connected components of I defined as the set of points $p \in (M \cap S)$ that are connected to x inside the ball B . (a,b): I has a single component, our criterion detects a macrosurface around x . (c,d,e): I is empty or has more than one component. The point x is analyzed as not belonging to a macrosurface.

Other examples are illustrated in Fig. 7.2. Note that our criterion does not analyze as sub-resolution geometry the boundaries of large surfaces, which is a good property, as such borders are indeed part of a large watertight surface.

7.2 Implementation

In this section, we propose an implementation of a macrosurface detection algorithm based on the criterion introduced in the previous section, with a method for efficiently evaluate the criterion for one point of the input surface, and a cleaning step for improving the behavior of the algorithm in ambiguous cases.

7.2.1 Evaluating our macrosurface criterion

Our definition of macrosurfaces is based on the connected neighborhood of a point on a mesh. For a given point, it requires the computation of the number of intersections of connected polygons in the neighborhood of a point with a sphere, and to find the topology of the intersection (the number of connected component of the intersection). Therefore, efficient evaluation requires fast access to neighboring faces, edges, points, etc. The half-edge data structure [BSBK02] is appropriate since access to adjacent elements can be done with constant time.

We propose a strategy for exploring efficiently the connected neighborhood and computing the topology of I . We assume without loss of generality that the input mesh of the analysis is a triangle mesh, as other polygons can be triangulated for this computation. Our algorithm consists in a local exploration of the faces connected to x inside the sphere of radius r and center x . When we find an intersection between a face and the sphere, we store the information in order to retrieve the topology of I once all the local faces have been explored. Sphere-triangle intersections lead to various cases presented in Fig. 7.3. More precisely, our algorithm has the following steps:

1. Create a list of faces to explore, which is initialized with the face that contains the point x .
2. While there are faces to explore:
 - Choose one face in the list, look for intersections with the sphere S . The different cases are shown in Fig. 7.3.
 - If the face intersects the sphere, store some information about the intersection.
 - Add neighboring faces that have not been explored yet and that are connected to x inside S (Fig. 7.3, green arrows).

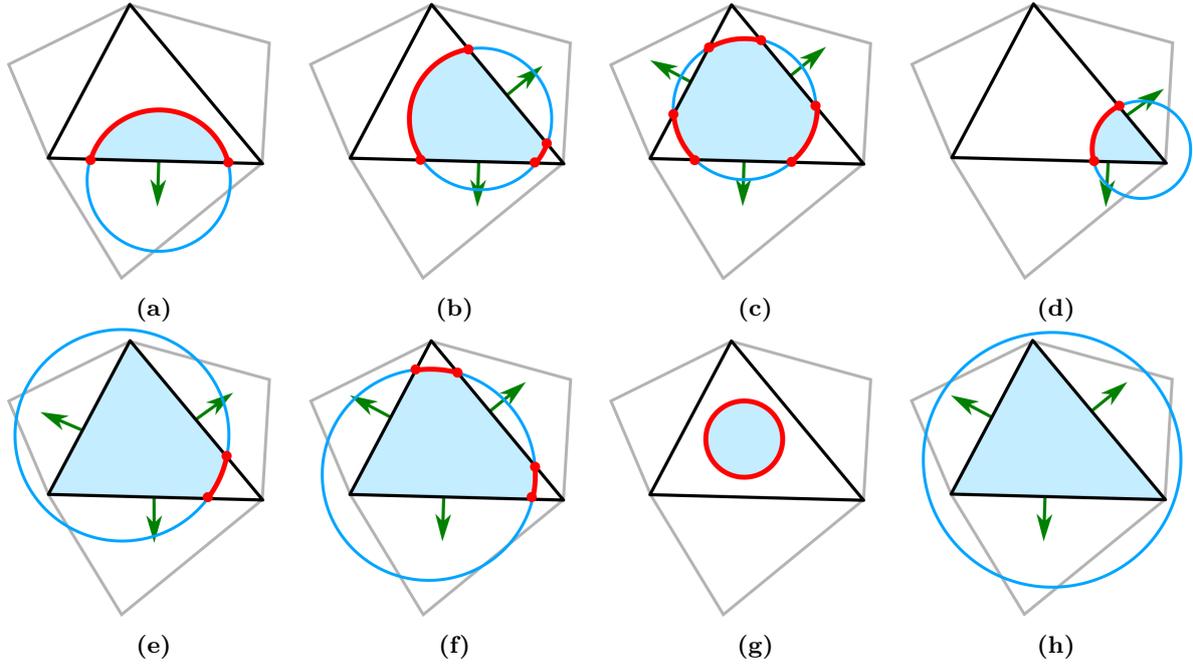


Figure 7.3: For evaluating our macrosurface criterion, we look for intersections between triangles and the sphere S . Our algorithm stores information about these intersections (red parts) and then explores neighboring faces that are connected inside the sphere. Neighboring faces that must be explored are shown with green arrows.

3. Finally, find the topology of I from all the intersections that have been found. If I has a single connected component, x belongs to a macrosurface, else x belongs to sub-resolution geometry.

The main idea of the algorithm is to ensure that explored faces are always connected to x inside the sphere S . For this reason, we start with the face that contains x and only add to the exploration list faces that have points connected to x inside S (Fig. 7.3). A pseudo code of the algorithm can be found in Appendix A.

7.2.2 Per-triangle evaluation and cleaning step

Our macrosurface criterion is defined for arbitrary points on a surface. In practice, we have some input meshes and we want to classify their polygons into macrosurfaces or sub-resolution geometry. In our implementation, we decided to evaluate our criterion once per triangle, for example using the center of mass as evaluation point, or the middle of the longest edge, which performs better for elongated triangles. After the analysis of the entire mesh, triangles are identified as parts of a macrosurface or as sub-resolution intricate geometry as shown in Fig. 7.4a. In the case on highly curved surfaces or cylinders, our method analyzes as macrosurfaces the endings, as shown in Fig. 7.2b and 7.4a. Although these cases are ambiguous, we found that in some cases they are best prefiltered as sub-resolution intricate geometry, and that a cleaning step could help classifying them more robustly. Our cleaning pass is the following: for each connected piece of potential macrosurfaces, we check that the bounding sphere is larger than S . If this is the case, we classify this geometry as a macrosurface, else we classify it as sub-resolution geometry. Indeed, groups of connected faces that do not satisfy this criteria cannot cover more than one pixel. This is illustrated in Fig. 7.4b.

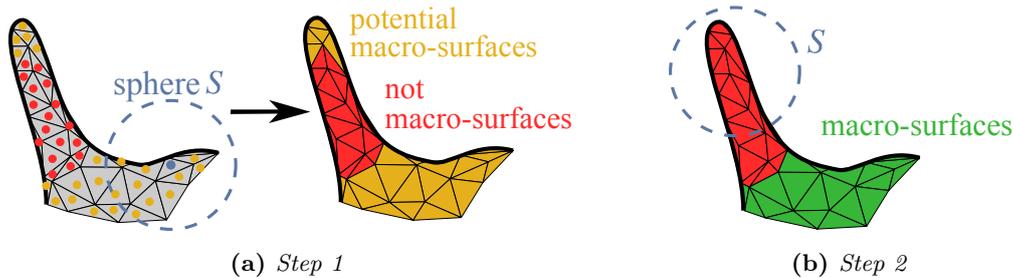


Figure 7.4: *Our two steps macrosurface analysis. (a): Detecting potential macrosurfaces (orange). Our criterion uses spheres whose radius is equal to the target resolution and whose centers are on the surface (Sec. 7.1.2). (b): Keeping only large connected surfaces (green). As the upper part of the bump is smaller than S , it is analyzed as sub-resolution geometry after our cleaning step.*

7.3 Results and discussion

Results on complex meshes are shown in Fig. 7.5 for different scales of analysis. They show that our algorithm robustly detects cylinder-like sub-resolution geometry and large surfaces. Ambiguities can be seen close to borders at some scales since the geometry is locally ribbon-like.

The cost of our analysis for a given point x depends on how many triangles are connected to x in B . Our implementation is more expensive if the mesh is very detailed compared to S because lots of triangles have to be explored. Fortunately, in our filtering pipeline (Fig. 6.7), the average triangle size increases at each LOD and their edge lengths are approximately $\frac{\epsilon}{2}$. Moreover, only macrosurfaces from the previous LOD have to be analyzed. Our implementation is therefore efficient for our pipeline. Our algorithm easily supports parallel implementations since each triangle analysis is independent of the others. In our implementation, the mean cost of the analysis per triangle is around $15 \mu s$ if the mesh comes from the previous LOD (the analysis of the input mesh highly depends on its resolution compared to the first target resolution). We used the OpenMesh library [BSBK02, Ope] and the half-edge data structure for efficient local explorations of meshes.

The main limitation of our approach is that it only detects macrosurfaces due to connected and smooth enough meshes. It cannot detect large scale watertightness due to disconnected elements or very rough surfaces (Fig. 7.2e). Our method could probably be improved by additional cleaning steps. For example, the small ambiguous regions on the leaves in Fig. 7.5 at transitions scales could be changed to macrosurfaces when these regions are small compared to S and surrounded by macrosurfaces.

7.4 Summary of the chapter

In this chapter:

- We proposed a criterion for classifying points on manifold surfaces into either points belonging to macrosurfaces or points belonging to sub-resolution intricate geometry.
- We discussed the implementation of our macrosurface analysis for separating meshes into macrosurfaces and sub-resolution geometry, for a given prefiltering scale. We proposed an additional cleaning step for improving the quality of the separation.
- Our method robustly detects sub-resolution cylinder-like and ribbon-like geometry, as well as large, smooth enough, watertight surfaces. It does not detect surface-like appearances due to disconnected geometry, which remains an open problem.

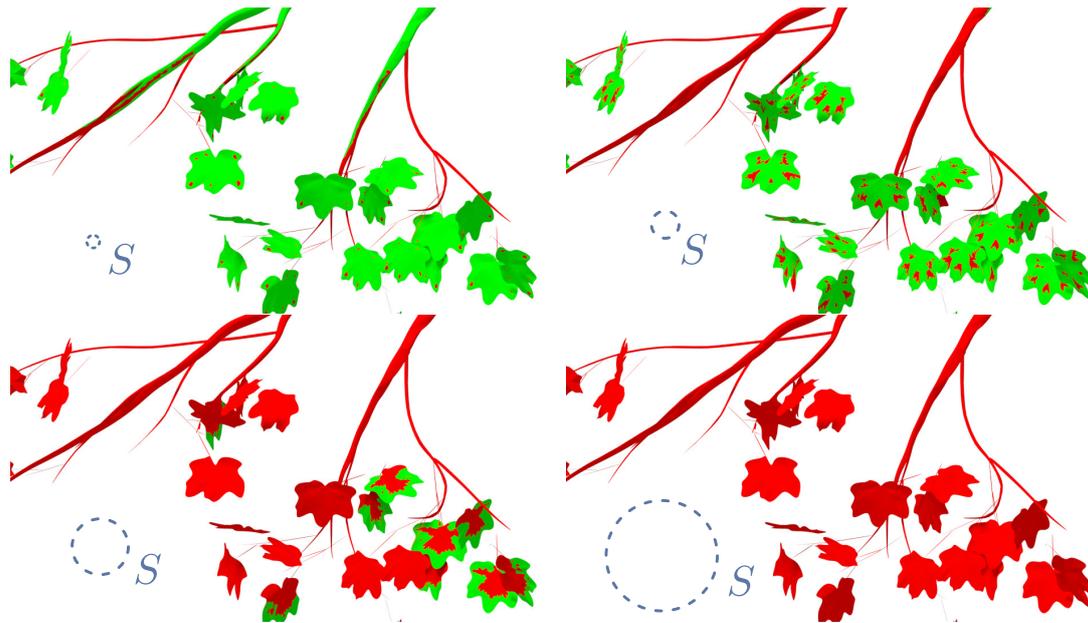


Figure 7.5: Results of our mesh-based macrosurface analysis with four different resolutions: blue circles represent the size of sphere S (Sec. 7.1.2) whose radius is equal to the target spatial resolution in the LOD. Green triangles are triangles analyzed as belonging to macrosurfaces by our algorithm. Top left: leaves and big twigs are macrosurfaces compared to the target resolution. Bottom right: the geometry has no surface wider than the target resolution.

7.5 Résumé du chapitre

Dans ce chapitre :

- Nous avons proposé un critère pour classer les points de surfaces comme des points appartenant à des macrosurfaces ou comme des points appartenant à de la géométrie complexe sous-résolution.
- Nous avons proposé un algorithme pour détecter les macrosurfaces d'un maillage à une échelle donnée, et commenté une étape supplémentaire de nettoyage pour améliorer la robustesse de l'algorithme.
- Notre méthode détecte de manière fiable les géométries assimilables à des cylindres ou des rubans sous-résolution, ainsi que les grands pans de surfaces. Elle ne détecte pas les apparences de surfaces dues à de la géométrie déconnectée, ce qui reste un problème non résolu.

Chapter 8

Resolution-based mesh simplification with material prefiltering

In previous chapters, we have seen that the geometry of an asset can be automatically separated into macrosurfaces and sub-resolution intricate geometry for a given prefiltering scale, and that prefiltering these two kinds of data requires different techniques: watertight surface representations for macrosurfaces, and occlusion-preserving techniques for sub-resolution geometry. In this chapter, we address the problem of prefiltering macrosurfaces given a target spatial resolution. Assuming that the input meshes are macrosurfaces at the desired prefiltering scale, we know that we are in the domain of validity of mesh simplification algorithms such as the edge collapse simplification algorithm (Sec. 3.1). As discussed in previous chapters, we rely on mesh simplification for several reasons: meshes are efficient as prefiltering primitives, some mesh simplification algorithms support seamless transitions, and many authors have studied accurate and robust mesh simplification so that we can build on their work.

Despite the large amount of previous work on geometric LODs, we had to adapt previous mesh simplification methods to satisfy our needs:

- We want to prefilter both the geometry and the materials. Most previous methods focus either on texture-space prefiltering (Chap. 4) or pure geometric simplification (Chap. 3). Some methods have been proposed for taking into account materials during the simplification [COM98, Hop99]. What we want instead is to prefilter materials at any scale, independently of the geometric simplification and of *uv* patches. In other words, we don't want the materials to be a constraint for the geometric simplification, we rather want to prefilter accurately the materials for arbitrary geometric coarsening operations.
- Most mesh simplification algorithms are *budget-based*, meaning that elements are iteratively deleted until a given vertex count or face count is reached, or *fidelity-based*, meaning that the mesh is simplified as long as the produced error (*e.g.* a geometric error or other quality metrics) does not exceed a chosen maximum error (Sec. 3.1.1). What we need instead is a *resolution-based* simplification algorithm, that is, an algorithm that runs until it is not possible to apply new coarsening operations without breaking resolution constraints. For example, the size of created polygons that should not be larger than the target spatial resolution.

In this chapter, we revisit two mesh simplification algorithms, the edge collapse algorithm [Hop96] and the face loop collapse algorithm for quad meshes [DSSC08]. We introduce several modifications so that these algorithms can be used for resolution-based macrosurface simplification with well-posed material prefiltering.

8.1 Requirements

In this section, we analyze various requirements for mesh simplification algorithms in our context, based on the conclusions drawn in previous chapters.

8.1.1 Resolution-based simplification and stopping criteria

As explained in Sec. 3.1, mesh simplification methods are characterized by coarsening operations, which are ordered using some metrics that quantify how much they affect the quality of the mesh. Operations that have a minimum impact on the accuracy of the mesh are typically applied first. These algorithms also require a criterion to stop the simplification when the desired level of simplification is reached.

In our case, we need a resolution-based stopping criterion: surfaces should not be simplified beyond the target spatial resolution for the LOD. A convenient strategy is to ensure that simplified polygons are never larger than the target resolution (even if this would be acceptable on perfectly flat surfaces). Using this strategy, it is sufficient to have per-face reflectance parameters in the LODs since prefiltered faces are always sub-pixel.

It is possible to add fidelity-based constraints to our resolution-based stopping criterion. In some of our implementations, we prevented the mesh simplification algorithm from creating a simplified surface whose distance to the input surface is larger than the target resolution, because such geometric error could be visible in pixels. We relied on a two-sided distance between the input surface M_1 and the simplified surface M_2 , as discussed in Sec. 3.1.4:

$$\text{dist}(M_1, M_2) = \max \left(\max_{x \in M_1} d(x, M_2), \max_{y \in M_2} d(y, M_1) \right)$$

with

$$d(x, M) = \min_{y \in M} \|x - y\|_2.$$

The constraint $\text{dist}(M_1, M_2) < \xi_{\text{tol}}$ ensures that all points of M_1 are at a distance of ξ_{tol} or less from M_2 , and that all points of M_2 are at a distance of ξ_{tol} or less from M_1 . The limit distance ξ_{tol} can be chosen depending on the target resolution for preventing visible errors in pixels. This quality constraint limits (often slightly) the geometric simplification in curved regions, but it leads to most robust LODs.

8.1.2 Seamless transitions

Mesh simplification algorithms mostly consist of local coarsening operations that are applied iteratively on meshes. Some operations support seamless transitions. For example, the edge collapse operation (Fig. 8.1a) allows the morphing of the old mesh to the new one, by displacing continuously the two vertices of the collapsed edge to the position of the resulting vertex. This property of the edge collapse operation extends to arbitrary numbers of collapses at the same time: for each input vertex v_i , it is possible to find a destination vertex v'_i given the set of edge collapse operations, and seamless transitions can be achieved by linearly interpolating positions v_i and v'_i . Therefore, it is possible to compute discrete LODs, separated by arbitrary numbers of collapse operations. Note that the vertex clustering operation [RB93, SS95] and the poly-chord collapse operation [DSSC08] can be seen as groups of edge collapse operations applied simultaneously, meaning that they also support seamless transitions. To the contrary, some local operations in mesh simplification methods do not support as easily seamless transitions, as for instance retopology operations (Fig. 8.1c). In this chapter, we essentially rely on edge collapse operations and the face loop collapse operation, which is a special case of poly-chord collapse [DSSC08].

8.1.3 Mapping for reflectance prefiltering

In production, the faces of an asset are textured and assigned a surface material, more precisely a reflectance model (Lambertian model, microfacet models, layered surface models, combinations of various

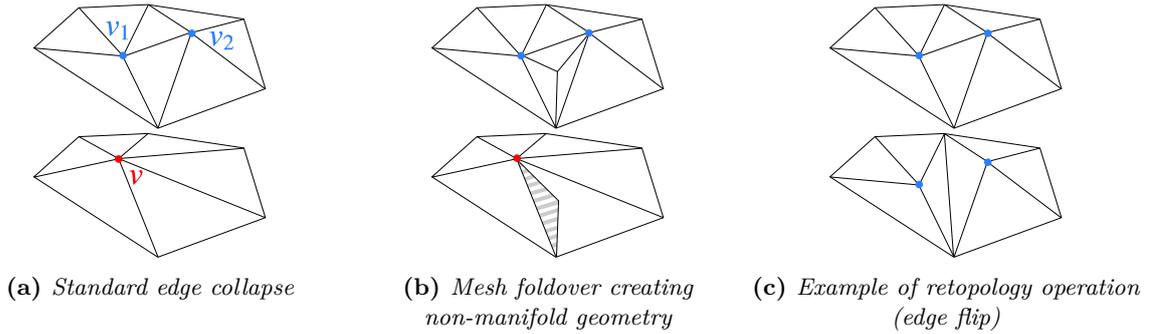


Figure 8.1: Examples of local coarsening and retopology operations on triangle meshes. Top: geometry before applying the operation. Bottom: geometry after the operation.

models, etc.) and reflectance parameters (albedo, microfacet distributions, etc.). As mentioned above, we ensure that simplified faces are not larger than the target spatial resolution, so that we can use per-face reflectance parameters in the LOD. We need to compute prefiltered reflectance parameters for each simplified face based on the materials of the high-resolution faces.

We have seen in chapter 4 that material prefiltering is often done in texture space, by neglecting correlations between reflectance parameters, masking-shadowing effects on rough surfaces and the macroscopic curvature of the surface. With these assumptions, material prefiltering reduces to prefiltering the reflectance models over a region of the surface (Sec. 4.2). In our context, the spatial filter of the prefiltering integral for each face of the simplified geometry is defined over one or various high-resolution faces (Fig. 8.2), which potentially have different texture spaces. Therefore, we need a mapping from simplified faces to high-resolution faces for extending existing texture-space material prefiltering techniques to mesh simplification algorithms.

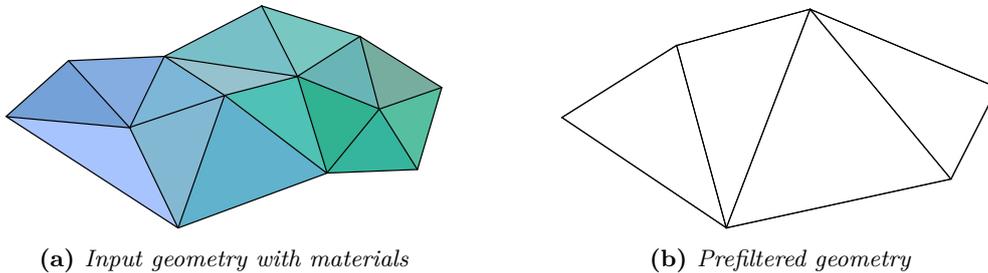


Figure 8.2: In mesh simplification frameworks such as edge collapse, a mapping from the simplified surface (b) to the input high-resolution surface (a) is missing, preventing from efficient material prefiltering. In this chapter, we address this problem for edge collapse simplification (Sec. 8.3) and quad mesh simplification using the face loop collapse operation (Sec. 8.4)

For prefiltering materials along with a geometric simplification algorithm, we need to solve two problems:

- Geometry simplification algorithms do not necessarily provide a clear mapping for material prefiltering during the simplification (Fig. 8.2). For example, in the edge collapse framework, there is no clear mapping from simplified to high-resolution polygons, in particular when merged vertices are set to arbitrary positions for reducing geometric errors.
- If we have such a mapping, we can express materials of the simplified faces as an integral of the materials over the high-resolution surface, which simplifies to a weighted sum of input materials in the case of per-face attributes. We can then address the problem of prefiltering them, that is, representing efficiently the combined materials to reduce the memory usage and the noise in rendered images due to high spatial frequencies.

In this chapter, we propose two solutions for the first problem: we introduce a new method for obtaining linear combination weights for prefiltering materials in the edge collapse framework (Sec. 8.3), and we propose a face loop collapse algorithm for simplifying quad meshes with a simple mapping between input faces and low-resolution faces (Sec. 8.4). We also partially address the second problem and show how to efficiently prefilter microfacet normal distributions in this context (Sec. 8.3).

8.1.4 Preserving manifold surfaces

Preserving manifold surfaces while prefiltering macrosurfaces has several benefits:

- Our macrosurface criterion (Chap. 7) was designed for analyzing manifold surfaces. New cases appear when meshes are not manifold and we only studied the robustness of our criterion for manifold surfaces.
- When simplifying a flat mesh without ensuring that the surface remains manifold, faces adjacent to non-manifold edges overlap as shown in Fig. 8.1b, which can introduce artifacts at rendering.
- Material prefiltering defined as an integral of the materials of the high-resolution surface is only well-posed if the simplified surface has the same topology as the input surface, which is assumed to be manifold in our work. Prefiltering becomes more ambiguous when the topology of the simplified mesh is different. For example, if a T junction is created by a coarsening operation on a manifold surface, it is unclear how to define front and back materials for each simplified face.

In the edge collapse framework, non-manifold geometry can be created from manifold surfaces, as shown in Fig. 8.1b. This is also true for related algorithms such as vertex merging and poly-chord collapsing. In the case of edge collapse on triangle meshes, one can check that a particular collapse operation preserves a manifold surface using the Link condition [Ede06]: when two adjacent vertices v_1 and v_2 are collapsed, it is sufficient to be sure that if there is another point v_3 that is adjacent to both v_1 and v_2 through edges, then there must be a face (v_1, v_2, v_3) . The Link condition can be checked very efficiently if a fast access to adjacent elements is provided by the mesh data structure. Preserving manifold surfaces in the case of face loop collapse operations is more challenging and discussed in the work of [DSSC08].

8.2 A new vertex placement strategy for robust mesh simplification with edge collapses

In mesh simplification algorithms based on edge collapse [LWC*02], one of the key ingredients is the strategy for placing merged vertices after each edge collapse. Several vertex placement strategies have been proposed. The most simple approach, known as *half-edge collapse*, consists in placing the vertex v resulting from a collapse at one of the initial vertices (v_1 and v_2 in Fig 8.3). This leads to inaccurate simplified meshes because objects tend to shrink [LT98, Hop99]. This is an important issue for complex objects such as trees because the global appearance is very sensitive to the occlusion of the small elements, and may be visibly changed if every leaf is inaccurately simplified. An example is given in Fig 8.4. Placing v between v_1 and v_2 does not perform better.

Vertex placement strategies based on quadric error metrics (Sec. 3.1, [GH97, Hop99]) have been successfully used for fast simplification and they improve the quality of simplified meshes compared to naïve approaches. Unfortunately, QEM methods also tend to shrink the geometry as noted by Hoppe [Hop99].

Several authors addressed the problem of preserving the volume of a mesh in the edge collapse framework [Gué99, Hop99, LT98, ALSS99]. These methods reduce the shrinking effect and generally improve the quality of simplified meshes. However, volume preservation is not always the right goal: a mesh could have a null or almost null volume locally, and volume-preserving methods are not robust in this case as shown in Fig. 8.4c. Preserving the area of the surface is not the proper criterion either because the

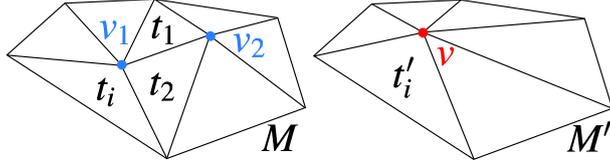


Figure 8.3: One edge collapse operation. Vertices v_1 and v_2 are merged into a new vertex v . Triangles t_1 and t_2 are collapsed. Every other triangle t_i corresponds to a new triangle t'_i .

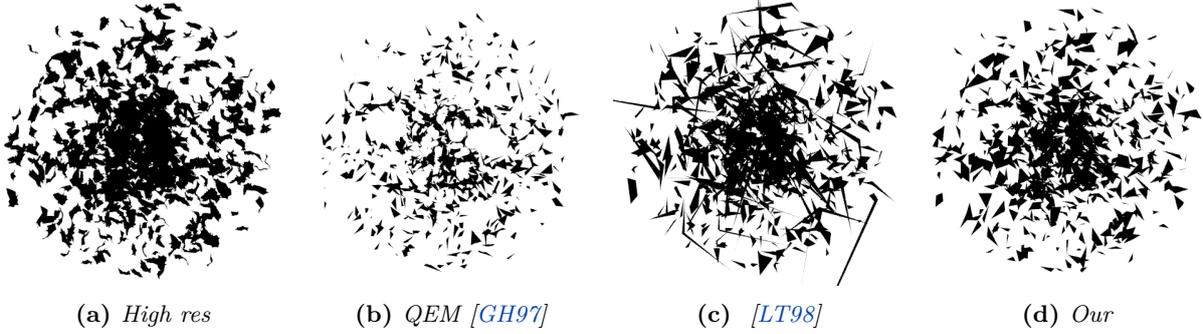


Figure 8.4: Comparison of vertex placement strategies for accurate mesh simplification with edge collapses. (a): The input high resolution mesh. (b): QEM-based simplification tends to shrink the geometry. (c): Volume-preserving simplification proposed in [LT98]. It leads to instabilities for thin geometries such as leaves. (d): Our vertex placement strategy. Unlike volume-preserving methods, it is robust for thin geometry and almost degenerated cases.

simplification of very rough surfaces would generate incorrect results. Actually, what we want for our LODs is to minimize a two-sided geometric distance between the input mesh and the simplified mesh, in order to minimize errors in pixels.

We propose a new vertex placement strategy based on the minimization of a two-sided mean square distance between the high-resolution surface and the simplified surface. Our algorithm produces results with quality similar to volume-preserving methods and it is more robust.

8.2.1 Background

For each collapse operation, we propose to place the merged vertex such that it minimizes a two-sided distance in the spirit of Aspert et al. [ASCE02]. Given two vertices v_1 and v_2 (Fig 8.3), M the set of triangles adjacent to v_1 and v_2 and M' the triangles adjacent to the new vertex v , we want to find a position for v that minimizes

$$d_{sq}(M, M') = \iint_{p \in M} d(p, M')^2 dp + \iint_{p' \in M'} d(p', M)^2 dp' \quad (8.2.1)$$

with

$$d(p, M') = \min_{p' \in M'} \|p - p'\|_2.$$

This formulation or similar ones have been used for measuring geometric errors in simplified meshes [OVBP11], but have never been used directly for vertex placement to our knowledge. Finding a position for v that minimize Eq. 8.2.1 is difficult because of non linearities: contrary to the quadric error metric, we consider the actual distances between points and surfaces, instead of simpler distances between points and the planes that contain high-resolution faces.

8.2.2 Our approach

We propose to approximate $d_{sqr}(M, M')$ (Eq. 8.2.1) with a sum of square distances. Our algorithm is iterative and it tries to find a better position \mathbf{x} for the vertex v at each step.

Let's consider an edge collapse such as the one in Fig. 8.3. Given a current position \mathbf{x} , our algorithm is the following:

1. We choose some points \mathbf{p}_i on M , and find for each one the nearest point \mathbf{p}'_i on M' . We write points \mathbf{p}'_i as functions of \mathbf{x} using barycentric coordinates: $\mathbf{p}'_i = a'_i \mathbf{x} + \mathbf{b}'_i$ with $a'_i \in \mathbb{R}$ and $\mathbf{b}'_i \in \mathbb{R}^3$.
2. We choose some points \mathbf{P}'_i on M' , and find for each one the nearest point \mathbf{P}_i on M . We write points \mathbf{P}'_i as functions of \mathbf{x} using barycentric coordinates: $\mathbf{P}'_i = A'_i \mathbf{x} + \mathbf{B}'_i$ with $A'_i \in \mathbb{R}$ and $\mathbf{B}'_i \in \mathbb{R}^3$.
3. Then, we minimize square distances between the points of each pair:

$$\mathbf{x}_{\text{next}} = \arg \min_{\mathbf{x}} \left(\sum \|\mathbf{p}'_i - \mathbf{p}_i\|^2 + \sum \|\mathbf{P}'_i - \mathbf{P}_i\|^2 \right). \quad (8.2.2)$$

4. We iterate until convergence or stop after a chosen number of steps.

The main idea behind this algorithm is that minimizing Eq. 8.2.2 also reduces distances between \mathbf{p}_i and M' and between points \mathbf{P}'_i and M . Computing \mathbf{x}_{next} can be done efficiently:

$$\begin{aligned} \mathbf{x}_{\text{next}} &= \arg \min_{\mathbf{x}} \sum \|a'_i \mathbf{x} + \mathbf{b}'_i - \mathbf{p}_i\|^2 + \sum \|A'_i \mathbf{x} + \mathbf{B}'_i - \mathbf{P}_i\|^2 \\ &= \arg \min_{\mathbf{x}} \sum a_i'^2 (\mathbf{x} \cdot \mathbf{x}) + 2a'_i (\mathbf{b}'_i - \mathbf{p}_i) \cdot \mathbf{x} + \sum A_i'^2 (\mathbf{x} \cdot \mathbf{x}) + 2A'_i (\mathbf{B}'_i - \mathbf{P}_i) \cdot \mathbf{x} \\ &= \arg \min_{\mathbf{x}} c(\mathbf{x} \cdot \mathbf{x}) + \mathbf{C} \cdot \mathbf{x} \\ &= -\frac{\mathbf{C}}{2c} \end{aligned}$$

with $c = \sum a_i'^2 + \sum A_i'^2$ and $\mathbf{C} = \sum 2a'_i (\mathbf{b}'_i - \mathbf{p}_i) + \sum 2A'_i (\mathbf{B}'_i - \mathbf{P}_i)$.

8.2.3 Implementation and discussion

We implemented this algorithm and we compute optimal positions of the merged vertex for each collapse operation. For points \mathbf{p}_i , we choose the positions of v_1 and v_2 (the vertices of the collapsed edge) as well as points at the middle of the edges that are adjacent to v_1 and v_2 . For points \mathbf{P}'_i , we use the position of the vertex v (its current position) and points at the middle of the edges that are adjacent to v . We stop the optimization after 200 iterations and use the tolerance error $\epsilon = \text{targetRes}/100$.

We observed that our algorithm converges most of the time with few iterations. During our tests, we measured that the average number of iterations was 7, and the average cost of the optimization was 164 μs per edge collapse. The most time-consuming task in our algorithm is finding the closest points \mathbf{p}'_i and \mathbf{P}_i . This is why we choose few points \mathbf{p}_i and \mathbf{P}'_i . We compared our method with the standard QEM strategy and one volume-preserving method (Fig. 8.4) implemented in CGAL [CGA16]. Our method leads to results that are similar to the volume-preserving method in simple cases, and it is more robust for thin geometry because our minimization is well conditioned.

8.3 A new material prefiltering method for the edge collapse framework

In this section, we propose a method for prefiltering materials when meshes are simplified with edge collapse operations. We introduce a strategy for writing the reflectance models of simplified faces as

linear combinations of the reflectance models of high-resolution faces. We show how to combine our method with the work of Heitz et al. [HDCD15] and Dong et al. [DWMG15] for prefiltering linearly microfacet normal distributions during the mesh simplification, without requiring a common parametric space for all the high-resolution faces.

8.3.1 Introduction

Let’s consider a single edge collapse operation on a manifold surface, as shown in Fig 8.3. We assume that each face of the input geometry is assigned a reflectance model with per-face parameters. Our goal is to preserve the appearance of the surface after the simplification, in which the merged vertex is positioned such that the geometric error is minimized (Sec. 8.2). As explained before, our strategy is to write the reflectance models of the simplified faces as linear combinations of the reflectance models of the high-resolution faces.

When the geometry is perfectly planar locally, each triangle of the geometry is seen proportionally to its size from a given point of view. In this case, it is possible to preserve exactly the appearance of the surface from a distance by assigning to each simplified face the average material of the input geometry, that is, a sum of the input materials weighted by the areas of the corresponding triangles. This naive strategy over-blurs the appearance of the surface as illustrated in Fig. 8.5b. We propose another method that preserves exactly the appearance in the planar case while avoiding over-blurring (Fig. 8.5c).

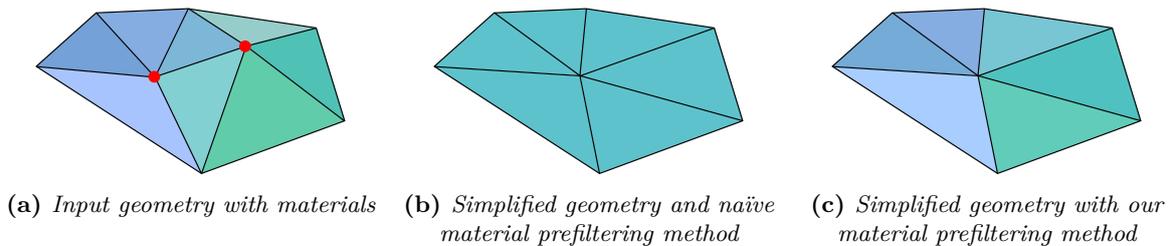


Figure 8.5: *In the case of a planar geometry simplified using edge collapse, a simple prefiltering solution consists in prefiltering the reflectance models of simplified faces by averaging the reflectance models of all the neighboring high-resolution faces (b). This solution preserves exactly the appearance of the surface from a distance, but it over-blurs the surface appearance. We propose a method that also preserves exactly the appearance in the planar case while avoiding over-blurring (c).*

The case of a non-planar geometry is more difficult. Indeed, the contribution of each face to the appearance is view dependent, because of different cosine factors for the projected area of each face depending of their normal, and because of masking between faces (Sec. 4.3). It is not possible in general to preserve exactly the appearance from all view directions. In our work, we derive weights from the planar case, and we assume that our strategy is good for almost-planar geometry, and good enough in average (over all directions) in more complex cases.

8.3.2 Notations

Let’s consider the collapse of the edge joining two vertices v_1 and v_2 as illustrated in Fig. 8.3. We call the merged vertex v . We assume for clarity that neither v_1 nor v_2 are border vertices in this explanation. Our results extend easily to the general case. Let t_i be the N triangles adjacent to v_1 and v_2 ($i \in [1, N]$) such that t_1 and t_2 are the collapsed faces. Let t'_i ($i \in [3, N]$) be the faces after the collapse such that each t_i becomes the face t'_i after the collapse. We call M the set of triangles t_i and M' the set of triangles t'_i (Fig. 8.3). Let A_i and A'_i be the areas of triangles t_i and t'_i . We have $A'_1 = 0$ and $A'_2 = 0$ since t_1 and t_2 are collapsed. We call w_i the area of each triangle t_i divided by the area of M :

$$\forall i \in [1, N], \quad w_i = \frac{A_i}{\sum_k A_k}. \quad (8.3.1)$$

Similarly, we define

$$\forall i \in [1, N], \quad w'_i = \frac{A'_i}{\sum_k A'_k}. \quad (8.3.2)$$

By definition, we have $\sum w_i = 1$ and $\sum w'_i = 1$, $w'_1 = w'_2 = 0$.

8.3.3 Preserving the mean radiance

In the case of a planar geometry with per-face BRDFs f_i , and assuming a locally uniform lighting (Sec. 4.2), the mean radiance of the surface M in direction ω_o is given by:

$$L_{mean}(\omega_o) = \sum_{i \in [1, N]} w_i \int_{\Omega} L(\omega) f_i(\omega, \omega_o) \langle \omega, n \rangle d\omega, \quad (8.3.3)$$

with n the normal of the surface and $L(\omega)$ the incident radiance in direction ω . We want to find BRDFs f'_i for each triangle t'_i so that the mean radiance is preserved after the collapse, meaning $L_{mean}(\omega_o) = L'_{mean}(\omega_o)$ with

$$L'_{mean}(\omega_o) = \sum_{i \in [3, N]} w'_i \int_{\Omega} L(\omega) f'_i(\omega, \omega_o) \langle \omega, n \rangle d\omega. \quad (8.3.4)$$

8.3.4 Our approach

We first write each BRDF f'_i as a linear combination of BRDFs f_i :

$$\forall j \in [3, N], \quad f'_j = \sum_{i \in [1, N]} w'_i{}^j f_i \quad \text{with} \quad \forall j, \quad \sum_i w'_i{}^j = 1 \quad (8.3.5)$$

and we need to find appropriate weights $w'_i{}^j$. The mean radiance of M' in the planar case is given by

$$\begin{aligned} L'_{mean}(\omega_o) &= \sum_j w'_j \int_{\Omega} L(\omega) \sum_i w'_i{}^j f_i(\omega, \omega_o) \langle \omega, n \rangle d\omega \\ &= \sum_i \sum_j w'_j w'_i{}^j \int_{\Omega} L(\omega) f_i(\omega, \omega_o) \langle \omega, n \rangle d\omega. \end{aligned} \quad (8.3.6)$$

Then, our problem is to find weights $w'_i{}^j$ so that $L'_{mean}(\omega_o) = L_{mean}(\omega_o)$. Thus, we want to ensure that

$$\forall i \in [1, N], \quad \sum_j w'_j w'_i{}^j = w_i. \quad (8.3.7)$$

As discussed in the introduction of this section, there is a simple and naïve solution, which consists in choosing $w'_i{}^j = w_i \forall j$ (Fig. 8.5b). This solution results in unnecessarily blurred surfaces: all the parameters are averaged in the neighborhood of the collapsed edge, even if the collapsed edge is actually very small compared to neighboring triangles. Instead, we propose a simple heuristic for the choice of weights $w'_i{}^j$: we maximize weights $w'_i{}^i$ (*i.e.* the weight of f_i in the simplified triangle t'_i) in order to avoid over-blurring, and so that the mean radiance $L(\omega_o)$ is preserved in every direction ω_o .

Let $G = \{i \in [1..N], w_i < w'_i\}$ be the set of triangles whose relative area increase with the edge

collapse, and $L = \{i \in [1..N], w_i \geq w'_i\}$ the other ones. Our weights are the following:

$$\begin{cases} w_i^i = 1 & \forall i \in L \\ w_j^i = 0 & \forall i \in L \text{ and } i \neq j \\ w_i^i = \frac{w_i}{w'_i} & \forall i \in G \\ w_j^i = \left(1 - \frac{w_i}{w'_i}\right) c_j & \forall i \in G \text{ and } i \neq j \\ \text{with } c_j = \frac{w_j - w'_j}{\sum_{i \in L} (w_i - w'_i)} \end{cases} \quad (8.3.8)$$

We can check that our weights satisfy Eq. 8.3.7. If $i \in G$,

$$\sum_j w_i^j w'_j = \frac{w_i}{w'_i} w'_i = w_i.$$

Else if $i \in L$,

$$\begin{aligned} \sum_j w_i^j w'_j &= w'_i + \sum_{k \in G} w_i^k w'_k \\ &= w'_i + \sum_{k \in G} c_i \left(1 - \frac{w_k}{w'_k}\right) w'_k \\ &= w'_i + \sum_{k \in G} \frac{w_i - w'_i}{\sum_{l \in L} (w_l - w'_l)} (w'_k - w_k) \\ &= w'_i + (w_i - w'_i) \frac{\sum_{k \in G} (w'_k - w_k)}{\sum_{l \in L} (w_l - w'_l)}. \end{aligned}$$

Observing that

$$\begin{aligned} \sum_{k \in G} (w'_k - w_k) - \sum_{l \in L} (w_l - w'_l) &= \sum_{k \in G} (w'_k - w_k) + \sum_{l \in L} (w'_l - w_l) \\ &= \sum_i w'_i - \sum_j w_j \\ &= 0 \end{aligned}$$

we can write

$$\begin{aligned} \sum_j w_i^j w'_j &= w'_i + (w_i - w'_i) \frac{\sum_{k \in G} (w'_k - w_k)}{\sum_{k \in G} (w'_k - w_k)} \\ &= w_i. \end{aligned}$$

Therefore, our method satisfies equation 8.3.7.

8.3.5 Application: using our weights for prefiltering microfacet distributions

Once all the weights w_i^j have been computed, we can address the problem of prefiltering the materials for each face. For example, if each input face was assigned a Lambertian BRDF, our weights can be used for prefiltering diffuse albedos. A more interesting problem is prefiltering microfacet normal distributions, as microfacet models are widely used for rendering rough dielectric and conductor materials, and because prefiltering microfacet distributions is closely related to prefiltering normals of displaced surfaces (Sec. 4.2.2, Sec. 4.4.2). In previous work on linear prefiltering of normal distributions [OB10, DHI*13], normals were prefiltered in texture space assuming that the distributions were defined in the

same frame. In our case, there is no such common frame for arbitrary groups of high-resolution faces. Finding a convenient frame would be possible for nearly planar geometry but rather ill-posed in general.

Walter et al., in the supplemental material of the work of Dong et al. [DWMG15], proposed a microfacet distribution based on the normal distribution of an ellipsoid that is not necessarily aligned with the geometric normal. This distribution was introduced for rendering anisotropic materials with off-centered microfacet distributions. Independently of this work, Heitz et al. [HDCD15] proposed to use a distribution based on an ellipsoid for representing microflake orientations in the microflake volume model introduced by Jakob et al. [JAM*10a]. Heitz et al. have shown that such distributions can be prefiltered linearly with sufficient quality in many cases.

We propose to combine these techniques for our LODs: we rely on the microfacet model of Walter et al. for both our input and LOD faces, and we prefilter microfacet distributions using our weights and the linear method of Heitz et al. [HDCD15]. For specular microfacets, the BRDF of Walter et al. [DWMG15] writes:

$$f_s(\omega_i, \omega_o, \mathbf{S}) = \frac{D(\mathbf{h}, \mathbf{S}) G_2(\omega_i, \omega_o, \mathbf{h}, \mathbf{S}) F(\omega_i \cdot \mathbf{h})}{4|\omega_i \cdot \mathbf{n}||\omega_o \cdot \mathbf{n}|} \quad (8.3.9)$$

with \mathbf{S} the SGGX matrix, D the microfacet SGGX normal distribution, G_2 the masking-shadowing term, F the Fresnel term, \mathbf{h} the half-vector and \mathbf{n} the normal. Analytic expressions and sampling procedures can be found in the supplemental material [DWMG15]. For diffuse microfacets, we follow the formulation and sampling procedure proposed by Heitz et al. [HD15]. It is given by

$$f_d(\omega_i, \omega_o, \mathbf{S}) = \frac{1}{\pi} \frac{1}{|\omega_o \cdot \mathbf{n}|} \int_{\Omega} \langle \omega_o, \omega \rangle \frac{G_2(\omega_i, \omega_o, \mathbf{h}, \mathbf{S})}{G_1(\omega_i, \mathbf{h}, \mathbf{S})} D_{\omega_i}(\omega, \mathbf{S}) d\omega \quad (8.3.10)$$

with D_{ω_i} the visible normal distribution in direction ω_i and G_1 the masking term. $\langle \cdot, \cdot \rangle$ is the clamped dot product. Our per-triangle BRDF consists of specular and diffuse terms:

$$f_i(\omega_i, \omega_o) = s_i f_s(\omega_i, \omega_o, \mathbf{S}_i) + d_i f_d(\omega_i, \omega_o, \mathbf{S}_i) \quad (8.3.11)$$

with s_i the triangle specular albedo and d_i the diffuse albedo. \mathbf{S}_i is the SGGX matrix.

Using the notations introduced above and our weights, our prefiltering strategy writes:

$$\begin{aligned} \forall j \in [3, N], \quad f'_j &= \sum_{i \in [1, N]} w_i^j f_i \\ &= \sum_i w_i^j \left(s_i f_s(\omega_i, \omega_o, \mathbf{S}_i) + d_i f_d(\omega_i, \omega_o, \mathbf{S}_i) \right) \\ &\simeq s'_j f_s(\omega_i, \omega_o, \mathbf{S}'_j) + d'_j f_d(\omega_i, \omega_o, \mathbf{S}'_j) \end{aligned} \quad (8.3.12)$$

with $s'_j = \sum_i w_i^j s_i$ the mean specular albedo, $d'_j = \sum_i w_i^j d_i$ the mean diffuse albedo and $\mathbf{S}'_j = \sum_i w_i^j \mathbf{S}_i$ the filtered normal distribution. This approximation is valid if the SGGX linear prefiltering is accurate, which is not always the case as highlighted by Zhao, Wu et al. [ZWDR16] who used a multi-lobe representation when needed. Another underlying assumption in our prefiltering method is that albedos and SGGX distributions are uncorrelated.

Note that when the ellipsoid is aligned with the geometric normal, the microfacet distribution of Walter et al. reduces to a simple GGX distribution. This means that GGX microfacet distributions [WMLT07] can easily be converted to the ellipsoid microfacet distributions. Our method could be extended to specular and diffuse transmission terms [WMLT07, DWMG15].

8.3.6 Results and discussion

Our prefiltering method is fast because weights are given by very simple formulas that only depend on the area of each triangle, and because we only rely on linear prefiltering of albedos and SGGX matrices. In our implementation, material prefiltering at each collapse takes around 7 μ s. This is small compared to the cost of the vertex placement (around 170 μ s in our implementation). Our method can prefilter anisotropic roughness due to small surface details as shown in Fig. 8.6.

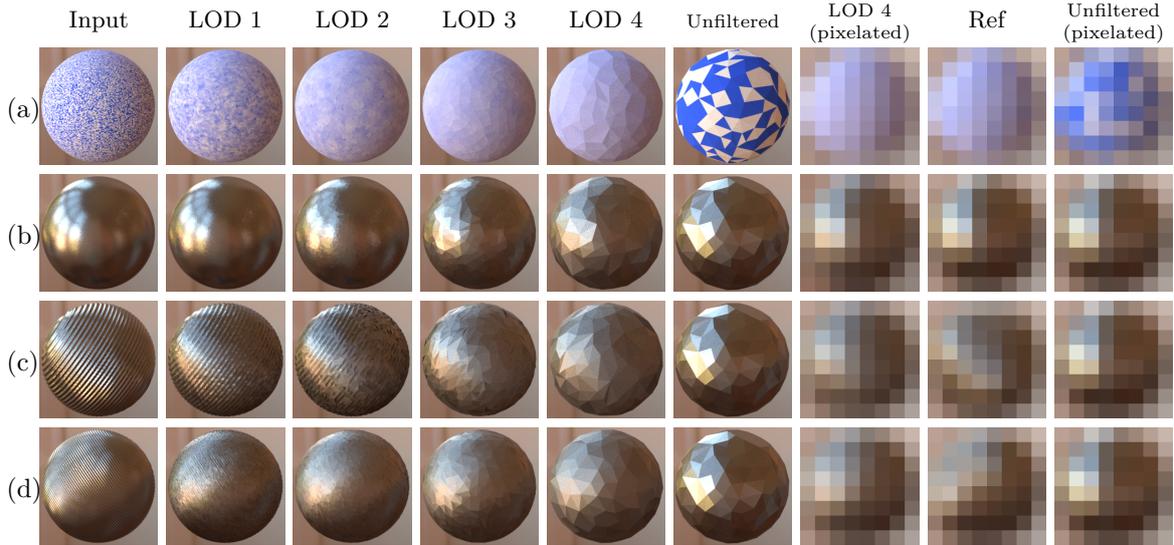


Figure 8.6: Meshes simplified with edge collapse operations and prefiltered with our method. (a): Sphere with diffuse microfacets. (b): Sphere with specular microfacets. Coarse LODs have rougher materials since our prefiltering is done in world space and takes into account surface curvature. (c,d): Spheres with specular materials and grooves leading to anisotropic normal distributions in coarse LODs.

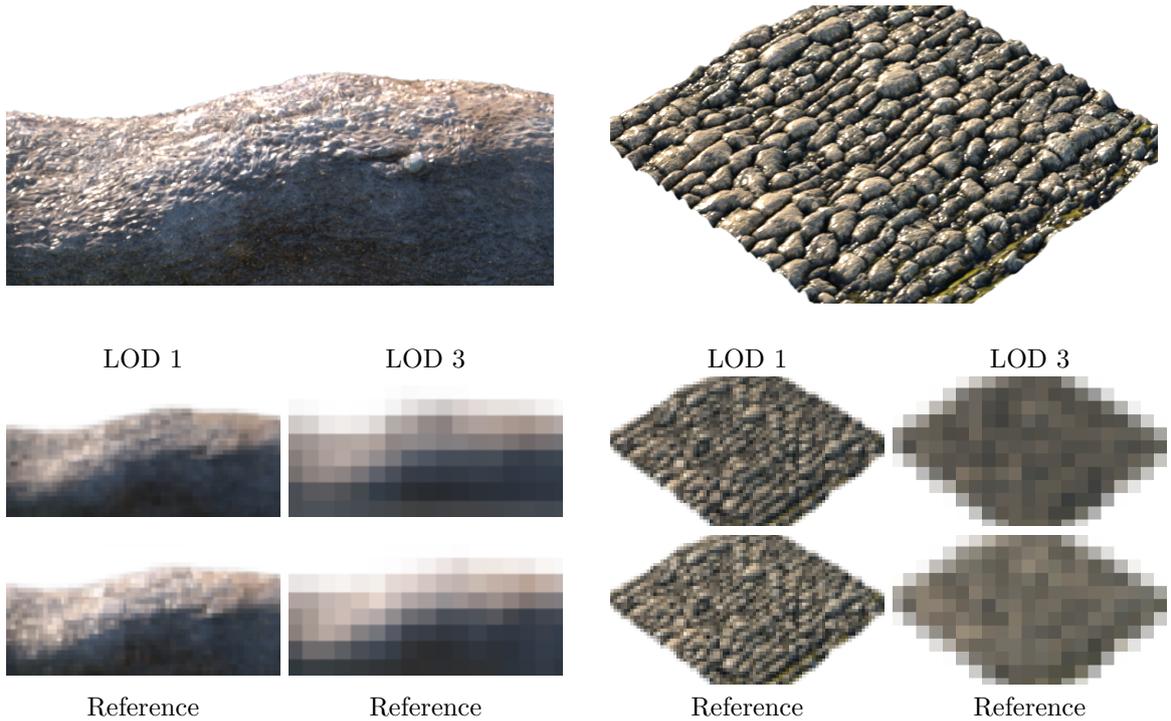


Figure 8.7: Macrosurface prefiltering of two assets with diffuse and specular microfacets. Left: a tree trunk. Right: Stone pavers. Rendered in Mistuba (*sunsky* emitter).

We derived our weights from the case of a planar geometry but we use them for all kinds of edge collapses operations. As mentioned above, our method is less accurate on non-planar cases, because the emitted radiance in direction ω_o depends on cosine weights $\langle \omega_o, \mathbf{n}_i \rangle$ for each triangle with normal \mathbf{n}_i , and also on view-dependent visibility terms taking into account masking between triangles.

Our method is also limited by the use of a single SGGX lobe: this representation cannot be used for

prefiltering surfaces with strong microscopic correlations between the geometry and albedos [HSRG07, HN12] or normal distributions that are very different from SGGX lobes. Multi-lobe SGGX distributions could be used for improving the quality of the simplified materials following ideas of Zhao, Wu et al. [ZWDR16] or Xu et al. [XWZB17].

8.4 Prefiltering macrosurfaces by collapsing face loops

In previous sections, we proposed to prefilter macrosurfaces relying on edge collapse operations. We introduced a new strategy for placing merged vertices and a method for prefiltering materials at each collapse operation. In this section, we propose an alternative algorithm for prefiltering macrosurfaces assuming that input assets consist of well-structured quadrilateral meshes with feature-aligned edge loops. We have seen that production quad meshes sometimes have tiny polygons close to large polygons (Sec. 6.1), especially around sharp features for controlling subdivision surfaces (Fig. 6.3b). Our solution based on quad mesh simplification brings several benefits in this case:

- We rely on methods that simplify the geometry while keeping structured quad meshes, which are more adapted to some production pipelines, in particular in pipelines with per-face textures [BL08].
- Our method provides a simpler mapping between the simplified faces to the high-resolution quads, compared to mesh simplification based on edge collapse operations.

Our contribution is not the mesh simplification algorithm itself but its adaptation to the problem of prefiltering macrosurfaces with control of the geometric error and material prefiltering. We designed and implemented the method proposed in this section at the Walt Disney Animation Studios and found that it is well compatible with production assets with per-face texture patches and heterogeneous quad sizes.

8.4.1 Background

Quad mesh simplification has been studied by various authors and it is now as mature as triangle mesh simplification. We reviewed previous work and existing coarsening operations in section 3.1.3. Previous methods mostly addressed the problem of generating low-resolution quad meshes with good geometric properties, for example evenly shaped quads and vertices with a valence of 4.

Some coarsening operations do not easily support seamless transitions. This is because at rendering, each quad is replaced by a pair of triangles. There are two possible triangulations for each quad and they are not visually equivalent (except for perfectly planar quads). Therefore, methods that change the topology of neighboring polygons (such as qeMerge [DSC09] and edge collapse [TPC*10]) could introduce popping artifacts unless quad triangulation is carefully controlled.

Several local coarsening operations (ring collapse, diagonal collapse) could be adapted for macrosurface simplification by using weights in the same way as what we proposed for edge collapse in section 8.3. However, we rely on special poly-chord collapse operations that lead to a better mapping from simplified quads to high-resolution quads. As discussed by Tarini et al. [TPC*10], the poly-chord collapse operation lacks flexibility for some applications, because it is a non-local operation that sometimes improves the mesh in some areas and deteriorates other areas. They propose to rely only on local coarsening operations for a better control of the simplification. However, we found that poly-chord collapse is flexible enough for production meshes which generally have clean, feature-aligned edge loops with equally sized faces. We adapt the poly-chord collapse algorithm proposed by Daniels et al. [DSSC08] for our requirements: resolution-driven simplification with control of the geometric error, seamless transitions and clear mappings for well-posed material prefiltering.

8.4.2 Face loops

Before giving the details of our quad mesh prefiltering algorithm, we recall some properties of structured quad meshes. We use the term *face loop* for designating a set of quad faces connected by their opposite edges. Face loops are similar to rings [tBPHJ02] and its dual representation, the poly-chord [DSSC08]. We use the term face loop because this is the name used in modeling software, and because we think it is clearer to talk about faces rather than their dual representation, at least for our application. Examples of face loops are given in Fig. 8.8. Face loops have the following properties:

- On 2-manifold meshes without boundaries, face loops are always closed.
- On 2-manifold with boundaries, they are either closed or with their two endings on boundaries.
- Face loops can have one or more self-intersections (Fig. 8.8e).
- Each face belongs to exactly two face loops, except in the case of a self-intersection, in which case the face only belongs to a single face loop.
- A single face loop can span the entire geometry. This rarely happen on structured quad meshes.
- Face loops can be adjacent to themselves, as shown in Fig. 8.8d and 8.8e.

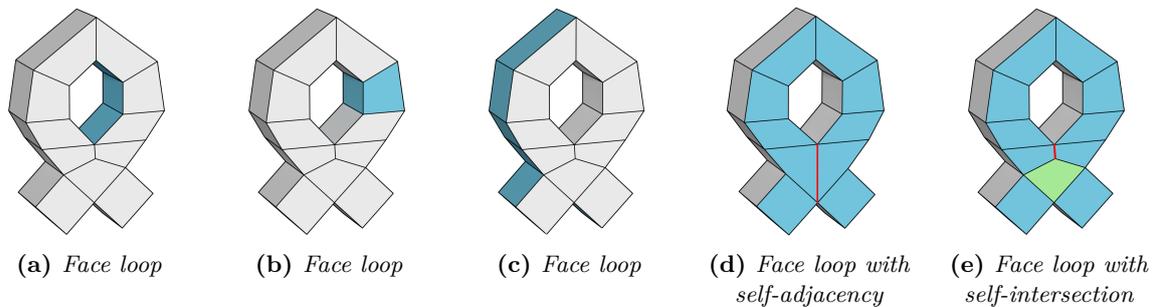


Figure 8.8: Examples of face loops on closed 2-manifold quad meshes. Face loops are shown in blue, with self-intersection faces in green and self-adjacency edges in red.

8.4.3 Collapsing face loops

The face loop collapse operation consists in collapsing the quads of a face loop by merging one side of the loop with the other side. As for the half-edge collapse equivalent in the edge collapse framework, the face loop collapse operation can be applied in two directions as shown in Fig. 8.9. The remaining vertices have the same position as the vertices of one of the sides of the face loop. This adds validity constraints for candidate face loops: a valid face loop cannot be adjacent to itself more than two times orthogonally to the loop direction, as shown in Fig. 8.9. Else, the collapse is not well defined. This operation is a special case of the poly-chord collapse operation [DSSC08], and it is well compatible with material prefiltering as discussed later.

8.4.4 Mappings with the face loops collapse operations

The face loop collapse operation provides a simple mapping between high-resolution faces and simplified faces, as shown in Fig. 8.10. Simplified faces map to two high-resolution faces, except around face loop self-intersections. Material prefiltering can be done using this mapping, with weights proportional to face areas, in the same spirit as our weights in the case of edge collapse operations. Because each low-resolution face actually corresponds to an array of quad faces, the mapping could even be used for rewriting textures for the low-resolution faces from the high-resolution textures.

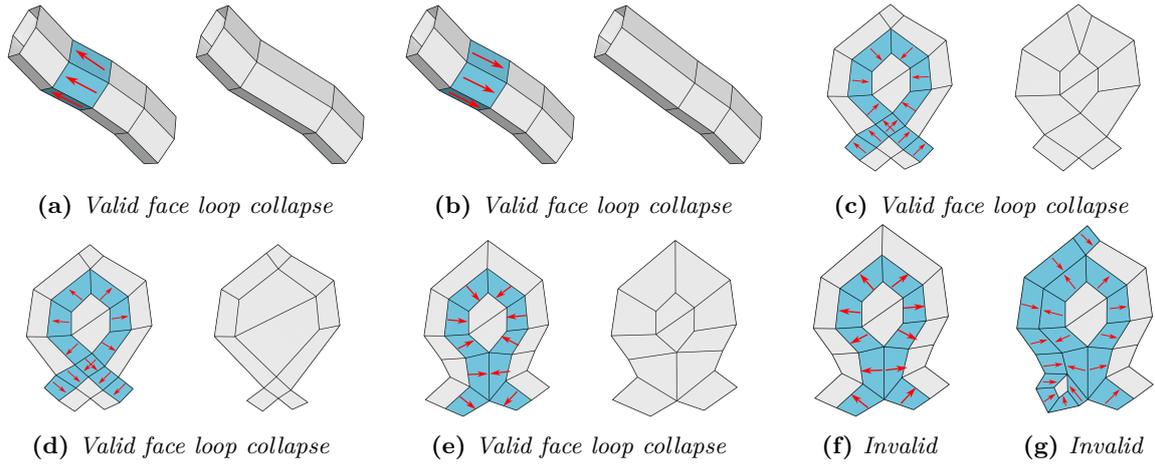


Figure 8.9: Examples of face loop collapse operations. In the case of self-adjacency, some collapse operations are not valid (**f,g**) because there is no well-defined collapse direction for all the self-adjacency edges.

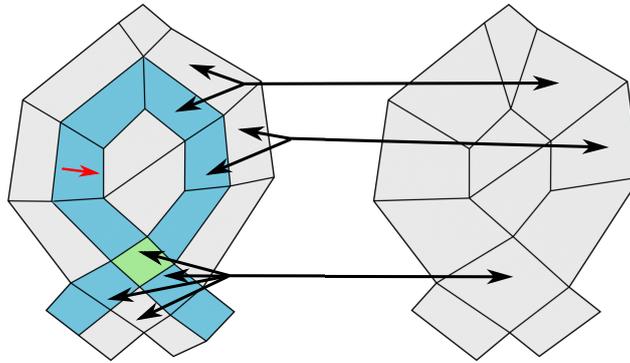


Figure 8.10: With the face loop collapse operation, most low-resolution faces map to two of the high-resolution faces, except around self-intersection faces. This mapping is important for material prefiltering. Note that some high-resolution faces are affected by the collapse operation even if they do not map to other faces, such as the face at the very top of the geometry.

8.4.5 Resolution-based stopping criteria

As for mesh simplification based on edge collapse, we need to control the quad mesh simplification based on the desired prefiltering scale, in order to avoid visible errors in pixels due to overly aggressive coarsening operations. In our implementation, we used the following constraints:

- A face loop collapse operation is only valid if the faces it affects are smaller than the target resolution, before and after the coarsening operation. This ensures that faces larger than the prefiltering scale are not modified, and that small faces do not become larger than the prefiltering scale.
- We control the geometric error using a two-sided distance, as mentioned in Sec. 8.3. We estimate this error by choosing a set of points on the low- and high-resolution surfaces, and looking at the distances from these points to the other surface. Then, we look at the maximum of these distances, and we only allow the collapse if it stays below the target resolution times a tolerance factor $\epsilon \in (0, 1)$, ϵ being a parameter of our simplification algorithm.

8.4.6 Topology verifications

We have seen that the Link condition must be evaluated in edge collapse algorithms for ensuring that the surface remains a 2-manifold (Sec. 8.3). As discussed in the work of Daniels et al. [DSSC08], manifold preservation is more challenging for poly-chord collapse operations because they are cases where all the individual edge collapse of a face loop collapse are topologically valid, but the whole face loop collapse operation is not. For example, in the case of a cube, all edge collapse are valid individually but all the face loop collapse operations result in degenerated faces and are therefore invalid. In their article, Daniels et al. [DSSC08] propose an equivalent of the Link criterion for face loops where each edge collapse is verified individually while virtually collapsing all the other edges.

8.4.7 Algorithm

The first step of our simplification algorithm is to find all the face loops of a given quad mesh, to select only those that are valid given our resolution-based criteria (no faces larger than the target resolution, etc.) and to collapse the face loop that introduces the lowest geometric error. Finally, some information about the collapse operation can be stored so that the entire mapping between high- and low-resolution faces can be retrieved at the end of the simplification, for both prefiltering and smooth geometric transitions. Then, other collapse operations have to be found and applied iteratively, until all valid operations have been applied.

A correct but inefficient simplification algorithm would be to recompute the topological validity and the geometric errors of all the remaining face loops after each collapse, since the last collapse operation have potentially affected all of them. We optimized the algorithm by maintaining a list of collapse operations, sorted by geometric errors. After each collapse operation, we analyze again the first collapse operation on the list, and we apply it if it is still valid if it has a geometric error below the chosen tolerance factor (otherwise, we update the list and try again with the next collapse operation). This strategy does not ensure that the operation with the lowest geometric error is applied, but it guarantees that the geometric error remains under the tolerance factor and it avoids time waste for recomputing all the errors after each collapse. There is room for improvement in this algorithm. For example, we could explore smarter ways to keep some information about the potential geometric error for each face of the mesh, for the 4 collapse operations for this faces, and update only what is needed after each collapse operation.

8.4.8 Discussion

This algorithm is less flexible and more time consuming than simplification based on edge collapse operations, but it brings several benefits due to the preservation of structured quads in prefiltered macrosurfaces. In the case of assets modeled with subdivision surfaces with a mix of large and tiny base faces, it may be useful to support the subdivision of large faces at rendering, while prefiltering areas with sub-resolution faces. We know that the subdivision of a base face is only affected by the neighboring faces: it is possible to add some constraints during the prefiltering to prevent the simplification of faces that influence the subdivision of large faces. We did not handle this case in our implementation. To support simultaneously adaptive subdivision and prefiltering in the same mesh, other implementation details should to be studied, for instance ensuring crack-free surfaces and artifact-free seamless transitions.

8.5 Summary of the chapter

In this chapter, we addressed the problem of prefiltering macrosurfaces with their materials. We revisited some previous work on geometric simplification and proposed new tools for generating accurate prefiltered macrosurfaces. Our main contributions in this chapter are the following:

- We analyzed the requirements of macrosurface prefiltering for our LOD application, and in particu-

lar seamless transitions, well-posed material prefiltering by ensuring that surfaces remain manifold, and the existence of a mapping between low- and high-resolution surfaces.

- We discussed existing vertex placement strategies for algorithms based on edge collapse and we proposed a new strategy that addresses the lack of robustness of volume-preserving methods.
- We proposed a method for prefiltering materials in the edge collapse framework. We introduced prefiltering weights that preserve the appearance of the high-resolution surface in flat regions and avoid over-blurring.
- We showed how to combine our prefiltering weights, the SGGX linear prefiltering method [HDCD15] and the ellipsoid microfacet normal distribution [DWMG15] to efficiently prefilter microfacet normal distribution in the edge collapse framework.
- We introduce an alternative algorithm for prefiltering macrosurfaces in the case of quad meshes. We showed that the face loop collapse operation preserves structured quad meshes, defines a simple mapping between low- and high-resolution surfaces, and is more adapted to assets with both large textures faces and sub-resolution faces.

8.6 Résumé du chapitre

Dans ce chapitre, nous nous sommes penchés sur le problème du préfiltrage de macrosurfaces avec leurs matériaux. Nous avons modifié des approches existantes de simplification géométrique et nous avons proposé de nouveaux outils pour générer des surfaces préfiltrées précises. Les principales contributions présentées dans ce chapitre sont les suivantes :

- Nous avons analysé les besoins de notre approche de préfiltrage pour les macrosurfaces et souligné l'importance des transitions invisibles entre les différentes résolutions, des méthodes de filtrage bien posées qui s'assurent de la bonne topologie des surfaces, et de l'existence d'une bijection entre les surfaces de haute et basse résolution.
- Nous avons commenté les techniques existantes de placement de sommets pour les algorithmes de simplification de maillage par edge collapse, et nous avons proposé une nouvelle méthode qui est plus fiable.
- Nous avons proposé une méthode de préfiltrage des matériaux dans le cadre de la simplification de maillage par edge collapse. Nous avons introduit des poids de préfiltrage qui préservent l'apparence de la surface dans les régions planes et qui évitent de flouter inutilement l'apparence de la surface de basse résolution.
- Nous avons montré comment utiliser notre méthode avec le filtrage linéaire de distributions SGGX [HDCD15] et la distribution de normales d'ellipsoïde [DWMG15] pour préfiltrer efficacement les distributions de normales sans paramétrisation globale de la surface.
- Nous avons introduit un algorithme alternatif pour le préfiltrage des macrosurfaces faites de quadrilatères. Nous avons montré que l'opération de simplification des boucles de faces peut préserver la structure régulière des maillages de quadrilatères, tout en garantissant une bijection claire entre les surfaces de haute et basse résolution. Cet algorithme est aussi mieux adapté aux objets 3D ayant à la fois de grandes et de petites faces texturées.

Chapter 9

Prefiltering sub-resolution intricate geometry

We have seen in chapter 6 that an important requirement for prefiltering sub-resolution intricate geometry is the preservation of the local probability of occlusion. This is why existing surface-based methods such as mesh simplification algorithms cannot be used for prefiltering complex geometry at arbitrary scales. We discussed in chapter 5 some previous work on approximating aggregate elements using volume models, and we have seen that these methods target specific aggregations such as hair or granular materials, whereas our goal is to prefilter complex geometry without strong assumptions about the input data.

We propose a voxelization algorithm for prefiltering intricate geometry with estimations of the local occlusion in the input geometry. Based on this, we show how to estimate scattering parameters for the microflake volume model, and we discuss an alternative representation based on a few sub-resolution polygons.

9.1 Background: voxelization algorithms

9.1.1 Voxelization

The word *voxelization* can be rather confusing due to its multiple uses in computer graphics and other fields. First, let's recall that a voxel is a volume element used for storing any kind of data depending on the application. Voxelization designates an algorithm that takes as input some data and converts it to a voxel-based representation, usually regular voxel grids, sparse voxels grids or multi-resolution volume hierarchies such as sparse voxel octrees.

Example of such algorithms include:

- Voxelization using binary voxels, for instance empty and non-empty voxels.
- Voxelization using planar representations per voxel or a distance field for prefiltering meshes, that is storing positions, normals and reflectance parameters [YLM06, LK10a, HN12, PES15].
- Voxelization with occlusion estimation [Pan14].
- The work of Lacewell et al. [LBBS08] is closely related: they prefilter the occlusion of complex meshes, not exactly in voxels but for each node of a bounding volume hierarchy.

In the next section, we propose a new voxelization algorithm that measures local occlusion probabilities for each voxel by casting rays on the input geometry. Our approach is related to previous work on

occlusion prefiltering [LBBS08]. Contrary to them, we address aliasing problems that appear with current voxelization algorithms. Then, we estimate parameters for a participating medium model to preserve both the occlusion and the scattering properties.

9.1.2 Aliasing in voxelization algorithms

In previous approaches, the occlusion of the geometry was measured in cubes or in cuboids. Pan-telev [Pan14] addressed the problem of fast GPU voxelization and estimated the occlusion using local rasterizations of the triangles inside the voxel. Lacewell et al. [LBBS08] proposed to render the geometry inside BVH nodes in the three main axis to get directional opacities.

In these methods, each voxel is obtained only from the geometry contained in the corresponding cube or box, that is, the voxelization uses a sort of 3D box filter that is prone to artifacts (Fig. 9.1), as already noted in [WK93]. Other examples of aliasing artifacts can be found in the work of Schroder et al. [SKZ11], in which they prefilter high-frequency periodic fiber patterns.

The second problem of these methods is that they only measure opacity in the directions of the three main axes. This may lead to other kinds of artifacts: for example, an axis aligned polygon of unit area has a projected area of 1 in one direction and 0 in the other directions, whereas the same polygon rotated such that its normal becomes $\frac{1}{\sqrt{3}}(1, 1, 1)$ have a projected area of $\frac{1}{\sqrt{3}} \simeq 0.58$ in the three axis. This means that estimating the average occlusion of a geometry by averaging 3 projected areas gives significantly different results depending on the orientation of the geometry with respect to the grid. Ideally, estimating occlusion and scattering parameters should not involve specific directions.

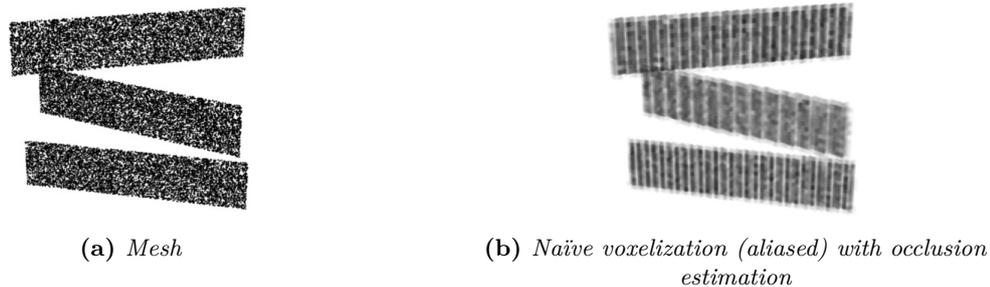


Figure 9.1: Example of an aliased voxel grid (b) obtained with a simple voxelization algorithm. In this case, the occlusion estimation in each cubic voxel leads to aliasing due to sub-resolution well-organized elements (a) that are not aligned with the voxel grid.

Although these artifacts are relatively uncommon or not visible in most cases, we will show that it is actually simple to avoid them by choosing appropriate ray casting strategies:

- Casted rays should have random directions in order to take into account the appearance from all the points of view.
- Voxels values such as opacity should not be computed only from the geometry inside the corresponding cube. Artifacts can be avoided if the voxelization process satisfies the Nyquist sampling criterion: some kind of low-frequency and large-enough prefiltering kernel should be used. In other words, each voxel should represent a neighborhood around of the voxel center, and this neighborhood should be large enough.

9.2 A new aliasing-free voxelization algorithm with ray casting

In this section, we address the problem of retrieving a local occlusion probability from some input geometry, for each voxel of a 3D grid. In the next section, we will show how to use this information to preserve

a correct occlusion probability in LODs. We propose to estimate occlusion probabilities by casting rays on the input mesh and looking for local intersections with the geometry. One of our main contributions is an algorithm for ray casting that has good prefiltering properties and avoids aliasing problems compared to naïve methods.

9.2.1 Occlusion estimation with ray casting

Our approach consists in casting rays around each voxel center and looking for intersections with the geometry in the voxel neighborhood. The main ideas of our voxelization algorithm is that all rays have the same length and are equally taken into account, and that we use smooth 3D probability density functions (*pdf*) for sampling ray origins, which acts like a low-pass filter and prevents object-space aliasing. We sample a random direction uniformly on the spherical domain for each ray in order to prevent directional artifacts.

We cast all rays with a fixed length L_r , meaning that we ignore intersections farther than L_r from ray origins. This allows to derive a mean probability of occlusion for a distance L_r around the voxel center. We only take into account the first intersection with the geometry, if any.

The ray length L_r is a parameter of our algorithm. Long rays due to a large value L_r would tend to over-blur the voxelization, because the geometry that is far from the voxel center would be taken into account. To the contrary, small rays would only measure how much the geometry fills the space, and the measured occlusion would tend to zero when L_r tends to zero in the case of infinitely thin polygons. In our implementation, we set L_r to the voxel size, which avoids over-blurring and takes into account correlations between occluders at the prefiltering scale.

An estimation of the probability of occlusion in the voxel neighborhood over a distance L_r is given by the proportion of rays that intersect the geometry:

$$P_{occ} = \frac{\text{nbHits}}{\text{nbRays}} \quad (9.2.1)$$

where nbRays is the number of rays casted for this voxel and nbHits is the number of rays that intersected some geometry at a distance smaller that L_r . In Section 9.3.1, we show how to use P_{occ} to retrieve a density parameter for the voxel.

9.2.2 Artifact-free voxelization

We now propose a sampling algorithm for casting rays around voxels in order to avoid object-space aliasing. Let v_i be the center of a voxel i , and D_i the probability density function used to randomly sample ray origins for that voxel (Fig. 9.2). We want to know how much (*i.e.* with which probability) each point $x \in \mathbb{R}^3$ can be hit by a ray during the estimation of the occlusion for a given voxel. This probability must vary smoothly in space in order to prevent object-space aliasing.

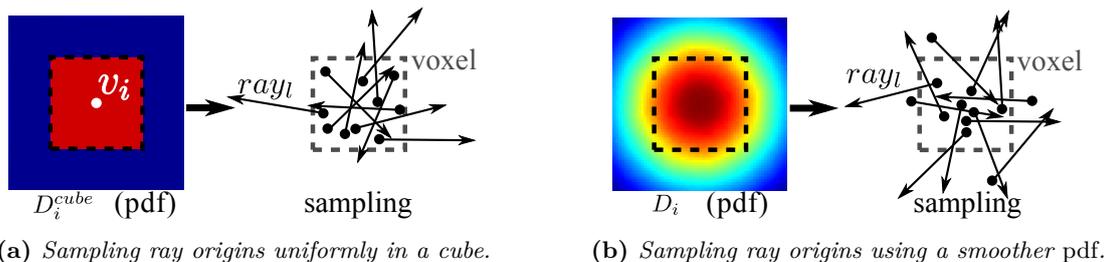


Figure 9.2: Sampling strategies for ray origins using different probability density functions. We sample a random direction for each ray in order to avoid artifacts.

We introduce the function P_i that gives, for each point $x \in \mathbb{R}^3$, the probability that a ray whose origin is sampled using the *pdf* D_i intersects a spherical occluder with radius γ centered in x . This function P_i should be smooth enough: it is the low-pass filter of the voxelization. Moreover, given that there are N_v voxels, we want the function P_{all} defined by

$$P_{all}(x) = \frac{1}{N_v} \sum_{\text{voxels } i} P_i(x) \quad (9.2.2)$$

to be constant for all points x inside the voxel grid, *i.e.* all points in space have the same probability of being intersected by rays during the entire voxelization (this is also for preventing aliasing). In general, an arbitrarily chosen distribution D_i would not satisfy Eq. 9.2.2, meaning that some parts of the input asset will be more sampled than others. In the rest of this section, we propose some distributions D_i that ensure a uniform sampling of the voxelized asset.

Probability of intersecting an occluder. Although we do not strictly need it for the rest of our derivation, we first discuss how rays casted with random directions for a given origin *sample* the neighboring geometry. This gives some intuitions about our voxelization algorithm. These rays can intersect the geometry in a neighborhood with some probability that depends on the distance to the origin. Let P_r be the radial function that gives the probability that a ray hit a spherical occluder of radius γ , given that the occluder is at a distance d from the ray origin. Intuitively, $P_r(d)$ decreases as long as d increases because the solid angle of the spherical occluder decreases. The solid angle of a cone with apex angle θ is given by $2\pi(1 - \cos(\theta))$. The solid angle of a spherical occluder is the one of a cone with apex

$$\theta = \arccos\left(\sqrt{1 - \frac{\gamma^2}{d^2}}\right)$$

where d is the distance between the ray origin and the occluder center. For $d \geq \gamma$ and $d \leq L_r$, we have

$$P_r(d) = \frac{1}{2} \left(1 - \sqrt{1 - \frac{\gamma^2}{d^2}}\right)$$

which tends to $\frac{\gamma^2}{4d^2}$ when γ becomes very small compared to d . We can see that the probability of being sampled for polygons around the origin is roughly inversely proportional to the square distance to the origin.

Sampling strategies. For a given D_i (the *pdf* used for sampling ray origins) and P_r (the probability of intersection depending on the distance from a ray origin), we can write the probability P_i of being intersected at x during the voxelization of voxel i as

$$P_i(x) = \int_{\mathbb{R}^3} P_r(\text{dist}(x, p)) D_i(p) dp.$$

The probability of being intersected at x during the entire voxelization is the weighted sum of $P_i(x)$ over all the voxels:

$$\begin{aligned} P_{all}(x) &= \frac{1}{N_v} \sum_i \int_{\mathbb{R}^3} P_r(\text{dist}(x, p)) D_i(p) dp \\ &= \frac{1}{N_v} \int_{\mathbb{R}^3} P_r(\text{dist}(x, p)) \left(\sum_i D_i(p) \right) dp \\ &= \frac{1}{N_v} \int_{\mathbb{R}^3} P_r(\text{dist}(0, p)) \left(\sum_i D_i(p + x) \right) dp. \end{aligned}$$

Now P_{all} is constant in space if $\sum_i D_i$ is constant. A simple solution is to choose a cubic uniform probability density function:

$$D_i^{cube}(x) = \begin{cases} \frac{1}{\text{voxelSize}^3} & \text{if } \|x - v_i\|_\infty \leq \text{voxelSize} \\ 0 & \text{otherwise} \end{cases}$$

Algorithm 1: *Our sampling algorithm for ray origins (Section 9.2.2). We sample origins in a cube around the voxel and then use a 3D normal distribution in order to avoid object-space aliasing.*

```

nbHits = 0;
rayL = voxelSize;
mu = (0,0,0);
sigma = 0.6×voxelSize ;
for i = 1 to nbRays do
    rayD = SampleDir ();
    rayO = voxelCenter + (SampleUnitCube () - (½,½,½)) × voxelSize + SampleNormalDistrib (mu, sigma);
    if Intersect (rayO, rayD, rayL) then
        | nbHits ++;
    end
end
end

```

where $\| \cdot \|_\infty$ is the infinity norm. Moreover, any convolution of a solution with another probability density functions is also a solution. Smooth functions P_i can be achieved using $D_i = D_i^{cube} * D^{smooth}$, with D^{smooth} any smooth probability density function. In our implementation, we used a 3D Gaussian function with standard deviation $\sigma = 0.6 \times \text{voxelSize}$ (example (c) in Fig. 9.3). A pseudo-code for our sampling method can be found in Algorithm 1.

9.3 Prefiltering sub-resolution intricate geometry using the microflake model

Once the local occlusion probability has been measured as explained in the previous section, we can address the problem of prefiltering sub-resolution intricate geometry while preserving a correct local occlusion in the LOD. We show how to estimate density and scattering parameters for the microflake model, which is the model we used for prefiltering complex assets. Results are shown in the next chapter.

9.3.1 Estimating density parameters for the microflake model

At a given voxel, Eq. 9.2.1 estimates the local probability of occlusion of rays with length L_r in the voxel neighborhood. When using a volume model for prefiltering the sub-resolution geometry, the first step is to estimate an attenuation coefficient for each voxel (or density parameters in the microflake model) such that the local probability of occlusion is preserved. For isotropic participating media, the attenuation coefficient σ_t of a voxel must satisfy

$$1 - e^{-\sigma_t L_r} = P_{occ}$$

and we obtain

$$\sigma_t = -\frac{1}{L_r} \log \left(1 - \frac{\text{NbHits}}{\text{NbRays}} \right)$$

In the microflake model, the attenuation coefficient in direction ω is given by

$$\sigma_t(\omega) = \rho \sigma(\omega)$$

and

$$\sigma(\omega) = \int_{\Omega} \langle \omega \cdot m \rangle D(m) dm,$$

with D the microflake normal distribution function, ρ the density parameter and $\langle \cdot \rangle$ the clamped dot product. We discuss how to obtain microflake distributions in the next paragraph. Given a microflake distribution for the voxel, we look for a density parameter ρ such that the average attenuation in all directions equals P_{occ} :

$$1 - \frac{1}{4\pi} \int_{\Omega} e^{-\rho \sigma(\omega) L_r} d\omega = P_{occ}$$

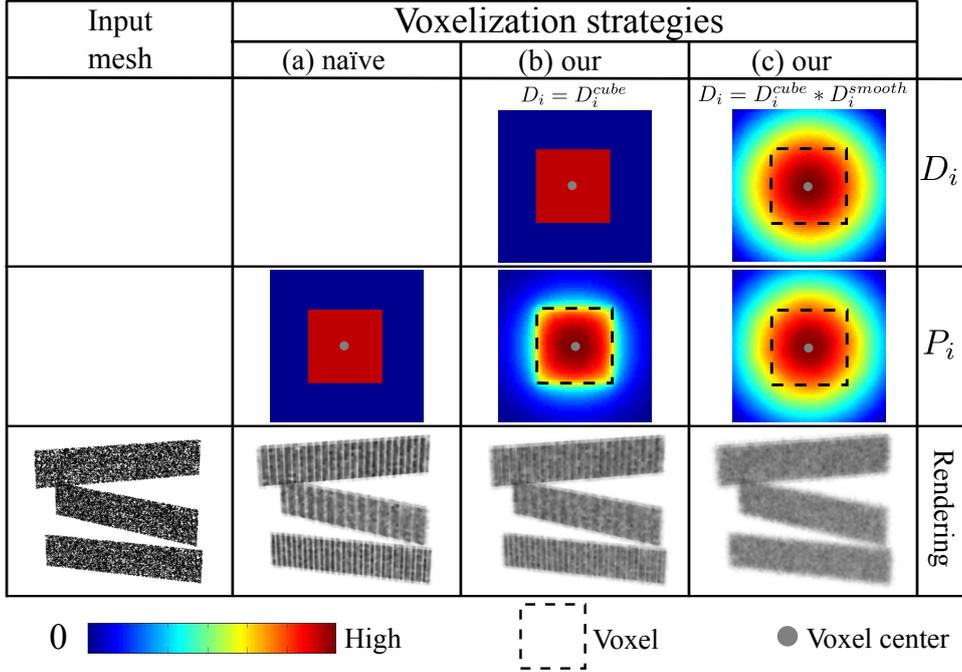


Figure 9.3: Effect of voxelization strategies on object-space aliasing. The naïve voxelization (a) consists in measuring the occlusion of geometry in a small cube around the voxel center. Our voxelization algorithm can prevent object-space aliasing and acts as a low-pass filter (b,c). Images represent slices of the 3D probability density functions D_i for sampling ray origins (Section 9.2.2) and slices of 3D functions P_i giving for each point the probability of being intersected by a ray.

In general, there is no closed-form expression for the parameter ρ . One can show that the function f defined as

$$f(\rho) = \frac{1}{4\pi} \int_{\Omega} e^{-\rho\sigma(\omega)L_r} d\omega$$

is a strictly decreasing function and has C^∞ continuity, which makes numerical methods based on 1D gradient descent very efficient for estimating ρ . It requires evaluating f and $\frac{\partial f}{\partial \rho}$, which we estimate using numerical integration of the spherical integral in our implementation.

9.3.2 Estimating other scattering parameters: albedos, microflake distributions

For prefiltering sub-resolution geometry using the microflake model, we need also to estimate microflake albedos and microflake distribution (the microflake distribution must be found before optimizing the density parameter). We rely on the SGGX microflake distribution introduced by Heitz et al. [HDCD15], because it is flexible and memory efficient.

We propose to obtain a distribution of microflake directly from the distribution of normals of the polygons of the sub-resolution geometry. For the occlusion estimation, we have proposed an algorithm based on ray casting. We leverage this and store the SGGX matrix describing the normal distribution at each ray intersection [HDCD15]. At each ray intersection, we also get the specular and diffuse albedos of the surface reflectance model. We average the SGGX matrices and the albedos and we use specular and diffuse SGGX phase functions [HDCD15]. Examples of voxelization and prefiltering using microflake volumes are shown in the next chapter.

9.4 Discussion

Prefiltering sub-resolution intricate geometry using the microflake model as proposed above has several limitations:

- We only estimate the average occlusion, and do not try to estimate view dependent occlusion, for example due to foliage-like geometry. Our strategy (Sec. 9.3.1) is to preserve this average occlusion in the microflake media, which has some anisotropy but not necessarily the same anisotropy as in the input geometry. However, because we retrieve microflake normal distributions from the microfacet distributions of the voxelized faces, the anisotropic occlusion in the microflake medium is often similar to the one of the geometry. In the next part of the thesis, we show that preserving a correct anisotropic occlusion in the LOD is not critical compared to the phase functions and albedos (Chap. 14), and we believe that preserving a correct average occlusion is sufficient for almost all the assets.
- We only estimate and store one SGGX distribution per voxel, which has a limited accuracy. For example, this cannot preserve the sparkling appearance of foliage due to specular reflections on large leaves, as discussed in the next chapter.
- We only store two albedos per voxel (specular and diffuse), and did not try to encode view-dependent volume albedos.
- We linearly prefilter albedos. We did not try to take into account the local probability of self-shadowing in the geometry and to derive more accurate albedos from this information. We address a related problem in part III.

During our collaboration with the Walt Disney Animation Studios, we experimented an alternative prefiltering method in which we generate a few polygons per voxel such that they preserve a correct average occlusion, instead of relying on a volume representation. Polygons are more convenient than volumes in production pipelines, for example for animation or when many assets are overlapping. Our test on occlusion using polygons were encouraging, although the problem of estimating correct reflectance parameters for such polygons remains to be addressed.

9.5 Summary of the chapter

In this chapter, we discussed previous voxelization algorithms with occlusion estimation, and we proposed a new algorithm that avoids aliasing artifacts. The main idea of our algorithm is to cast rays with arbitrary directions around voxel centers and to estimate occlusion probabilities from the number of ray intersections.

Based on our algorithm for estimating local occlusion, we show how to prefilter sub-resolution intricate geometry using the microflake volume model, and we propose strategies for estimating density, microflake distributions and albedos for each voxel.

9.6 Résumé du chapitre

Dans ce chapitre, nous avons présenté plusieurs algorithmes de voxélisation qui estiment l’occlusion locale, et nous avons introduit un nouvel algorithme qui évite les problèmes d’aliasing. L’idée principale de notre algorithme est de lancer des rayons dans des directions aléatoires dans le voisinage d’un voxel pour estimer les probabilités d’occlusion à partir du nombre d’intersections trouvées.

Avec notre algorithme d’estimation de l’occlusion locale, nous montrons comment préfiltrer de la géométrie complexe sous-résolution en utilisant le modèle de microflakes, et nous proposons des stratégies pour estimer la densité, les distributions de normales et les albedos pour chaque voxel.

Chapter 10

Results of our hybrid LOD approach

In this chapter, we show and discuss the results of the hybrid LOD approach (Fig. 6.7) proposed in chapter 6. We first give some details about how we combined our macrosurface analysis (Chap. 7) with macrosurface prefiltering (Chap. 8) and the voxelization of sub-resolution intricate geometry (Chap. 9).

10.1 Implementation of our hybrid LOD approach

10.1.1 Choice of scattering models

Our hybrid LOD pipeline is rather independent of the surface reflectance models used in the input asset and the LODs, as it is independent of the participating media model used for prefiltering sub-resolution geometry. However, some scattering models easily support material prefiltering, and we have shown in chapter 8 that the ellipsoid NDF combined with SGGX linear prefiltering allows for the efficient prefiltering of microfacet distribution in our pipeline. We have also explained that the microflake volume model is well adapted to geometry prefiltering because phase functions can be derived from the reflectance and orientations of sub-resolution polygons.

For generating the results shown in the next sections, we used the following models:

- The reflectance model of input surfaces consists of a microfacet model with the ellipsoid NDF distribution. Microfacets have both a diffuse and a specular microscopic BRDF. For each input face, we store a SGGX distribution (6 coefficients), a specular albedo and a diffuse albedo (RGB).
- We use the same reflectance model for prefiltered macrosurfaces.
- For the volume model, we use the microflake model with one SGGX distribution and we store diffuse and specular albedos (RGB).

We didn't experimented other kinds of materials with, for instance, transmission lobes or light emission. Prefiltering arbitrary materials such as layered materials, sparkling materials, measured materials or fabrics models remains an open problem and this was not the goal of our work.

10.1.2 Seamless transitions between LODs

Seamless transitions were one of the main requirement for our hybrid method, and we took this constraint into account when choosing mesh simplification algorithms. Then, perfectly seamless transitions can be achieved quite easily.

Seamless transitions for macrosurface geometry. Given a macrosurfaces of LOD_N , the set of edge collapse operations that leads to LOD_{N+1} defines a mapping between the vertices of LOD_N and the vertices of LOD_{N+1} [Hop96]. We interpolate vertex positions in order to achieve smooth transitions between the two LODs.

Seamless transitions for surface materials. In the edge collapse framework (Sec. 8.3), each triangle in LOD_{N+1} maps to a triangle in the previous LOD LOD_N . We linearly interpolate their albedos and SGGX coefficients during the transition between LOD_N and LOD_{N+1} . As we use SGGX coefficients in world space, specular reflections stay consistent during the transition even if the geometric normal of the triangles is changing. The same technique can be used for quad-based mesh prefiltering (Sec. 8.4).

Seamless transitions from meshes to volume. Triangles of LOD_N that are voxelized in LOD_{N+1} (Fig. 6.7) must smoothly fade out during the transition, while voxel densities smoothly increase. We simply use an opacity coefficient for each voxelized triangle that goes from 1 to 0 during the transition.

Seamless transitions between volumes. 8 voxels in LOD_N correspond to 1 voxel in LOD_{N+1} . In our implementation, we linearly interpolate densities, albedos and SGGX coefficients of voxels in LOD_N with the corresponding voxels in LOD_{N+1} . We do not use spatial interpolation between voxels of the same LOD because our voxelization algorithm prevents high frequencies in the volumetric data and because it facilitates seamless transitions (otherwise, a quadrilinear interpolation must be used).

10.1.3 Other implementation details

Voxelization. We used Jakob’s Mistuba renderer [Jak10] for casting rays during our voxelization (Chap. 9). We casted 200 rays for each non-empty voxel and stored 16 bytes: 4 for the density parameter (one float), 6 for the SGGX coefficients (1 byte per coefficient), 3 for the specular albedo and 3 for the diffuse albedo (1 byte per color channel).

Rendering. We also used Mitsuba for rendering our LODs. We implemented plugins for the diffuse and specular SGGX phase functions based on the code provided by Heitz et al. [HDCD15]. We modified the core of the renderer in order to render microflake media with different specular and diffuse albedos. We also implemented the BRDF model of Dong et al. [DWMG15] for specular and diffuse microfacets based on the work of Heitz et al. [HD15].

Efficiency. The precomputation times for prefiltering can be found in Fig. 10.4. Our macrosurface prefiltering was run on a single core, while voxelization was computed using 8 hyperthreaded cores on a Intel Xeon E5-2609. For the first LOD of the weeping willow model the total computation time was 20 minutes, which includes 2 minutes for loading the input mesh and its textures, 5 minutes for the macrosurface analysis and 4 minutes for the mesh simplification and prefiltering. The remaining time corresponds to various other tasks such as preparing and writing the outputs of the algorithm.

10.2 Results

Results of our hybrid LOD method are shown in Fig. 10.1, 10.2 and 10.3. These examples include trees and a complex sailing ship model with tiny ropes and wood elements. Our method produces high quality LODs with little errors when compared to reference images. The local and global occlusion of the assets is accurately preserved, meaning that our LODs cast correct shadows in a scene and have a correct semi-transparent appearance from a distance. Seamless transitions from the input mesh to the low-resolution volume have been demonstrated in a supplemental video of our publication [LN17]. This videos shows that the transition from meshes to volumes is barely perceptible, in particular for the foliage

of the weeping willow and the tiny ropes of the ship. To our knowledge, these examples are the first to show seamless transitions from input complex meshes to coarse LODs with fully controlled resolution and correct appearance at all scales.

10.3 Limitations

Our method has different kinds of limitations. Due to its modularity and the several steps involved, the limitations come from different parts of the LOD pipeline:

- Our macrosurface analysis is limited to connected watertight manifold surfaces, as discussed in Chap. 7. In Fig. 10.5, we show an example of our LODs on a planar geometry with holes: the geometry has the appearance of a surface (single normal, perfectly aligned occluders), although it is not watertight. Volumes cannot represent very precisely this surface-like appearance. Our method computes density parameters for the voxels so that the average local occlusion is preserved, which produces an overly transparent appearance in this case. Finally, our macrosurface analysis can lack robustness for very complex surfaces such as the grass-like surface of the asset shown in Fig. 10.5c.
- At the Walt Disney Animation Studios, we found that mesh simplification algorithms suffer from the intersection problem: two distinct macrosurfaces can intersect each other when they are simplified, so that a hidden interior surface can appear on top of an exterior surface in the LOD. Although we have shown how to control two-sided geometric errors during the simplification, our method does not prevent this problem, and intersection verifications remain to be studied.
- Our material prefiltering method relies on single lobe SGGX prefiltering, which cannot represent complex microfacet normal distribution and sparkling appearance. We assume that albedos and microfacet normal distributions were not correlated, which could be incorrect for some types of surfaces. We only prefilter albedos and normal distributions, and do not take into account the self-shadowing and masking of very rough surfaces nor the view-dependent appearance of surfaces with correlated albedos and heights or slopes [HNPN14].
- For our volume representation, we used a single specular albedo and a single diffuse albedo, meaning that our LODs cannot represent volumes with view-dependent colors. As for our BRDFs, we used a single SGGX lobe per voxel. The sparkling appearance of the foliage of the weeping willow is not preserved in coarse LODs, since a single SGGX lobe cannot represent this type of appearance.

10.4 Discussion

We presented results generated by our hybrid pipeline. Actually, several results of this part of the thesis can be used in alternative LOD methods. For example, we used our quad-based method for prefiltering production assets at the Walt Disney Animation Studios, without voxelizing sub-resolution details. The control of the geometric errors allows for simplifying assets *as much as possible* while satisfying some quality constraints for each target resolution. This strategy is not appropriate for trees but allowed high compression rates and high-quality LOD at intermediate scales, while supporting well-posed material prefiltering and seamless transitions.

On the other hand, our voxelization method can be used for prefiltering foliage-like geometry with the a priori knowledge that leaves are sub-resolution, independently of our macrosurface analysis.

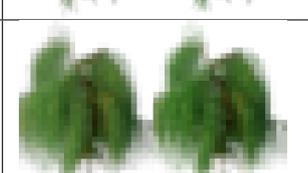
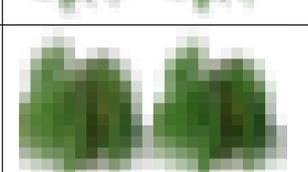
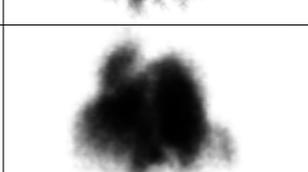
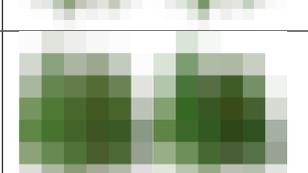
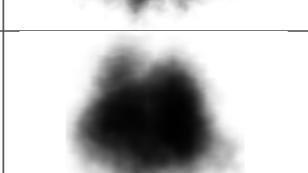
| Resolution | LOD / Ref | Occlusion | Occ. error | Memory usage |
|------------------|---|---|------------|--|
| Input |  |  | 0% | Mesh: 692 MB Mat: 60.4 MB Vol: none Sparsity: none Full grid: none |
| 512 ³ |  |  | 0.90% | Mesh: 17.7 MB Mat: 2.7 MB Vol: 84 MB Sparsity: 97% Full grid: 2 GB |
| 256 ³ |  |  | 1.7% | Mesh: 1.9 MB Mat: 405 kB Vol: 18 MB Sparsity: 94% Full grid: 256 MB |
| 128 ³ |  |  | 2.1% | Mesh: 379 kB Mat: 91 kB Vol: 4 MB Sparsity: 90% Full grid: 32 MB |
| 64 ³ |  |  | 2.8% | Mesh: 60.2 kB Mat: 16.4 kB Vol: 827 kB Sparsity: 83% Full grid: 4 MB |
| 32 ³ |  |  | 4.4% | Mesh: 0 Mat: 0 Vol: 139 kB Sparsity: 76% Full grid: 512 kB |
| 16 ³ |  |  | 8.3% | Mesh: 0 Mat: 0 Vol: 33 kB Sparsity: 57% Full grid: 64 kB |

Figure 10.1: Our LODs rendered with a volumetric path tracer and the *sunsky* emitter in Mitsuba. All assets have diffuse and specular microfacets. The size of volumetric data takes into account the sparsity. We added bytes for voxel indices in the grid (6 bytes for the LOD 512³ and 3 for the other resolutions, per voxel). Occlusion errors were measured in image space.

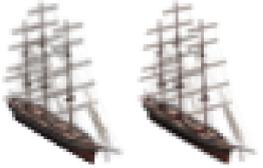
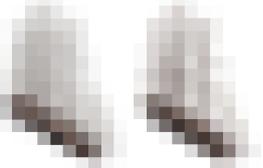
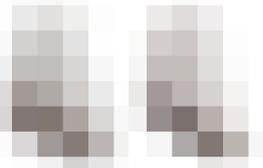
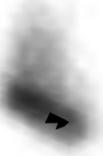
| Resolution | LOD / Ref | Occlusion | Occ. error | Memory usage |
|------------|---|---|------------|--|
| Input |  |  | 0% | Mesh: 43 MB Mat: 2.2 MB Vol: none Sparsity: none Full grid: none |
| 512^3 |  |  | 1.3% | Mesh: 2.7 MB Mat: 456 kB Vol: 6 MB Sparsity: 99.8% Full grid: 2 GB |
| 256^3 |  |  | 0.94% | Mesh: 654 kB Mat: 134 kB Vol: 1.5 MB Sparsity: 99.5% Full grid: 256 MB |
| 128^3 |  |  | 2.8% | Mesh: 176 kB Mat: 44.2 kB Vol: 389 kB Sparsity: 99.0% Full grid: 32 MB |
| 64^3 |  |  | 0.63% | Mesh: 52.4 kB Mat: 15.9 kB Vol: 131 kB Sparsity: 97.3% Full grid: 4 MB |
| 32^3 |  |  | 1.3% | Mesh: 6.7 kB Mat: 3.7 kB Vol: 30 kB Sparsity: 95.0% Full grid: 512 kB |
| 16^3 |  |  | 7.3% | Mesh: 1.3 kB Mat: 2.0 kB Vol: 8 kB Sparsity: 89.0% Full grid: 64 kB |

Figure 10.2: Our LODs rendered with a volumetric path tracer and the *sunsky* emitter in Mitsuba. All assets have diffuse and specular microfacets. The size of volumetric data takes into account the sparsity. We added bytes for voxel indices in the grid (6 bytes for the LOD 512^3 and 3 for the other resolutions, per voxel). Occlusion errors were measured in image space.

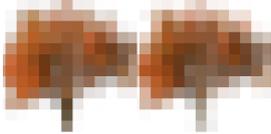
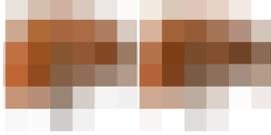
| Resolution | LOD / Ref | Occlusion | Occ. error | Memory usage |
|------------------|---|---|------------|--|
| Input |  |  | 0% | Mesh: 78 MB Mat: 9.2 MB Vol: none Sparsity: none Full grid: none |
| 512 ³ |  |  | -2.4% | Mesh: 35 MB Mat: 5 MB Vol: 45 MB Sparsity: 98.4% Full grid: 2 GB |
| 256 ³ |  |  | -1.35% | Mesh: 1.4 MB Mat: 332 kB Vol: 13 MB Sparsity: 95.7% Full grid: 256 MB |
| 128 ³ |  |  | 0.75% | Mesh: 235 kB Mat: 61 kB Vol: 23.4 MB Sparsity: 92.3% Full grid: 32 MB |
| 64 ³ |  |  | 2.86% | Mesh: 24 kB Mat: 8.6 kB Vol: 623 kB Sparsity: 87.2% Full grid: 4 MB |
| 32 ³ |  |  | 5.11% | Mesh: 2.8 kB Mat: 2.8 kB Vol: 246 kB Sparsity: 59.6% Full grid: 512 kB |
| 16 ³ |  |  | 7.97% | Mesh: 0 Mat: 0 Vol: 26 kB Sparsity: 65.2% Full grid: 64 kB |

Figure 10.3: Our LODs rendered with a volumetric path tracer and the *sunsky* emitter in Mitsuba. All assets have diffuse and specular microfacets. The size of volumetric data takes into account the sparsity. We added bytes for voxel indices in the grid (6 bytes for the LOD 512³ and 3 for the other resolutions, per voxel). Occlusion errors were measured in image space.

| | Willow | Vessel | Maple |
|--|--------|--------|-------|
| Macrosurface prefiltering (mesh & materials) | 40 mn | 3 mn | 3 mn |
| Voxelization, 512^3 | 16 mn | 12 mn | 13 mn |
| Voxelization, 256^3 | 4 mn | 2 mn | 2 mn |

Figure 10.4: Prefiltering times for the 3 assets shown in Fig. 10.1, 10.2 and 10.3. Computation times for prefiltering macrosurfaces include the loading time for input meshes and textures, our macrosurface analysis, the mesh simplification step with our prefiltering algorithm and other tasks like writing the new meshes and textures.

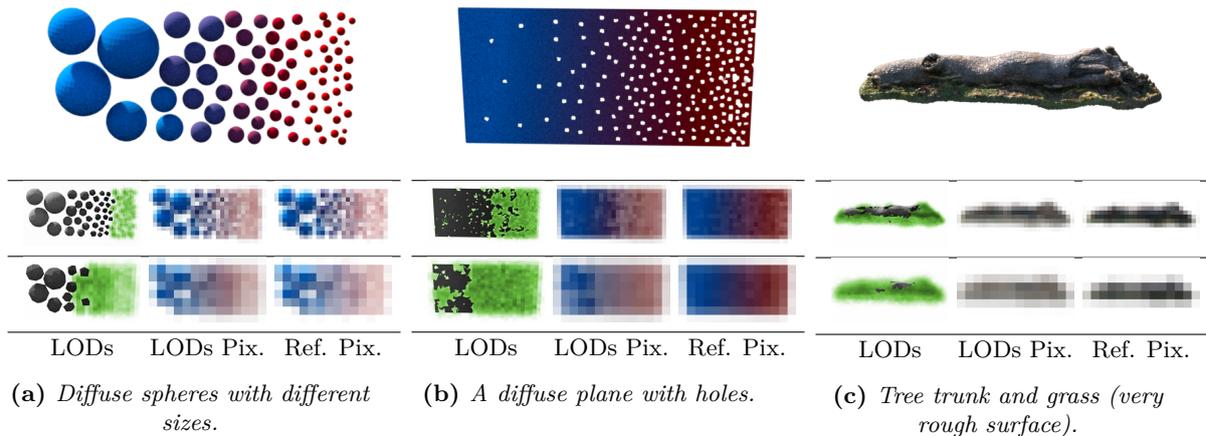


Figure 10.5: Our hybrid LODs for 3 input meshes (top). Volumetric parts are shown in green (left columns). (b) and (c) show some limitations of our method: our mesh analysis cannot detect surface-like appearance when the surface is disconnected, has holes (b) or is not smooth enough, e.g. because of the grass under the trunk in (c). Our volumetric approximation is too transparent in such cases.

10.5 Summary of the chapter

In this chapter, we presented results generated with our LOD pipeline that combines the contributions presented in the previous chapters:

- We discussed the implementation of the hybrid LOD pipeline, and in particular seamless transitions.
- We showed that our method generates accurate LODs for complex assets such as trees, with consistent appearance across scales and correct occlusion.
- We have discussed the limitations of our method coming from the limitations of each sub-part of our LOD pipeline. The main limitation is that our macrosurface analysis does not always detect surface-like appearances (due to non-manifold geometry or complex surfaces), which can lead to inaccurate volumetric approximations.

10.6 Résumé du chapitre

Dans ce chapitre, nous avons présenté des résultats produits avec notre méthode de préfiltrage hybride, qui combine les contributions proposées dans les chapitres précédents :

- Nous avons détaillé l'implémentation de notre méthode de préfiltrage hybride, et en particulier les transitions entre les différentes résolutions.
- Nous avons montré que notre méthode produit des résultats précis pour des objets 3D complexes comme des arbres, avec des apparences cohérentes entre les échelles et des transparences correctes.
- Nous avons décrit les limitations de notre approche provenant des limitations des différentes sous-parties de notre algorithme. La limitation principale de nos travaux est que notre analyse de macrosurface ne détecte pas toujours les apparences qui ressemblent à des surfaces si la géométrie sous-jacente est déconnectée, auquel cas les approximations volumiques manquent de précision.

Part III

A new microflake model with microscopic shadowing for volume downsampling

High-resolution voxel grids can be used for representing and rendering complex volumetric appearances. 3D grids can be generated from CT scanners [ZJMB11], physical simulations, procedural techniques or voxelized 3D geometry (Chap. 9). Rendering such high-resolution data with path-tracing techniques is very challenging because of large loading times and substantial memory access costs.

In this part of the dissertation, we address the problem of downsampling voxel grids containing the parameters of a heterogeneous participating medium, so that the medium can be rendered efficiently from a distance. As in part II, our main goal is appearance-preserving prefiltering at all scales. We show in chapter 11 that preserving the appearance of heterogeneous media is not as easy as it seems, and that naïve downsampling methods have insufficient accuracy in several important cases.

Our work is closely related to the previous work of Zhao, Wu et al. [ZWDR16] and we review it in Sec. 11.3.3. We address the same problem in a different way and we show that our approach is faster and supports various types of volume that were not accurately downsampled by their method. We introduce an extension of the microflake model [JAM*10a] by adding parameters describing the anisotropic correlation of microflake positions at a microscopic scale (Chap. 12). We discuss the implementation of our model in chapter 13, and we propose new downsampling algorithms based on this model in chapter 14.

Chapter 11

Problem analysis: LODs of voxel-based heterogeneous participating media

11.1 Introduction

11.1.1 High-resolution heterogeneous participating media

Heterogeneous participating media are commonly used in computer graphics for rendering complex semi-transparent media such as clouds, smoke and marble. They are also used as approximations of aggregate detail such as fabrics and granular materials (Chap. 5). The heterogeneity of such media is almost always described explicitly by regular grids of cubic voxels, each voxel storing scattering parameters of the volume model used for rendering. As introduced in Sec. 5.3, commonly used parameters include attenuation coefficients, density of microflakes [JAM*10a], scattering coefficients, microflake albedos and parameters for phase functions. At rendering, scattering parameters at a given position in space are obtained either by looking at the nearest voxel center or with some kind of interpolation, typically trilinear interpolation.

As discussed in chapter 3 and 5, voxel grids are used in many computer graphics applications for storing various types of data such as normals, distance fields [HN12], albedos, view-dependent attributes [CNS*11] or radiance [MWM08]. Therefore, the words *volume* or *volume LODs* refer to different algorithms and techniques that are not always related to semi-transparent media. In this part of the thesis, we focus on voxel-based heterogeneous participating media, and we use the word *volume* to designate participating media with scattering models based on the radiative transfer equation (Sec. 5.3).

High-resolution heterogeneous media in computer graphics often come from physical simulations, scanned data [ZJMB11], and voxelization of complex geometry ([MWM08, SKZ11], Chap. 9). This data often have a high resolution, and memory usage expresses in gigabytes. Rendering such data is challenging because the spatially-varying attenuation must be integrated along each ray and memory access operations are time consuming.

11.1.2 Appearance-preserving volume downsampling

Level-of-detail representations for volumes can bring huge benefits in terms of rendering time when the input voxels are sub-pixel or off-screen. Voxel grids are well suited for LODs because of their well-organized structure. If we know how to replace each block of $n \times n \times n$ voxels by a single low-resolution voxel without impacting the appearance of the volume from a distance, then the resolution of the voxel grid can be adapted for a given frame and rendering is more efficient. The problem of simplifying volumes can be summarized by the following question: *how can we approximate a block of $n \times n \times n$ voxels with a single low-resolution voxel, in order to reduce memory usage but without changing the appearance*

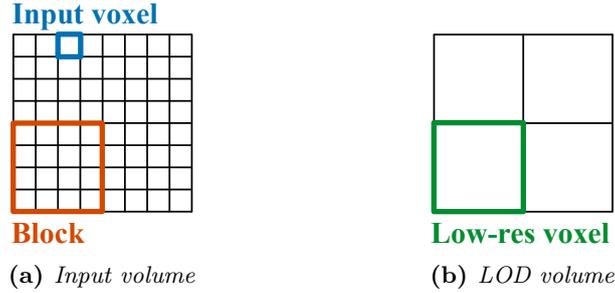


Figure 11.1: A simple framework for volume LODs consists in approximating blocks of input voxels (a) by low-resolution voxels (b).

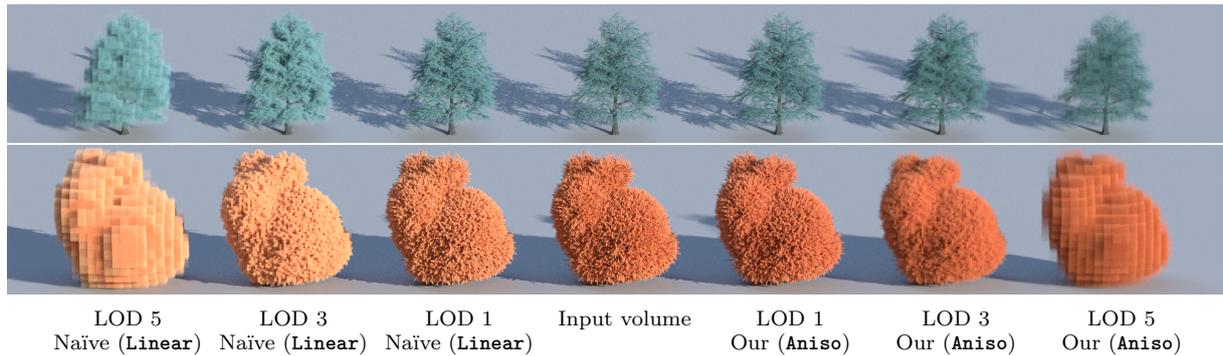


Figure 11.2: Comparison between naïve downsampling of microflake volumes and our method (*Aniso*, Chap. 14). Naïve downsampling of dense heterogeneous volumes often lead to inaccurate LODs, due to the loss of masking and shadowing effects that occur between and inside dense input voxels. Rendered with volume path tracing (the trunk of the cedar is a mesh), without trilinear interpolation.

from a distance? Generating low-resolution grids from high-resolution grids is sometimes called *volume downsampling* [ZWDR16]: voxels are viewed as samples of a 3D signal, and low-resolution voxel grids allow for reconstructing a low-frequency (filtered) signal from less samples. A simple downsampling framework is illustrated in figure 11.1.

In this part of the thesis, we address the problem of generating volume LODs such that the appearance of the heterogeneous participating medium is preserved at all scales when rendered with a full computation of the volumetric multiple scattering. Examples of results are shown in figure 11.2. In chapter 4, we have seen that linear texture downsampling can lead to inaccurate results, in particular if scattering parameters have a non-linear relation to the appearance, and if various scattering parameters are correlated (for example the slopes of a surfaces and its albedos). The same phenomena occur for volumes, and we will give various examples in the next section.

11.1.3 Evaluating the quality of volume LODs

Image-space evaluation. The accuracy of LODs can be evaluated using image-space errors, by comparing rendered LODs with results obtained from high-resolution data. Ideally, such errors should be computed for many different lighting setup and from many points of view, in order to evaluate the accuracy of LODs in all possible rendering contexts.

Local evaluation. An alternative to the view- and light-dependent image-space errors consists in evaluating LODs *locally*. For example, light scattering probabilities in each low-resolution voxel can be compared with light scattering in the corresponding blocks of input voxels. In the same spirit as in the

standard definition of BRDFs (Sec. 4.1), the radiative transfer in a voxel or in a block of voxels can be characterized by a 9D function

$$T(p_i, \omega_i, p_o, \omega_o, \lambda) = \frac{dL_o(p_o, \omega_o, \lambda)}{L_i(p_i, \omega_i, \lambda) d\omega_i dp_i},$$

with 2D for the direction ω_i of the incident radiance, 2D for its position p_i , and similar dimensions for the outgoing radiance (ω_o and p_o). The last dimension is the wavelength λ .

For the local evaluation of LODs, we can assume that only the directions need to be considered, because the exact incident and outgoing positions p_i and p_o are not important when the volume is rendered from a distance. Under this far-field assumption, we can restrict the local evaluation to a function $T(\omega_i, \omega_o, \lambda)$. It is convenient to omit light transmitted without scattering through the volume in the definition of T , and to describe the directional transparency using a separate function.

Effective albedo and phase function. From the definition of the local transfer function T , we can introduce the concept of *effective albedo*, that is the average scattering probability given an incident direction ω_i :

$$\alpha^{\text{eff}}(p_i, \omega_i, \lambda) = \frac{1}{|\mathcal{P}|} \int_{\mathcal{P}} \int_{S^2} T(p_i, \omega_i, p, \omega, \lambda) d\omega dp$$

with \mathcal{P} the surface of the cube of medium and $|\mathcal{P}|$ its area. Contrary to the single scattering albedo, this effective albedo takes into account multiple scattering in the volume, and is related to the apparent color of the medium. Similarly, we can define an *effective phase function* $f^{\text{eff}}(p_i, \omega_i, \lambda, \omega_o)$ that takes into account all scattering orders locally. We will rely on these concepts in the following chapters.

Advantage of local LOD evaluation. Evaluating the accuracy of local transfer functions is more efficient than computing errors in image space, because local computations can be fully parallelized, each task only requires a small part of the entire data. Nevertheless, local evaluations of LODs give less guaranties about the appearance of the entire LOD viewed from a distance, because effects of masking and multiple scattering between neighboring blocks of voxels are not taken into account. In chapter 14, we optimize scattering parameters using similar local measurements and we show that this is more efficient than image-space optimization, and sufficient to ensure a correct appearance of the entire medium.

11.2 Case study

In the following examples, we consider voxels as cubes of participating media for simplicity: we assume that box reconstruction filters are used for downsampling, and that volumes are rendered without trilinear interpolation. The phenomena highlighted in these examples also occur if smoother filters are used for rendering and prefiltering. We first present some examples where scattering parameters can be linearly prefiltered while preserving the appearance of the volume, before discussing more complex cases. We do not address all these cases in the next chapters, this section aims at giving intuitions about what governs the appearance of heterogeneous volumes and at explaining why volume downsampling is challenging.

11.2.1 Examples where linear downsampling is accurate

Let's first consider a block of voxels such that the scattering parameters are exactly the same in all the voxels (Fig. 11.3a). In this specific case, prefiltering linearly the parameters and replacing the block by a single low-resolution voxel (Fig. 11.3b) preserves exactly the appearance of the volume (*i.e.* its opacity, effective albedos and effective phase functions).

Intuitively, we see that linear prefiltering would be a good approximation in cases where the block is *almost* homogeneous (Fig. 11.3c). For example, if there are only subtle variations of density between the voxels of the block, averaging the parameters and replacing the block by a unique low-resolution voxel should be accurate enough.

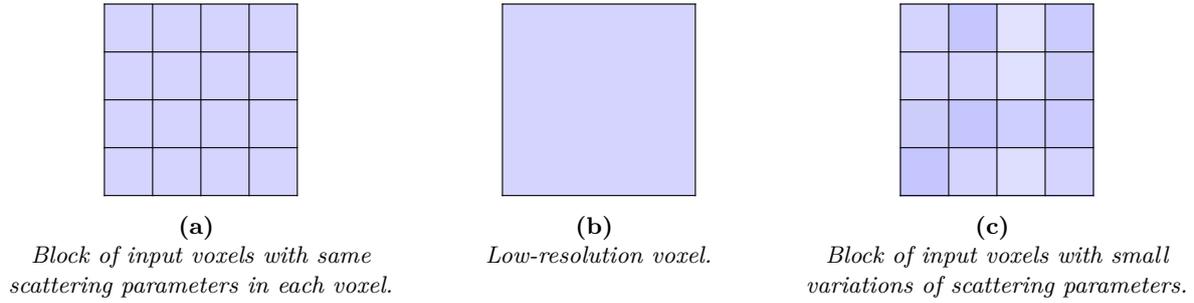


Figure 11.3: If all the voxels of a block have the same scattering parameters (a), then the appearance can be exactly preserved in the LOD using the same parameters (b). Linear prefiltering is often an accurate approximation for the case of almost homogeneous blocks of voxels (c).

There are other cases where linear prefiltering gives good approximations. Let's consider a block of voxels with relatively low attenuation coefficients and spatially-varying scattering coefficients (Fig. 11.4a). For instance, let's consider a block of $4 \times 4 \times 4$ voxels, each voxels blocking 5% of the light in each of the three main directions (the medium is isotropic). Then, for an incident light beam in one of the main directions and going through four voxels, light is scattered in the four voxels with probabilities of approximately 0.05, 0.048 (that is, $0.05 \times (1 - 0.05)$), 0.045 and 0.043, from the first intersected voxel to the last one. In this example, we can consider that scattering occur in all the voxels with roughly the same probability, and also that single scattering is largely predominant compared to multiple scattering, because most of the scattered light leaves the block with only one scattering event. Under these assumptions, it is correct to assume that the spatial variations of the scattering coefficient do not really matter: seen from a distance, the medium can be accurately described statistically by the average scattering coefficient. In this case, it would be accurate enough to prefilter scattering coefficients linearly (Fig. 11.4b).

These assumptions hold for other low-density volumes, for example microflake volumes with uniform attenuation coefficients and scattering coefficients but spatially-varying microflake normal distributions (Fig. 11.4c). In this case, it does not matter where each type of microflake is located in the block, the average microflake normal distribution is sufficient for describing the medium under a far-field assumption (Fig. 11.4d).

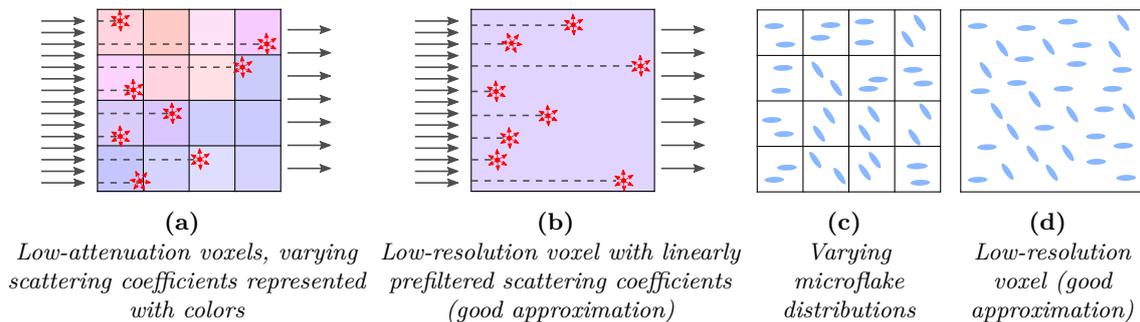


Figure 11.4: Under the hypothesis that voxels in the block are transparent enough (a, c), some linear prefiltering operations give accurate approximations (b, d) because the spatial variations do not matter from a distance. The red arrows in (a) and (c) represent scattering events (they occur everywhere in the volume because the voxels are transparent enough).

11.2.2 Examples with correlations

When scattering parameters are correlated, it is sometimes inaccurate to prefilter each parameter independently. We have already discussed this effect in the case of surfaces in chapter 4. A volumetric example is given in figure 11.5 with correlations between albedos and microflake distributions.

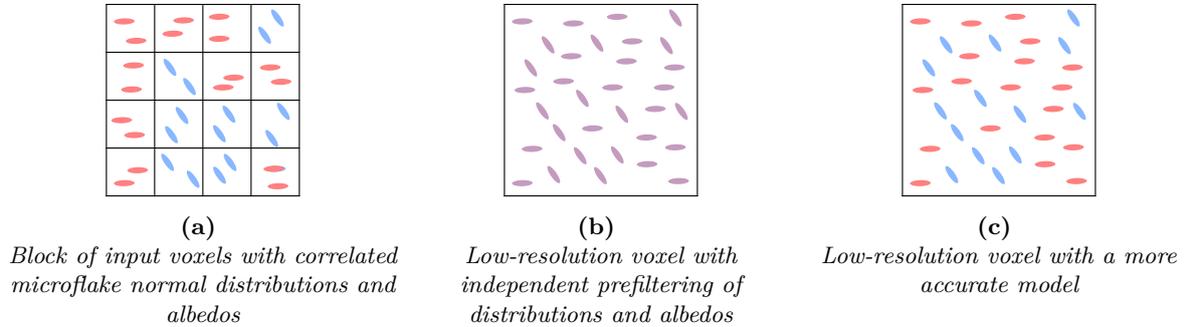


Figure 11.5: When scattering parameters such as microflake albedos and microflake distributions are correlated in the block of input voxels **(a)**, independent linear prefiltering for each parameter lacks accuracy: in **(b)**, light scattered by the volume is purple in all directions, while scattered light can be red or blue depending on the scattering direction in the input volume **(a)**. In this example, using two types of microflakes in the low-resolution voxel **(c)** allows for preserving more accurately the radiative transfer of the block of input voxels.

Another common example of correlations concerns density of microflakes and microflake albedos (Fig. 11.6). If the input block contains opaque voxels with red microflakes and very transparent voxels with blue microflakes, we intuitively know that microflake albedos should be weighted by the density of microflakes and not prefiltered independently of density parameters. Note that unlike albedos, scattering coefficients can usually be prefiltered linearly and independently of attenuation coefficients because they directly represent the radiative transfer by unit of length in the media.

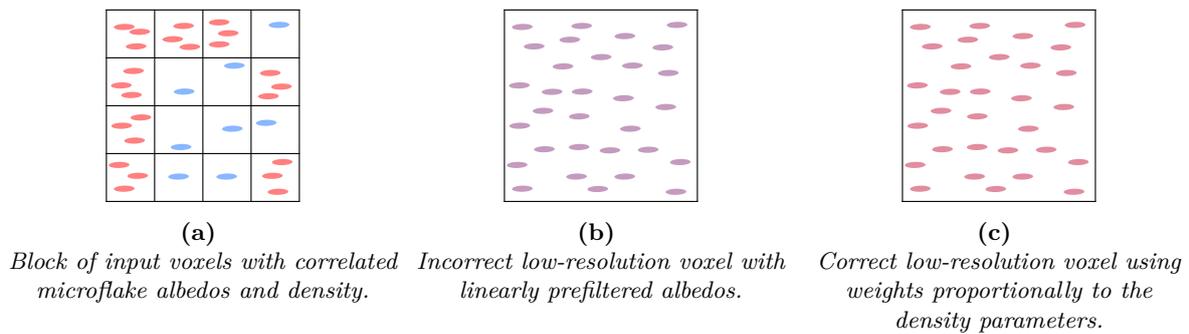


Figure 11.6: Example where simple linear prefiltering leads to inaccurate results for the low-resolution voxel. In the block of input voxels, density parameters and albedos are correlated **(a)**, leading to incorrect appearance if albedos are prefiltered independently of density parameters **(b)**. Using weights proportionally to the density parameters solves the problem in this case **(c)**.

11.2.3 Incorrect opacity with linearly prefiltered density

We now consider a $4 \times 4 \times 4$ block with a very opaque voxel (with attenuation coefficient σ_t^{input}) surrounded with transparent voxels (Fig. 11.7a). In this case, the opacity of the block in one of the main direction is approximately $1/16$. By linearly downsampling the attenuation coefficients, we get an attenuation

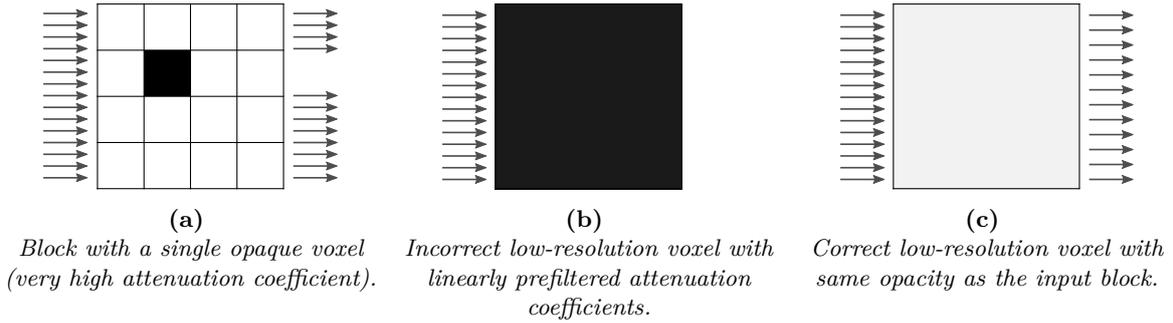


Figure 11.7: *The linear prefiltering of attenuation coefficients can lead to inaccurate results with incorrect opacity, especially in the case of very heterogeneous volumes.*

coefficient of $\sigma_t^{\text{input}}/16$ for the low-resolution voxel, and its opacity is

$$1 - \exp\left(\frac{-d_v \sigma_t^{\text{input}}}{16}\right)$$

with d_v the length of the low-resolution voxel. We can see that if σ_t^{input} is very large, we get a very opaque low-resolution voxel that would be an inaccurate approximation of the input volume. The same thing occurs with the microflake model if density is prefiltered linearly in very heterogeneous blocks. This is the reason why we propose to estimate density parameters that preserve the opacity in the next chapters, instead of relying on linear prefiltering.

11.2.4 Examples with incorrect effective albedo and effective phase function

We now consider again the previous example with a single opaque voxel with attenuation σ_t^{input} in a block of $4 \times 4 \times 4$ input voxels (Fig. 11.7a), but we now consider the scattering inside this volume. We call σ_s^{input} the scattering coefficient of the opaque voxel, and all the other voxels are transparent. Ideally, a low-resolution voxel approximating this block should have the same occlusion, effective phase functions and effective albedos, that is, light should be scattered with the same probability and in the same directions. A light ray intersecting the low-resolution voxel (along one of the main axes) is going through the medium over a length that is 4 times the length of high-resolution voxels. Therefore, if we set the attenuation and scattering coefficients of the low-resolution voxel to respectively $\sigma_t^{\text{input}}/4$ and $\sigma_s^{\text{input}}/4$, then the scattering of light in the low-resolution voxel is exactly similar to light scattering in the input opaque voxel (Fig. 11.8a and 11.8b): we simply *scaled* the medium such that the effective albedo and phase function of the high-resolution dense voxel are preserved in the low-resolution voxel. However, the occlusion of the low-resolution voxel is incorrect: we know that even linear pre-filtering (giving $\sigma_t^{\text{input}}/16$ for the low-resolution voxel) results in an overly opaque voxel (Fig. 11.8d).

Now if we choose an attenuation coefficient that gives a correct opacity for the low-resolution voxel (Fig. 11.8e), light scattering becomes quite different from the scattering in the input voxel:

- The low-resolution voxel becomes a transparent medium in which scattering occurs everywhere in the volume (Fig. 11.8e), whereas it was occurring mostly in the vicinity of the boundaries of the opaque voxel in the input block of voxels (Fig. 11.8a and 11.8b). This means, for example, that light entering the low-resolution voxel in direction ω_i could be scattered forward and leave the voxel in the same direction. These light paths have a very low probability in the input opaque voxel. In other words, the effective phase function of the low-resolution voxel is not accurate.
- Single scattering is largely predominant in the low-resolution voxel (Fig. 11.8e), whereas in the input opaque voxel multiple scattering contributed a lot to the appearance. Because light paths in the low-resolution voxel have less scattering events, the effective albedo of this voxel is brighter and less saturated.

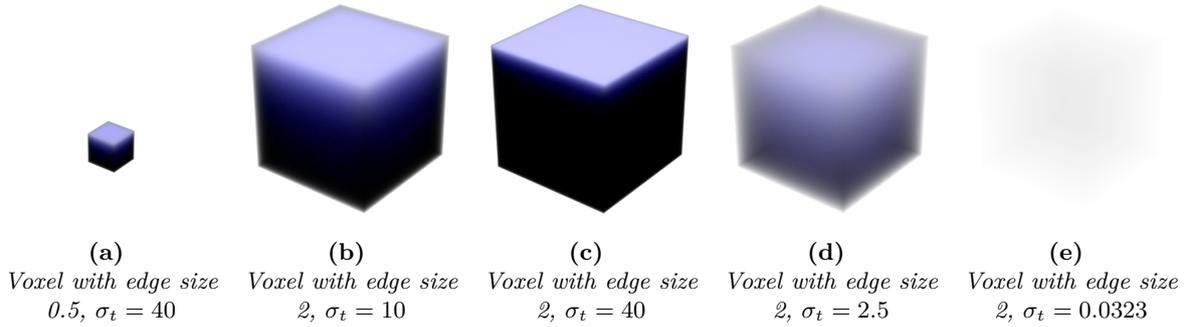


Figure 11.8: Cubes of participating media rendered with a top light (spot light), a RGB albedo of (0.8, 0.8, 0.97) and the Henyey-Greenstein phase function ($g = 0.3$). It is possible to preserve the appearance of a voxel while scaling it by dividing the attenuation coefficient and the scattering coefficient by the scale factor, 4 in this example (**a**, **b**). Otherwise, the effective albedos and effective phase functions are not preserved (**c**, **d**, **e**). Case (**d**) corresponds to the linear prefiltering described in Fig. 11.7a (the attenuation coefficient is divided by 16), and (**e**) corresponds to a medium with an opacity of 1/16 (Fig. 11.7c). In low-opacity voxels (**e**), light is scattered everywhere in the voxel, it can leave the voxel in every direction depending on the phase function, and multiple scattering does not contribute a lot to the radiative transfer. In denser voxels (**b**, **c**), light is mainly scattered in the vicinity of the entry point, it leaves the voxel backward because it cannot reach the opposite side of the voxel, and multiple scattering creates more saturated colors in some directions. Therefore, effective albedos and phase functions in (**c**, **d**, **e**) are different from (**a**) and (**b**).

From this example, we understand that it seems difficult to preserve both a correct opacity and correct effective albedos and phase functions in the low-resolution voxel when the input medium is very heterogeneous and has very opaque voxels. This is the fundamental problem we address in the next chapters, by introducing a new volume model with parameters for characterizing the probability of self-shadowing in the medium, independently of its attenuation coefficient.

11.2.5 Examples of view-dependent appearances

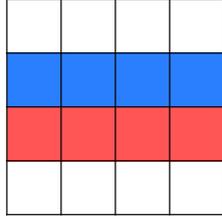
In the general case, a block of $n \times n \times n$ input voxels can have a wide range of appearances, including complex view-dependent opacity and view-dependent colors. When we try to approximate such complex appearances with a single voxel of participating medium, we face a fundamental problem: light scattering in the input block has little to do with the scattering in a homogeneous cube of participating medium. Therefore, it is difficult and sometimes impossible, with available volume models, to accurately approximate the input volume with low-resolution voxels while preserving the opacity and local effective albedos and phase functions. Examples are shown in Fig. 11.9.

As for surface prefiltering, volume downsampling always requires hypotheses about the input data so that LOD methods can be derived. For instance, we can often assume that albedos and phase functions are not correlated, which allows for prefiltering phase functions independently of albedos. This assumption does not hold for every dataset, for example for some types of fabrics [ZWDR16].

11.3 Previous work on volume appearance matching

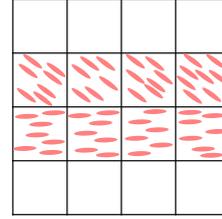
11.3.1 The lack of analytical models for effective albedos and phase functions

We have seen in the previous section that the distant appearance of participating media can be characterized by their local directional attenuation, their effective phase functions and their effective albedos



(a)

Block of voxels with a view-dependent appearance, due to colored dense voxels.



(b)

Block of voxels with a view-dependent appearance due to spatially-varying microflake distributions.

Figure 11.9: The appearance of a block of voxels can have strong view-dependent effects that are difficult to preserve accurately using a single voxel of participating medium. View-dependent effects may be due to spatially-varying albedos or microflake distributions in the block of voxels.

(Sec. 11.1.3). Our goal is to downsample volume data while preserving such properties in low-resolution volumes, and it would be useful to obtain analytically such functions directly from scattering parameters.

Unfortunately, it is difficult to predict the effective albedos and effective phase functions of a voxel of participating medium given its scattering parameters, because of the complexity of multiple scattering in a medium with arbitrary density and phase functions. The lack of models for the *effective* appearance of voxels raises two main problems for volume LODs:

- First, it is difficult to derive closed-form expressions for the appearance of a block of input voxels in terms of effective phase functions and effective albedos. Both can be evaluated numerically using Monte Carlo approaches but this is time consuming and prone to noise because the full volume light transport must be integrated. Nevertheless, we rely on such sampling methods in the next chapters (Chap. 14), using appropriate hypotheses for simplifying the problem.
- Secondly, given a *target* appearance (for example a target effective albedo estimated in the block of input voxel), it is difficult to derive scattering parameters for a low-resolution voxel. Because of the lack of analytically models, we address this problem with an numerical approach (Chap. 14).

In fact, some models are available for the appearance of low-density volumes and very opaque voxels. Light scattering in low-attenuation volumes is simple because multiple scattering can be neglected. Effective albedos are roughly equivalent to single scattering albedos, and effective phase functions to single scattering phase functions. To the contrary, scattering in very dense media occur in the vicinity of the entry point and light cannot reach the opposite side of the voxel. Light scattering in this situation is similar to the case of a homogeneous slab of medium of infinite width, for which sub-surface scattering models have been studied, for both isotropic media (with angle dependent phase functions) [JMLH01, DI11, HCJ13] and anisotropic media [JAM*10a]. Solutions used in computer graphics rely on a diffusion approximation, in which the radiance due to multiple scattering in the volume is assumed to have low angular frequencies, such that it can be approximated using its first moments (*i.e.* a constant lobe plus a cosine lobe). Then, practical solutions for the radiance at the surface are derived from the choice of boundary conditions and approximations of the first scattering events in the medium. Has noted by Hery, Wrenninge and Villemin [Her03, WVH17], closed-form expressions for the effective albedo allow for controlling directly the effective appearance of diffusion-based sub-surface scattering models, and such controllability is very important for 3D artists.

In conclusion, we have closed-form expressions for the effective albedos and effective phase functions in volumes with either low or high optical depth, but we have no model for the situations in between, where the diffusion approximation does not hold and multiple scattering and shadowing impact both the effective albedo and the effective phase functions. We don't contribute to this problem and we rely on numerical estimations of effective albedos (Chap. 14). Efficient closed-form solutions remain to be found.

11.3.2 Previous work on the relationships between scattering parameters and the appearance of media

The problem of finding the relationship between scattering parameters and the effective appearance of a volume has been noted and addressed by several authors in computer graphics. Zhao et al. [ZJMB11] have used microflake volumes [JAM*10a] for rendering fabrics and they estimated scattering parameters of their volume model from photographs. Assuming that roughness parameters of microflake distributions and albedos were uniform in the entire volume, and using the fact that density parameters have little effect on the appearance of dense volumes, they optimized the albedo and the roughness by iteratively updating one of the parameters until the image-space statistics of the rendered volume matched the statistics of reference photographs.

Retrieving the relation between scattering parameters and global appearance has then been improved by estimating derivatives of pixel values with respect to scattering parameters, for the acquisition from photographs of real-world liquids [GZB*13] and fabrics [KSZ*15], and for the interactive editing of volume albedos [HR13]. Estimating such derivatives allows faster convergence of the appearance matching process because stochastic gradient descent methods can be used and all the parameters are optimized jointly at each iteration. Zhao et al. [ZRB14] used the similarity theory [WPW89] to explore sets of scattering parameters that lead to the same global appearance, and showed that the appearance matching problem may have several solutions with similar accuracy for optically dense and high-albedo media. They showed that the non-linear relation between scattering parameters and pixel radiances can be leveraged for decreasing scattering coefficients in dense media and reducing significantly render time without changing the appearance.

Similar problems have been noted concerning the complex relation between the global appearance of dense packs of fibers such as hair and their individual scattering properties. The non-linear relationship between the perceived color and scattering parameters of fibers such as azimuthal roughness and transmission color makes authoring difficult for artists. Chiang et al. [CBTB16] addressed this problem by pre-rendering images of fibers with different scattering parameters and finding a mapping between perceived colors and scattering parameters. The mapping function was deduced from observations and least-square fitting, and more general analytic models are still missing. Chiang et al. [CKB16] proposed a similar numerical inversion for the appearance of path-traced media.

11.3.3 Downsampling scattering parameters [ZWDR16]

Zhao, Wu et al. [ZWDR16] recently addressed the problem of downsampling very heterogeneous volumes while preserving their appearance, in the case of microflake volumes rendered with volume path tracing. Their work is closely related to previous work on appearance matching [ZJMB11, GZB*13, KSZ*15] because they have optimized albedos of low-resolution voxels such that the appearance of LOD volumes matches the appearance of the reference high-resolution volumes. As in previous work, they rely on iterative optimization using partial derivatives of pixel values with respect to scattering parameters [GZB*13, KSZ*15]. In the next chapters, we introduce a new downsampling approach for generating accurate LODs of microflake volumes. Because we address almost exactly the same problem as Zhao, Wu et al. [ZWDR16], we review several aspects of their method in this section. The main difference between our work and their method is that we optimize scattering parameters locally for each low-resolution voxel, instead of using global optimization based on image-space error metrics which restricts the generality of their downsampling algorithm.

Overview

The input data of Zhao, Wu et al. [ZWDR16] consists of high-resolution voxel grids, in which each voxel stores spatially-varying scattering parameters for the microflake model (Sec. 5.3, [JAM*10a]): microflake densities (scalar value), microflake albedos (RGB value), and parameters for the microflake distribution (one SGGX lobe, 6 coefficients).

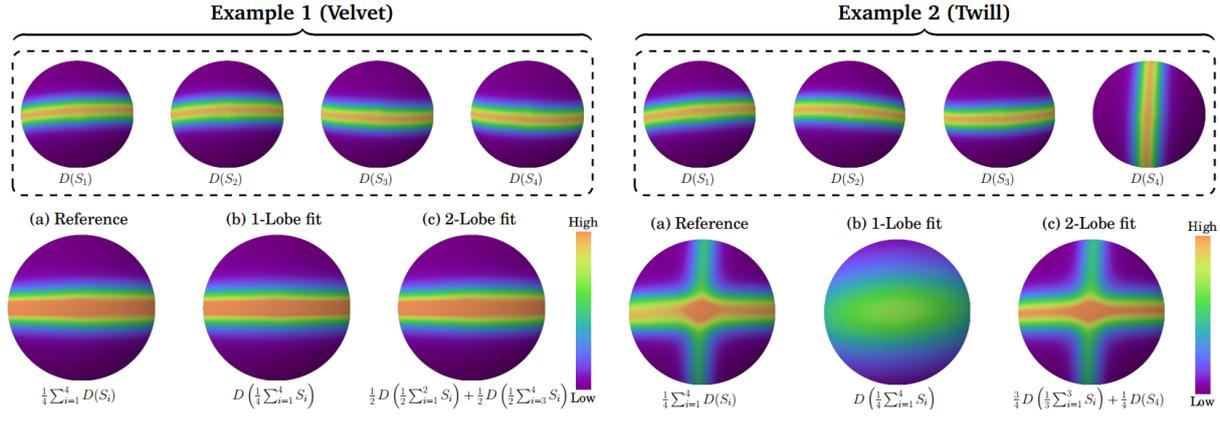


Figure 11.10: Figure from the work of Zhao, Wu et al. [ZWDR16]. Using a single SGGX lobe for prefiltering microflake distributions is a good approximation when microflake distributions have low frequencies or are coherently oriented (example 1). However, a single lobe is insufficient when the combined distribution is more complex (example 2), for example when input voxels have fiber-like microflake distributions with distinct fiber orientations. The downsampling algorithm of Zhao, Wu et al. [ZWDR16] supports multiple SGGX lobes in low-resolution voxels.

They also use the standard microflake model for low-resolution voxels, but they use multiple SGGX lobes with different scattering coefficients for each lobe. As shown in Fig. 11.10, using several lobes allows more accurate prefiltering, in particular in the case of fabrics with specular fibers. They also handle the case of correlated albedos and microflakes distributions, as mentioned previously (Fig. 11.5) and shown in Fig. 11.11c (twill with red and yellow fibers).

Their method consists in the following steps:

1. For each low-resolution voxel, they **linearly downsample the density** of input voxels (*i.e.* they average density parameters). We have seen that this often leads to incorrect opacity (Sec. 11.2.3) but has sufficient accuracy for opaque media with simple silhouettes when the asset is seen from a distance.
2. For each low-resolution voxel, they **find a small number of SGGX lobes** for accurately representing the combined microflake distribution of input high-resolution voxels.
3. They **cluster the low-resolution voxels** into a few clusters with similar scattering parameters, so that they can optimize scattering parameters per cluster, instead of per low-resolution voxel. Clustering is mandatory in their method because the enormous amount of parameters that need to be optimized makes the problem intractable otherwise. This strategy requires the underlying hypothesis that blocks of input voxels with the same average albedo should have the same effective albedo. This assumption would be incorrect in datasets with both high-density and low-density regions with the same average albedo, and it raises problems for spatially-varying datasets because a large number of clusters should be used to avoid artifacts. Their method also requires solving additional problems such as finding SGGX lobe correspondence between low-resolution voxels belonging to the same cluster, because scattering parameters associated with each lobe are optimized for the entire cluster.
4. They **estimate scattering coefficients** for each lobe and for each cluster with a stochastic gradient descent, using the input high-resolution volume as the reference and iterating until reaching a local minimum of image-space errors. The scattering parameters of each cluster of low-resolution voxels are optimized using a stochastic gradient descent based on rendered images of the low-resolution volume and reference images of the high-resolution volume. They render volumes and compute error metrics for several camera positions and lighting conditions. Their error metric does not handle the case where a pixel value comes from several semi-transparent voxels belonging to different clusters, it only considers the first voxel seen by the pixel. This strategy implicitly assumes that all the low-resolution voxels are dense enough. Optimizing scattering parameters using image-space metrics

in the case of semi-transparent volumes would be more challenging and the gradient descent more complex.

Discussion

Zhao, Wu et al. demonstrated that their method can be used for generating accurate LODs of microflake volumes for which naïve linear downsampling would produce incorrect results. Results from their publication can be found in Fig. 11.11. Their test datasets include complex fabrics with correlations between microflake albedos and microflake orientations, and their method preserves the colored anisotropic reflections of the input datasets (Fig. 11.11c). Their work have the following main limitations:

- In their method, density parameters are downsampled linearly, which produce incorrect silhouettes, shadows and appearance for datasets with complex macroscopic shapes such as trees. Although the strategy for prefiltering density parameters could be changed in their method, it is unclear how the optimization algorithm should be adapted for robust parameter estimation in the case of semi-transparent voxels.
- Voxel clustering is mandatory in their method and prone to artifacts. They noted artifacts in the case of spatially-varying albedos and proposed an ad hoc jittering method which lacks generality and robustness. Other parameters such as density or roughness could also vary spatially, leading to other kinds of artifacts when using clusters. Using more clusters makes their algorithm less efficient because more gradient images need to be computed.

The method we introduce in the next chapters has similar goals but it bypasses these limitations: we estimate the scattering parameters independently for each low-resolution voxel using local metrics (mean effective albedo), which is faster, avoids the use of voxel clusters and allows for simplifying all kinds of spatially-varying datasets for the same precomputation cost. We do not rely on linear downsampling of density parameters, and we estimate density parameters that ensure correct transparency, shadows and silhouettes in our LODs.

11.4 Summary of the chapter

Rendering high-resolution heterogeneous participating media is challenging because of the memory requirements of large 3D voxel grids and costly path tracing. We have presented the problem of volume downsampling and we have introduced the definitions of *effective albedos* and *effective phase functions* for characterizing the appearance of a voxel or a block of voxels (Sec. 11.1.3).

Based on simple examples, we have shown that simplifying volumes while preserving their appearance is challenging:

- Linear prefiltering does not produce accurate results. In particular, averaging attenuation coefficients leads to low-resolution voxels with incorrect opacity.
- As in the case of surface prefiltering, scattering parameters such as density, albedos and microflake normal distributions should not be downsampled independently in cases where they are correlated.
- Blocks of high-resolution voxels can have some types of view-dependent appearance that are not easily approximated by common volume models.
- We have explained that generating low-resolution voxels with correct opacity, effective albedos and effective phase functions is impossible in some cases using simple volume models (Sec. 11.2.4), in particular because decreasing attenuation coefficients affects the amount of self-shadowing and multiple scattering in the media.

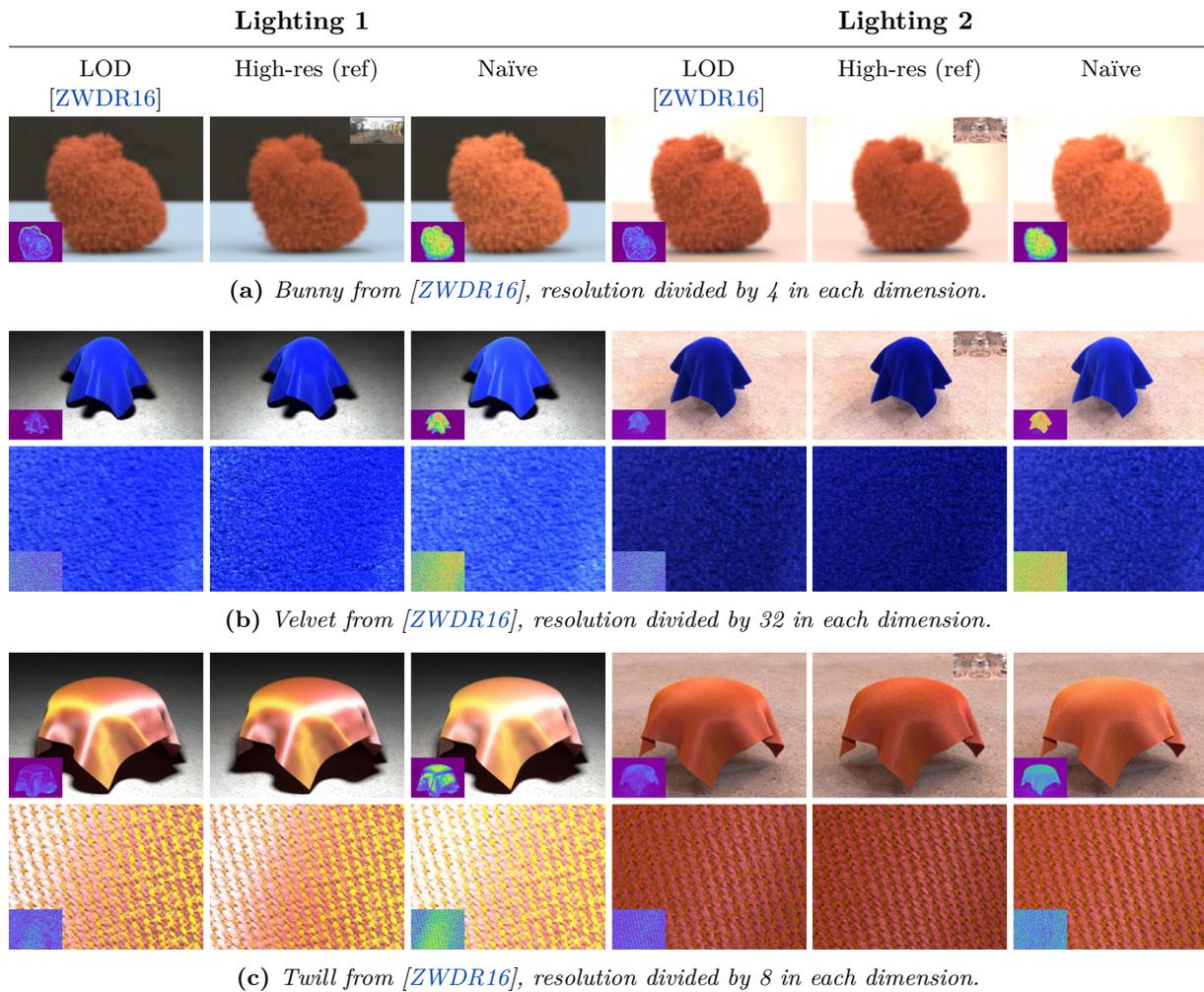


Figure 11.11: Results from the work of Zhao, Wu et al. [ZWDR16]. Their test datasets include fabrics with aligned dense voxels at the voxel scale (a,b) and oriented colored fibers (c). They have shown that optimizing albedos in low-resolution voxels allows for generating LODs with improved accuracy compared to naïve prefiltering, with up to four orders of magnitude storage saving.

We have presented some previous work on volume downsampling and non-linear relationships between the scattering parameters and the effective appearance of participating media, and we have highlighted the lack of analytical models.

11.5 Résumé du chapitre

Calculer efficacement le rendu de milieux participatifs hétérogènes de haute résolution est difficile à cause de la place que prennent les grilles 3D de voxels en mémoire. Nous avons présenté le problème du préfiltrage volumique et nous avons introduit les concepts d'albedos effectifs et de fonctions de phases effectives pour caractériser l'apparence d'un voxel ou d'un bloc de voxels (Sec. 11.1.3).

En partant d'exemples simples, nous avons montré que préfiltrer des volumes tout en préservant leur apparence peut être difficile :

- Le préfiltrage linéaire n'a pas une précision suffisante dans certains cas. En particulier, moyenner les coefficients d'atténuation donne des transparences incorrectes dans les volumes préfiltrés.
- Comme dans le cas du préfiltrage de surfaces, les paramètres de réflectance comme la densité, les

albedos ou les distributions de normales de microflakes ne devraient pas être préfiltrés indépendamment les uns des autres lorsque leurs valeurs sont corrélées.

- Les blocs de voxels de haute résolution peuvent avoir des apparences qui dépendent beaucoup de l'angle d'observation, et les modèles volumiques existants ne donnent pas toujours des approximations fiables.
- Nous avons expliqué que préfiltrer des volumes avec des transparences et des apparences correctes est parfois impossible avec les modèles volumiques existants (Sec. 11.2.4, en particulier parce que la diminution de la transparence d'un milieu va de pair avec la diminution des probabilités d'ombrage local et change la manière dont la lumière est diffusée.

Nous avons présenté des méthodes existantes pour le préfiltrage des volumes ainsi que des travaux portant sur l'étude des relations non linéaires entre les paramètres de réflectance et l'apparence des volumes.

Chapter 12

A new microflake model with microscopic self-shadowing

In this chapter, we introduce a new participating medium model based on the microflake model. We call it the *microscopic self-shadowing model* because it models media whose microflakes have correlated positions at a microscopic scale, leading to microscopic shadowing and masking effects. Examples are shown in Fig. 12.1. Microscopic self-shadowing impacts attenuation coefficients, as well as scattering coefficients and phase functions. The key motivations for this model are the following:

- In our new model, the anisotropic attenuation due to anisotropic correlations of microflake positions is independent of the microflake distribution. This means that our model can be used for pre-filtering microflake distributions and anisotropic attenuation independently. For example, our model can have isotropic microflake distributions and anisotropic attenuation (Fig. 12.1a), which was not possible with the standard microflake model.
- Our model have parameters for controlling the amount of microscopic self-shadowing, and therefore the perceived color of the medium due to multiple scattering, independently of the attenuation of the medium. This was not possible in the standard microflake model, in which the amount of multiple scattering was entirely determined by the attenuation coefficient and the single scattering phase function.

These additional degrees of freedom allow for more accurate volume downsampling as shown in Fig. 12.2.

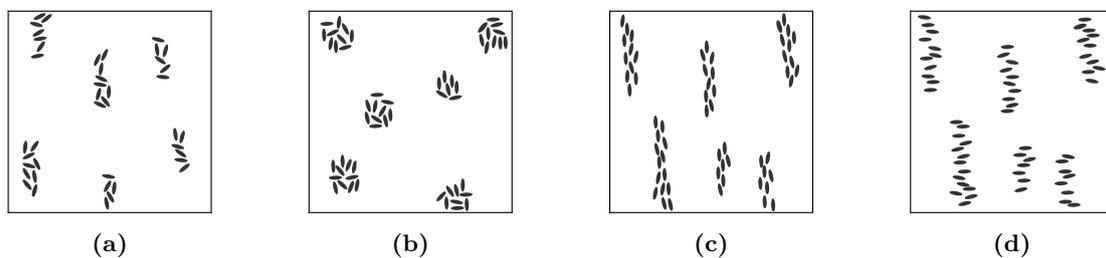


Figure 12.1: Our model can be used for rendering participating media with correlated microflakes at the microscopic scale (**a,b,c,d**). The spatial correlations of microflakes are independent of microflake normal distributions in the model, and can lead to anisotropic attenuation even if the microflake normal distribution is isotropic (**a**).

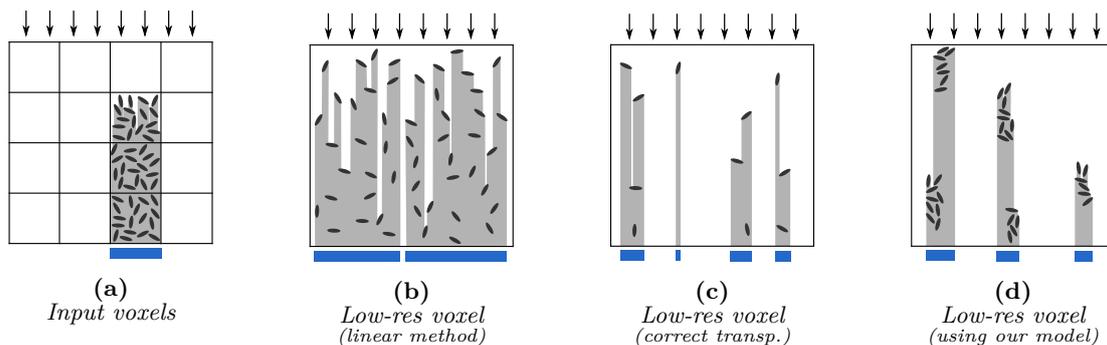


Figure 12.2: In the case of a highly heterogeneous volume (a), naïve downsampling algorithms can lead to inaccurate results (b-c). (b): Linear pre-filtering densities results in incorrect opacity (blue lines) in low-resolution voxels. (c): A correct opacity can be obtained by decreasing the density in low-resolution voxels, but this also decreases the probability of local self-shadowing and leads to overly bright LoDs. In the standard microflake model, the amount of multiple scattering entirely depends on the attenuation of the medium. (d): Our microscopic self-shadowing model can represent media with correlated microflakes and it preserves both anisotropic opacity and self-shadowing effects.

12.1 The microscopic self-shadowing model

Our model builds on the microflake model [JAM*10a]. The main formulas of this model have been given in Sec. 5.3.2.

12.1.1 Notations

In this chapter and the following ones, we use often use spherical coordinates and we call $\phi \in (0, 2\pi)$ the azimuth angle and $\theta \in (0, \pi)$ the polar angle, so that a normalized vector ω writes

$$\omega = \begin{bmatrix} \cos(\phi) \sin(\theta) \\ \sin(\phi) \sin(\theta) \\ \cos(\theta) \end{bmatrix}.$$

The symbols we use can be found in the following table:

| Symbol | Meaning | Unit |
|--------------------------------------|--|-----------|
| $D(\omega)$ | Microflake normal distribution | st^{-1} |
| $A(\omega)$ | Microscopic self-shadowing function | 1 |
| ρ | Area of microflake per unit volume | m^{-1} |
| σ_t | Attenuation coefficient | m^{-1} |
| $\sigma_s, \sigma_{ss}, \sigma_{ms}$ | Scattering coefficients (wavelength dependent) | m^{-1} |
| f, f_{ss}, f_{ms} | Normalized phase functions | st^{-1} |
| $\alpha, \alpha_{ss}, \alpha_{ms}$ | Albedos (wavelength dependent) | 1 |
| $\langle \cdot \rangle$ | Clamped dot product | - |
| $\chi_+(x)$ | Indicator function of positive reals | - |

12.1.2 Microscopic self-shadowing function (A)

Let's consider a volume whose microflakes are spatially correlated at a microscopic scale, as shown in Fig. 12.1. We characterize microflake correlations with the probability that a microflake is shadowed or masked by a neighboring microflake, in a given direction. We introduce a dimensionless scalar function

A on the sphere such that $1 - A(\omega)$ gives the probability of shadowing by other neighboring microflakes in the direction ω . A satisfies:

$$\forall \omega \in S^2, \quad 0 < A(\omega) \leq 1 \quad \text{and} \quad A(\omega) = A(-\omega). \quad (12.1.1)$$

The case $A(\omega) = 1$ corresponds to the standard microflake model in which microflakes are well separated at a microscopic scale and not correlated [JAM*10a]. Given this shadowing behavior at a microscopic scale, we derive new expressions for attenuation coefficients, scattering coefficients and phase functions.

12.1.3 Attenuation coefficient (σ_t)

The attenuation coefficient is given by the standard microflake model times the self-shadowing function A characterizing microflake correlations:

$$\sigma_t(\omega) = A(\omega)\rho \int D(\omega_m)\langle \omega \cdot \omega_m \rangle d\omega_m. \quad (12.1.2)$$

12.1.4 Single scattering coefficient (σ_{ss})

We consider the amount of *local single scattering* $\sigma_{ss}(\omega)$ at microscopic scale, from an incoming direction ω . Single scattering occurs when microflakes are both unmasked and unshadowed:

$$\sigma_{ss}(\omega) = \rho\alpha_{ss}(\lambda)A(\omega) \int A(\omega')D(\omega_m)\langle \omega \cdot \omega_m \rangle d\omega_m \quad (12.1.3)$$

where $\omega' = 2\omega_m(\omega_m \cdot \omega) - \omega$ is the reflected direction given an input direction ω and a microflake normal ω_m . This can be written as an integral over outgoing directions, using the Jacobian of the transformation from normals to specular reflections provided by Walter et al. [WMLT07]:

$$\sigma_{ss}(\omega) = \frac{\rho\alpha_{ss}(\lambda)A(\omega)}{4} \int A(\omega')D(\omega_h) d\omega'. \quad (12.1.4)$$

12.1.5 The single scattering phase function (f_{ss})

Microscopic self-shadowing also impacts the *single scattering phase function* f_{ss} , which is the standard phase function attenuated in some directions due to microscopic shadowing (and normalized):

$$\begin{aligned} f_{ss}(\omega \rightarrow \omega') &= \frac{A(\omega')D(\omega_h)}{\int A(\omega'')D((\omega'' + \omega)/\|\omega'' + \omega\|) d\omega''} \\ &= \frac{\rho\alpha_{ss}(\lambda)A(\omega)A(\omega')D(\omega_h)}{4\sigma_{ss}(\omega)}. \end{aligned} \quad (12.1.5)$$

Again, when $A = 1$, this reduces to the standard microflake specular phase function. It is easy to check that our model satisfies Helmholtz's reciprocity principle by construction:

$$\sigma_{ss}(\omega)f_{ss}(\omega \rightarrow \omega') = \frac{\rho\alpha_{ss}(\lambda)A(\omega)A(\omega')D(\omega_h)}{4}$$

which is symmetric in ω and ω' .

12.1.6 Microscopic multiple scattering

We want our model to take into account microscopic multiple scattering so that it satisfies the *white furnace test*, meaning that all incoming light should be scattered in the case of microflakes with albedo 1, for any self-shadowing function A .

Ideally, our model should have one scattering coefficient and one phase function for each number of local scattering events. Unfortunately, their expressions involve spherical convolutions of the microflake phase function, for which there is no analytic expressions in general. In our work, we chose to use a single multiple scattering coefficient $\sigma_{ms}(\omega)$ and a single multiple scattering albedo $\alpha_{ms}(\lambda)$ that approximate contributions of all scattering orders except single scattering. The multiple scattering coefficient is given by the amount of light that is scattered and then masked locally, times the multiple scattering albedo:

$$\begin{aligned}\sigma_{ms}(\omega) &= \alpha_{ms}(\lambda)\rho A(\omega) \int (1 - A(\omega')) D(\omega_m) \langle \omega \cdot \omega_m \rangle d\omega_m \\ &= \alpha_{ms}(\lambda) \left(\sigma_t(\omega) - \frac{\sigma_{ss}(\omega)}{\alpha_{ss}(\lambda)} \right).\end{aligned}\quad (12.1.6)$$

The multiple scattering albedo $\alpha_{ms}(\lambda)$ is a parameter of our model. We assume that multiple scattering is roughly diffuse, and we use the multiple scattering phase function

$$f_{ms}(\omega \rightarrow \omega') = f_{ms}(\omega') = \frac{\sigma_{ms}(\omega')}{\int \sigma_{ms}(\omega'') d\omega''}.\quad (12.1.7)$$

It satisfies reciprocity since

$$\sigma_{ms}(\omega) f_{ms}(\omega \rightarrow \omega') = \frac{\sigma_{ms}(\omega)\sigma_{ms}(\omega')}{\int \sigma_{ms}(\omega'') d\omega''}$$

which is symmetric in ω and ω' . This phase function scatters light in directions in which there is a lot of microscopic shadowing. Also note that in the case $\alpha_{ss}(\lambda) = \alpha_{ms}(\lambda) = 1$, we have

$$\sigma_{ss}(\omega) + \sigma_{ms}(\omega) = \sigma_{ss}(\omega) + \alpha_{ms}(\lambda) \left(\sigma_t(\omega) - \frac{\sigma_{ss}(\omega)}{\alpha_{ss}(\lambda)} \right) = \sigma_t(\omega)$$

meaning that our model preserves energy when microflakes do not absorb light (*i.e.* the model passes the white furnace test).

12.2 Simplified model with isotropic self-shadowing

Our model greatly simplifies when the self-shadowing function is isotropic, *i.e.* when $A(\omega) = A, \forall \omega$. As discussed in Sec. 13.3, this simplified model is easier to implement, it requires less parameters at rendering and it has a sufficient accuracy in practice. In the case of isotropic self-shadowing, the expressions of our model reduce to

$$\sigma_t(\omega) = A\rho \int D(\omega_m) \langle \omega \cdot \omega_m \rangle d\omega_m \quad (12.2.1)$$

$$\sigma_{ss}(\omega) = \alpha_{ss}(\lambda) A \sigma_t(\omega) \quad (12.2.2)$$

$$f_{ss}(\omega \rightarrow \omega') = \frac{\rho \alpha_{ss}(\lambda) A^2 D(\omega_h)}{4\sigma_{ss}(\omega)} \quad (12.2.3)$$

$$\sigma_{ms}(\omega) = \alpha_{ms}(\lambda) \sigma_t(\omega) (1 - A) \quad (12.2.4)$$

$$f_{ms}(\omega') = \frac{\sigma_{ms}(\omega')}{\int \sigma_{ms}(\omega'') d\omega''} = \frac{\sigma_t(\omega')}{\int \sigma_t(\omega'') d\omega''}.\quad (12.2.5)$$

Note that $\sigma_t(\omega)$ is equal to A times the attenuation coefficient of the standard microflake model, and f_{ss} is exactly the specular phase function of the standard microflake model.

12.3 Summary of the chapter

We extended the microflake model with new parameters that are useful for downsampling volumes. We introduced a self-shadowing function A , where $1 - A(\omega)$ represents the probability of shadowing and masking by neighboring microflakes at a microscopic scale. Self-shadowing impacts the attenuation

coefficient σ_t , the single scattering coefficient σ_{ss} and the associated phase function f_{ss} . For energy conservation, we introduce a local multiple scattering coefficient σ_{ms} and its associated phase function f_{ms} . Rendering with this model requires the evaluation of these functions ($\sigma_t, \sigma_{ss}, f_{ss}, \sigma_{ms}, f_{ms}$) and sampling procedures for f_{ss} and f_{ms} .

In practice, Eq. 12.1.2, 12.1.3 and 12.1.7 rely on spherical integrals of the microflakes distribution D with the shadowing function A , meaning that for most functions D and A , no closed-form expressions can be found. In the next chapter, we present appropriate functions D and A that lead to closed-form expressions, allowing efficient implementations of our self-shadowing model.

12.4 Résumé du chapitre

Nous avons étendu le modèle de volume de microflakes avec de nouveaux paramètres qui sont utiles pour le préfiltrage des volumes. Nous avons pour cela introduit une fonction d'ombrage local A , où $1 - A(\omega)$ représente la probabilité d'ombrage et de masquage par des microflakes voisins à une échelle microscopique. L'ombrage local affecte le coefficient d'atténuation σ_t , le coefficient de diffusion σ_{ss} et la fonction de phase associée f_{ss} . Pour que notre modèle conserve l'énergie lumineuse localement, nous prenons en compte les diffusions multiples locales en introduisant un coefficient de diffusion locale σ_{ms} et une fonction de phase de diffusion f_{ms} . Calculer le rendu de volumes avec ce modèle nécessite d'évaluer ces fonctions ($\sigma_t, \sigma_{ss}, f_{ss}, \sigma_{ms}, f_{ms}$) et de trouver des procédures pour échantillonner f_{ss} et f_{ms} .

En pratique, les équations 12.1.2, 12.1.3 et 12.1.7 contiennent des intégrales dans le domaine sphérique des distributions de microflakes D et des fonctions d'ombrage A . Ces intégrales n'ont pas de formes closes pour la plupart des fonctions D et A . Dans le prochain chapitre, nous proposons des fonctions D et A bien choisies qui permettent d'obtenir des formes closes, afin de pouvoir implémenter efficacement notre modèle.

Chapter 13

Implementing the microscopic self-shadowing model

The self-shadowing model involves spherical convolutions which do not have closed-form expressions for arbitrary microflake distributions D and shadowing functions A . The microflake normal distribution D is defined in the space of microflake normals, while A is defined in the space of incoming or outgoing directions. Because of this, it is very difficult to find closed-form expressions of integrals such as the one equation 12.1.3. For instance, we could not find solutions using the commonly used SGGX distribution, the Gaussian fiber distribution [ZJMB11], Spherical Gaussians [TS06] or Anisotropic Spherical Gaussians [XSD*13] despite their interesting mathematical properties.

Fortunately, we found that using *trigonometric distributions* (Sec. 13.2.2) and a carefully chosen shadowing function A (Sec. 13.2.1), we can derive several closed-form expressions and implement efficiently our model (Sec. 13.2). We also propose an implementation of the simplified model (Sec. 12.2) based on SGGX distributions [HDCD15] in Sec. 13.3. In the next chapter, we introduce algorithms that rely on these models for downsampling and rendering microflake volumes.

This chapter contains long derivations and technical details. We provide a brief summary of our main results (Sec. 13.1), so that the reader may skip the derivations and go directly to the next chapter.

13.1 Main results

13.1.1 Choosing the self-shadowing function A

We first introduce a convenient anisotropic self-shadowing function A :

$$A(\omega) = \omega^T S \omega \tag{13.1.1}$$

with S a symmetric positive definite matrix encoding anisotropy as in SGGX distributions [HDCD15]. This function can be seen as the *square* projected area of an ellipsoid. As we want $0 < A(\omega) \leq 1$, we ensure that S has eigenvalues smaller than or equal to 1. We found that this representation is simple and flexible enough for encoding smooth directional changes of the self-shadowing probability, and it can be stored with only 6 parameters. More importantly, its simplicity allows for deriving closed-form expressions of spherical integrals.

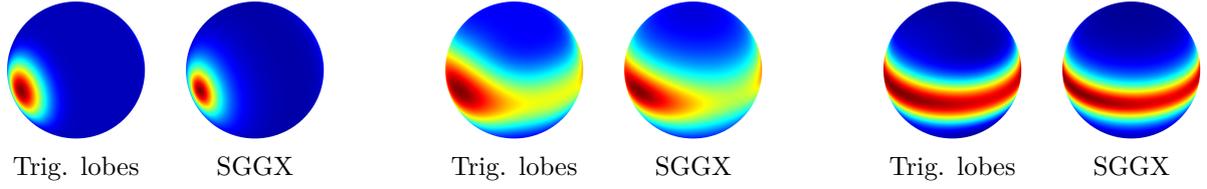


Figure 13.1: Trigonometric lobes (Eq. 13.1.2) can be used for representing foliage-like (left) and fiber-like (right) microflake distributions, similarly to SGGX lobes.

13.1.2 Using trigonometric lobes for the microflake normal distribution

Using self-shadowing functions as described by Eq. 13.1.1, we found closed-form expressions for our model using the following microflake normal distributions:

$$D_{\cos}(\omega, \xi, n) = \frac{\cos^{2n}(\omega, \xi)}{N_{\cos}(n)} = \frac{(\omega \cdot \xi)^{2n}}{N_{\cos}(n)}$$

$$D_{\sin}(\omega, \xi, n) = \frac{\sin^{2n}(\omega, \xi)}{N_{\sin}(n)} = \frac{(1 - (\omega \cdot \xi)^2)^n}{N_{\sin}(n)}$$

with $n \in \mathbb{N}$ and $N_{\cos}(n)$ and $N_{\sin}(n)$ the normalization factors whose closed-form expressions can be found in Sec. 13.2.3. These distributions will be referred to as *trigonometric lobes*. Any linear combination of such lobes still leads to closed-form expressions in our model (by the linearity of the integrals with respect to the microflake distribution). These lobes can represent the same kinds of distributions as the SGGX distribution (Fig. 13.1), using for instance

$$D(\omega) = w_c D_{\cos}(\omega, \xi_c, n_c) + w_s D_{\sin}(\omega, \xi_s, n_s) + \frac{w_{iso}}{4\pi} \quad (13.1.2)$$

with w_c, w_s, w_{iso} positive weights such that $w_c + w_s + w_{iso} = 1$.

13.1.3 Attenuation coefficients for \cos^{2n} and \sin^{2n} lobes

In Sec. 13.2.5, we derive a general closed-form expression for the attenuation coefficient σ_t^{\cos} , but this expression involves costly Gauss hypergeometric functions ${}_2F_1$ and cannot be evaluated efficiently at rendering. However, for each particular value n , we have found efficient closed-form expressions of the form:

$$\sigma_t^{\cos}(\omega) = \rho A(\omega) \begin{bmatrix} 1 \\ (\omega \cdot \xi_D)^2 \\ \dots \\ (\omega \cdot \xi_D)^{2n} \end{bmatrix} \cdot C_1 \quad (13.1.3)$$

with C_1 a $n + 1$ vector of coefficients. We computed coefficients for each n between 1 and 20, using symbolic integration [MGH*05]. Note that for each n , these expressions are exact. We have found similar expressions for \sin^{2n} lobes (only coefficients C_1 change).

13.1.4 Single scattering coefficients

Similarly, we have found closed-form expressions for σ_{ss} for each n :

$$\sigma_{ss}^{\cos}(\omega) = \alpha_{ss}(\lambda) \rho A(\omega) \begin{bmatrix} 1 \\ (\omega \cdot \xi_D)^2 \\ \dots \\ (\omega \cdot \xi_D)^{2n} \end{bmatrix} \cdot C_2 \cdot \begin{bmatrix} \omega^T S_A \omega \\ \xi_D^T S_A \xi_D \\ \text{trace}(A) \end{bmatrix}$$

$$+ \alpha_{ss}(\lambda) \rho A(\omega) (\omega^T S_A \xi_D) \begin{bmatrix} (\omega \cdot \xi_D) \\ \dots \\ (\omega \cdot \xi_D)^{2n-1} \end{bmatrix} \cdot C_3 \quad (13.1.4)$$

with C_2 a $(n + 1) \times 3$ matrix of coefficients and C_3 a vector of coefficients of length n . We found similar expression for \sin^{2n} lobes.

13.1.5 Evaluating and sampling f_{ss} and f_{ms}

Details concerning phase functions can be found in Sec. 13.2.10 and Sec. 13.2.11. The evaluation of our multiple scattering phase function f_{ms} (Eq. 12.1.7) involves the integral of the multiple scattering coefficient:

$$\int \sigma_{ms}(\omega) d\omega = \alpha_{ms}(\lambda) \int \sigma_t(\omega) d\omega - \frac{\alpha_{ms}(\lambda)}{\alpha_{ss}(\lambda)} \int \sigma_{ss}(\omega) d\omega.$$

We provide closed-form expressions for $\int \sigma_t(\omega)$ and $\int \sigma_{ss}(\omega)$ (Sec. 13.2.7). We also provide sampling procedures for visible normal distributions [HDCD15] for the trigonometric lobes. We rely on rejection sampling for sampling exactly f_{ss} and f_{ms} . Our sampling procedures are very efficient (*i.e.* samples are almost always accepted) when the shadowing function A has few anisotropic variations, which is often the case in practice. Our procedures are less efficient when A is highly anisotropic.

13.1.6 Limitations

Compared to SGGX distributions, implementing our model with the distribution proposed in Eq. 13.1.2 is about 2 to 3 times more expensive for evaluating $\sigma_t(\omega)$, and about 25 to 60 times more expensive for sampling the phase functions, depending on n_c , n_s , and $A(\omega)$. Despite these additional costs compared to the standard microflake model, rendering our LODs is much faster than rendering high-resolution volumes. In Sec. 13.3, we propose an implementation of our simplified self-shadowing model that is more efficient because it relies on SGGX distributions for σ_t and f_{ss} .

Another limitation of the implementation proposed here is that unlike SGGX distributions, interpolating trigonometric distributions cannot be done by linearly interpolating the parameters of the trigonometric lobes. This means that trilinear spatial interpolation at rendering would require 8 evaluations of σ_t .

13.1.7 Implementing the simplified self-shadowing model using the SGGX distribution

In our simplified self-shadowing model, σ_t and σ_{ss} almost reduce to the standard microflake model, for which the SGGX distribution allows for efficient rendering. For this reason, we propose an implementation of the simplified model based on the SGGX distribution (Sec. 13.3). However, using SGGX distributions, there is no closed-form expression for $\int \sigma_{ms}(\omega) d\omega$, which is involved in the multiple scattering phase function. Assuming normalized SGGX matrices, meaning that the highest eigenvalue is 1, the other two eigenvalues belong to the interval $(0, 1]$. We discretized this interval and precomputed $\int \sigma_t(\omega'') d\omega''$ for each pair of eigenvalues using numerical integration (Appendix D).

In the simplified self-shadowing model, sampling the multiple scattering phase function f_{ms} is equivalent to importance sampling σ_t , *i.e.* the projected area of the SGGX ellipsoid given by $\sqrt{\omega^T S \omega}$. In the supplemental material, we provide the details of a sampling procedure for f_{ms} , using the fact that $\omega^T S \omega$ is close enough to $\sqrt{\omega^T S \omega}$ to allow efficient rejection sampling.

13.2 Implementing the self-shadowing model with trigonometric lobes

In this section, we derive closed-form expressions and sampling procedures for implementing our self-shadowing model, using an anisotropic self-shadowing function (Sec. 13.2.1) and microflake normal distributions based on trigonometric lobes (Sec. 13.2.2). The derivations include:

- Normalization factors for trigonometric lobes (Sec.13.2.3).
- Expressions for the attenuation coefficient σ_t (Sec. 13.2.5).
- Expressions for the single scattering coefficient σ_{ss} (Sec. 13.2.6).
- Expressions for $\int \sigma_t$ and $\int \sigma_{ss}$, involved in the multiple scattering albedo σ_{ms} (Sec. 13.2.7).
- Sampling procedures for the single scattering phase function f_{ss} (Sec. 13.2.10 and 13.2.11).
- Sampling procedures for the multiple scattering phase function f_{ms} (Sec. 13.2.12).

13.2.1 Choice of a self-shadowing function

We introduce an anisotropic self-shadowing function A for which we can find closed-form expressions for the spherical convolutions and integrals involved in our self-shadowing model (Eq. 12.1.3, 12.1.7). This anisotropic self-shadowing function writes

$$A(\omega) = a_1(\xi_{A_1} \cdot \omega)^2 + a_2(\xi_{A_2} \cdot \omega)^2 + a_3(\xi_{A_3} \cdot \omega)^2. \quad (13.2.1)$$

It can also be written as the square projected area of an ellipsoid in direction ω [HDCD15] with axes ξ_{A_i} :

$$A(\omega) = \omega^T \begin{bmatrix} \xi_{A_1} & \xi_{A_2} & \xi_{A_3} \end{bmatrix} \begin{bmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{bmatrix} \begin{bmatrix} \xi_{A_1}^T \\ \xi_{A_2}^T \\ \xi_{A_3}^T \end{bmatrix} \omega = \omega^T S_A \omega. \quad (13.2.2)$$

13.2.2 Trigonometric lobes: $D_{\cos}(\omega, \xi, n)$ and $D_{\sin}(\omega, \xi, n)$

We use distributions based on trigonometric lobes:

$$D_{\cos}(\omega, \xi, n) = \frac{\cos^{2n}(\omega, \xi)}{N_{\cos}(n)} = \frac{(\omega \cdot \xi)^{2n}}{N_{\cos}(n)}$$

$$D_{\sin}(\omega, \xi, n) = \frac{\sin^{2n}(\omega, \xi)}{N_{\sin}(n)} = \frac{(1 - (\omega \cdot \xi)^2)^n}{N_{\sin}(n)}$$

$$D_{\text{iso}}(\omega) = \frac{1}{4\pi}$$

with $n \in \mathbb{N}$ and $N_{\cos}(n)$ and $N_{\sin}(n)$ the normalization factors.

13.2.3 Normalization factors

For the cosine lobe, the normalization factor writes

$$N_{\cos}(n) = \int (m \cdot \xi_D)^{2n} dm = \frac{4\pi}{2n+1}.$$

The normalization factor for the sine lobe writes

$$N_{\sin}(n) = \int \left(1 - (m \cdot \xi_D)^2\right)^n dm = \frac{2\pi^{\frac{3}{2}}\Gamma(1+n)}{\Gamma\left(\frac{3}{2}+n\right)}.$$

In our implementation, we precomputed the normalization factors for the sine lobe for $n \in [1..20]$.

13.2.4 Proof that distributions $D_{\sin}(\omega, \xi, n)$ can be written as a sum of distributions $D_{\cos}(\omega, \xi, n)$

Sine lobes $D_{\sin}(\omega, \xi, n)$ have an interesting property: they can be written as a sum of $2n$ cosine lobes:

$$D_{\sin}(\omega, \xi, n) = \frac{1}{2n} \sum_{p=0}^{2n-1} D_{\cos}(\omega, \xi_p, n) \quad (13.2.3)$$

with ξ_p being $2n$ evenly spaced axes in the plane orthogonal to ξ . For example, a $D_{\sin}(\omega, \xi, 5)$ distribution is equivalent to a sum of 10 $D_{\cos}(\omega, \xi_p, 5)$ distributions:



Without loss of generality, we derive the proof in spherical coordinates in the case $\xi_D^{\sin} = [0, 0, 1]^T$. The sum of cosine lobes writes

$$\frac{1}{2n} \frac{2n+1}{4\pi} \sum_{p=0}^{2n-1} \left(\begin{bmatrix} \cos\left(\frac{p\pi}{2n}\right) \\ \sin\left(\frac{p\pi}{2n}\right) \\ 0 \end{bmatrix} \cdot \begin{bmatrix} \cos(\phi) \sin(\theta) \\ \sin(\phi) \sin(\theta) \\ \cos(\theta) \end{bmatrix} \right)^{2n} \quad (13.2.4)$$

$$= \frac{2n+1}{8n\pi} \sum_{p=0}^{2n-1} \left(\cos\left(\frac{p\pi}{2n}\right) \cos(\phi) \sin(\theta) + \sin\left(\frac{p\pi}{2n}\right) \sin(\phi) \sin(\theta) \right)^{2n} \quad (13.2.5)$$

$$= \frac{2n+1}{8n\pi} \sin(\theta)^{2n} \sum_{p=0}^{2n-1} \left(\cos\left(\frac{p\pi}{2n}\right) \cos(\phi) + \sin\left(\frac{p\pi}{2n}\right) \sin(\phi) \right)^{2n} \quad (13.2.6)$$

$$= \frac{2n+1}{8n\pi} \sin(\theta)^{2n} \sum_{p=0}^{2n-1} \cos\left(\frac{p\pi}{2n} - \phi\right)^{2n}. \quad (13.2.7)$$

We can expand $\cos(x)^{2n}$ [Weil17] as follows:

$$\cos(x)^{2n} = \frac{1}{2^{2n}} \binom{2n}{n} + \frac{1}{2^{2n-1}} \sum_{k=0}^{n-1} \binom{2n}{k} \cos(2(n-k)x)$$

so we can write

$$\sum_{p=0}^{2n-1} \cos\left(\frac{p\pi}{2n} - \phi\right)^{2n} = \sum_{p=0}^{2n-1} \left(\frac{1}{2^{2n}} \binom{2n}{n} + \frac{1}{2^{2n-1}} \sum_{k=0}^{n-1} \binom{2n}{k} \cos\left(2(n-k)\left(\frac{p\pi}{2n} - \phi\right)\right) \right).$$

We can check that terms involving ϕ cancel to 0:

$$\begin{aligned} \sum_{p=0}^{2n-1} \sum_{k=0}^{n-1} \binom{2n}{k} \cos\left(2(n-k)\left(\frac{p\pi}{2n} - \phi\right)\right) &= \sum_{k=0}^{n-1} \binom{2n}{k} \sum_{p=0}^{2n-1} \cos\left(2(n-k)\left(\frac{p\pi}{2n} - \phi\right)\right) \\ &= \sum_{k=0}^{n-1} \binom{2n}{k} \Re \left(\sum_{p=0}^{2n-1} \exp\left(2i(n-k)\left(\frac{p\pi}{2n} - \phi\right)\right) \right) \\ &= \sum_{k=0}^{n-1} \binom{2n}{k} \Re \left(\exp(-2i(n-k)\phi) \sum_{p=0}^{2n-1} \exp\left(i(n-k)\frac{2\pi}{2n} p\right) \right) \\ &= \sum_{k=0}^{n-1} \binom{2n}{k} \Re \left(\exp(-2i(n-k)\phi) \frac{1 - \exp\left(i(n-k)2\pi\right)}{1 - \exp\left(i(n-k)\frac{2\pi}{2n}\right)} \right) \\ &= 0. \end{aligned}$$

Finally, we obtain

$$\frac{2n+1}{8n\pi} \sum_{p=0}^{2n-1} \left(\begin{bmatrix} \cos\left(\frac{p\pi}{2n}\right) \\ \sin\left(\frac{p\pi}{2n}\right) \\ 0 \end{bmatrix} \cdot \begin{bmatrix} \cos(\phi) \sin(\theta) \\ \sin(\phi) \sin(\theta) \\ \cos(\theta) \end{bmatrix} \right)^{2n} = \frac{2n+1}{8n\pi} \sin(\theta)^{2n} \sum_{p=0}^{2n-1} \frac{1}{2^{2n}} \binom{2n}{n} \quad (13.2.8)$$

$$= \sin(\theta)^{2n} \frac{2n+1}{8n\pi} \frac{2n}{2^{2n}} \binom{2n}{n} \quad (13.2.9)$$

$$= \sin(\theta)^{2n} \frac{2n+1}{\pi 4^{n+1}} \frac{(2n)!}{2n!} \quad (13.2.10)$$

$$= \sin(\theta)^{2n} \frac{\sqrt{\pi}(2n+2)!}{4^{n+1}(n+1)!} \frac{(n+1)}{(2n+2)\pi^{3/2}n!} \quad (13.2.11)$$

$$= \sin(\theta)^{2n} \frac{\Gamma(n+3/2)}{2\pi^{3/2}n!} \quad (13.2.12)$$

$$= D_{\sin}(\omega, \xi_D^{\sin}, n). \quad (13.2.13)$$

13.2.5 Closed-form expression of the attenuation coefficient $\sigma_t(\omega)$

Derivation of general closed-form expressions for $\sigma_t(\omega)$

A general closed-form expression can be found for $\sigma_t(\omega)$ but this expression is not convenient for evaluation because it involves costly evaluations of Gauss hypergeometric functions ${}_2F_1$. For information, we provide a derivation of this expression for the cosine lobe (Eq. C.18) in Appendix C. For efficient implementation, we have precomputed simpler closed-form expressions for each $n \in [1..20]$ as explained in Section 13.2.5. Note that a general closed-form expression could also be derived for the sine lobe, using the fact that a sine lobe can be written as a sum of cosine lobes (Sec. 13.2.4).

Pre-computing simple closed-form expressions for particular values n

Using symbolic integration [MGH*05], we have found that for each particular n , $\sigma_t^{\cos}(\omega)$ can be written

$$\sigma_t^{\cos}(\omega) = \rho A(\omega) \begin{bmatrix} 1 \\ (\omega \cdot \xi_D)^2 \\ (\omega \cdot \xi_D)^4 \\ \dots \\ (\omega \cdot \xi_D)^{2n} \end{bmatrix} \cdot C_1$$

with C_1 a $n+1$ vector of coefficients. The evaluation cost is linear in n and this expression is very efficient when n is small. In practice we precomputed coefficients from $n = 1$ to 20 (corresponding to a \cos^{40} lobe). We have found similar expressions for sine lobes. These coefficients can be found in our code, in the supplemental material of our publication [LN18] (files *matrixcoslobe.h* and *matrixsinlobe.h*).

13.2.6 Single scattering coefficients σ_{ss} for D_{\cos} and D_{\sin} microflake distributions

The general closed-form expressions for σ_{ss} are even more complex than the expressions for σ_t (Eq. C.18) and they have no practical interest for rendering. Like we did for σ_t , we precomputed simpler and exact closed-form solutions for each $n = 1$ to 20. We have found using symbolic integration that

$$\begin{aligned}
\int (\xi_A \cdot \omega')^2 D_{\cos}(m, \xi_D, n) \langle m \cdot \omega \rangle dm &= \begin{bmatrix} 1 \\ (\omega \cdot \xi_D)^2 \\ (\omega \cdot \xi_D)^4 \\ \dots \\ (\omega \cdot \xi_D)^{2n} \end{bmatrix} \cdot C_2 \cdot \begin{bmatrix} (\omega \cdot \xi_A)^2 \\ (\xi_A \cdot \xi_D)^2 \\ 1 \end{bmatrix} \\
&+ (\xi_A \cdot \xi_D)(\xi_A \cdot \omega) \begin{bmatrix} (\omega \cdot \xi_D) \\ (\omega \cdot \xi_D)^3 \\ \dots \\ (\omega \cdot \xi_D)^{2n-1} \end{bmatrix} \cdot C_3 \quad (13.2.14)
\end{aligned}$$

with $\omega' = -\omega + 2m(\omega \cdot m)$, C_2 a $(n+1) \times 3$ matrix of coefficients and C_3 a vector of coefficients of length n . Given Equation 13.2.1, we sum the contributions of the three shadowing axes and we obtain

$$\begin{aligned}
\int A(\omega') D_{\cos}(m, \xi_D, n) \langle m \cdot \omega \rangle dm &= \begin{bmatrix} 1 \\ (\omega \cdot \xi_D)^2 \\ (\omega \cdot \xi_D)^4 \\ \dots \\ (\omega \cdot \xi_D)^{2n} \end{bmatrix} \cdot C_2 \cdot \begin{bmatrix} a_1(\omega \cdot \xi_{A1})^2 + a_2(\omega \cdot \xi_{A2})^2 + a_3(\omega \cdot \xi_{A3})^2 \\ a_1(\xi_{A1} \cdot \xi_D)^2 + a_2(\xi_{A2} \cdot \xi_D)^2 + a_3(\xi_{A3} \cdot \xi_D)^2 \\ a_1 + a_2 + a_3 \end{bmatrix} \\
&+ (a_1(\xi_{A1} \cdot \xi_D)(\xi_{A1} \cdot \omega) + a_2(\xi_{A2} \cdot \xi_D)(\xi_{A2} \cdot \omega) + a_3(\xi_{A3} \cdot \xi_D)(\xi_{A3} \cdot \omega)) \begin{bmatrix} (\omega \cdot \xi_D) \\ (\omega \cdot \xi_D)^3 \\ \dots \\ (\omega \cdot \xi_D)^{2n-1} \end{bmatrix} \cdot C_3. \quad (13.2.15)
\end{aligned}$$

We can write:

$$\begin{aligned}
a_1(\omega \cdot \xi_{A1})^2 + a_2(\omega \cdot \xi_{A2})^2 + a_3(\omega \cdot \xi_{A3})^2 &= \omega^T S_A \omega \\
a_1(\xi_{A1} \cdot \xi_D)^2 + a_2(\xi_{A2} \cdot \xi_D)^2 + a_3(\xi_{A3} \cdot \xi_D)^2 &= \xi_D^T S_A \xi_D \\
a_1(\xi_{A1} \cdot \xi_D)(\xi_{A1} \cdot \omega) + a_2(\xi_{A2} \cdot \xi_D)(\xi_{A2} \cdot \omega) + a_3(\xi_{A3} \cdot \xi_D)(\xi_{A3} \cdot \omega) &= \omega^T S_A \xi_D \\
a_1 + a_2 + a_3 &= \text{tr}(A).
\end{aligned}$$

Finally we obtain, for a cosine lobe D_{\cos} ,

$$\begin{aligned}
\sigma_{ss}(\omega) = \alpha_{ss}(\lambda) \rho A(\omega) \begin{bmatrix} 1 \\ (\omega \cdot \xi_D)^2 \\ (\omega \cdot \xi_D)^4 \\ \dots \\ (\omega \cdot \xi_D)^{2n} \end{bmatrix} \cdot C_2 \cdot \begin{bmatrix} \omega^T S_A \omega \\ \xi_D^T S_A \xi_D \\ \text{trace}(A) \end{bmatrix} + \alpha_{ss}(\lambda) \rho A(\omega) (\omega^T S_A \xi_D) \begin{bmatrix} (\omega \cdot \xi_D) \\ (\omega \cdot \xi_D)^3 \\ \dots \\ (\omega \cdot \xi_D)^{2n-1} \end{bmatrix} \cdot C_3. \quad (13.2.16)
\end{aligned}$$

Expressions in the case of sine lobes D_{\sin} are similar, only coefficients in C_2 and C_3 are different. All the coefficients can be found in our code in the files *matrixcoslobe.h* and *matrixsinlobe.h* [LN18].

13.2.7 Closed-form expression for $\int \sigma_t$ and $\int \sigma_{ss}$

Integrals $\int \sigma_t(\omega) d\omega$ and $\int \sigma_{ss}(\omega) d\omega$ are involved in our multiple scattering phase function (Eq. 12.1.7). We provide here the derivations of their closed-form expressions.

Closed-form expression of $\int \sigma_t(\omega)$ for cosine lobes $D_{\cos}(m, \xi_D, n)$

We want to find a closed-form expression for a cosine distribution:

$$\int \sigma_t(\omega) d\omega = \rho \int A(\omega) \int D_{\cos}(m, \xi_D, n) \langle m \cdot \omega \rangle dm d\omega. \quad (13.2.17)$$

We first invert the integrals:

$$\int A(\omega) \int D_{\cos}(m, \xi_D, n) \langle m \cdot \omega \rangle dm d\omega = \int D_{\cos}(m, \xi_D, n) \int A(\omega) \langle m \cdot \omega \rangle d\omega dm. \quad (13.2.18)$$

We can decompose our shadowing function in three parts using equation 13.2.1:

$$\int A(\omega) \langle m \cdot \omega \rangle d\omega = \sum a_i \int (\omega \cdot \xi_{A_i})^2 \langle m \cdot \omega \rangle d\omega. \quad (13.2.19)$$

We choose a convenient parametrization: we take $m = [1, 0, 0]^T$ and $\xi_{A_i} = [\sin(\theta_A), 0, \cos(\theta_A)]^T$, so that we can write

$$\begin{aligned} \int (\omega \cdot \xi_{A_i})^2 \langle m \cdot \omega \rangle d\omega &= \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\pi/2} (\sin(\theta_A) \cos(\phi) \sin(\theta) + \cos(\theta_A) \cos(\theta))^2 \cos(\theta) \sin(\theta) d\phi d\theta \\ &= \frac{\pi}{4} (\cos(\theta_A)^2 + 1) = \frac{\pi}{4} ((\xi_{A_i} \cdot m)^2 + 1). \end{aligned} \quad (13.2.20)$$

We can now substitute this expression in Eq. 13.2.18 and we obtain

$$\int D_{\cos}(m, \xi_D, n) \frac{\pi}{4} ((\xi_{A_i} \cdot m)^2 + 1) dm = \int \frac{2n+1}{4\pi} (\xi_D \cdot m)^{2n} \frac{\pi}{4} ((\xi_{A_i} \cdot m)^2 + 1) dm. \quad (13.2.21)$$

Choosing a parametrization such that $\xi_D = [0, 0, 1]^T$ and $\xi_{A_i} = [\sin(\theta_A), 0, \cos(\theta_A)]^T$, this writes

$$\int_{\phi=0}^{2\pi} \int_{\theta=0}^{\pi} \frac{2n+1}{4\pi} \cos(\theta)^{2n} \frac{\pi}{4} ((\sin(\theta_A) \cos(\phi) \sin(\theta) + \cos(\theta_A) \cos(\theta))^2 + 1) \sin(\theta) dm = \frac{\pi (n (\xi_D \cdot \xi_{A_i})^2 + n + 2)}{4n + 6}. \quad (13.2.22)$$

Finally, we can sum and factor the formula for the three axes, and we obtain

$$\int \sigma_t(\omega) d\omega = \rho a_1 \frac{\pi (n (\xi_D \cdot \xi_{A_1})^2 + n + 2)}{4n + 6} + \rho a_2 \frac{\pi (n (\xi_D \cdot \xi_{A_2})^2 + n + 2)}{4n + 6} + \rho a_3 \frac{\pi (n (\xi_D \cdot \xi_{A_3})^2 + n + 2)}{4n + 6}. \quad (13.2.23)$$

Using Eq. 13.2.2, we can write, for a distribution $D_{\cos}(m, \xi_D, n)$:

$$\boxed{\int \sigma_t(\omega) d\omega = \rho \int A(\omega) \int D_{\cos}(m, \xi_D, n) \langle m \cdot \omega \rangle dm d\omega = \rho \frac{\pi (n(\omega^T S_A \omega) + \text{tr}(S_A)(n + 2))}{4n + 6}.} \quad (13.2.24)$$

Closed-form expression of $\int \sigma_t(\omega)$ for sine lobe $D_{\sin}(m, \xi_D, n)$

The derivation is almost the same as for the $D_{\cos}(m, \xi_D, n)$ in Sec. 13.2.7. We need to compute

$$\int D_{\sin}(m, \xi_D, n) \frac{\pi}{4} ((\xi_{A_i} \cdot m)^2 + 1) dm = \int \frac{2\pi^{\frac{3}{2}} \Gamma(1+n)}{\Gamma(\frac{3}{2}+n)} (1 - (\xi_D \cdot m)^2)^n \frac{\pi}{4} ((\xi_{A_i} \cdot m)^2 + 1) dm. \quad (13.2.25)$$

Choosing the same convenient frame in which $\xi_D = [0, 0, 1]^T$ and $\xi_{A_i} = [\sin(\theta_A), 0, \cos(\theta_A)]^T$, this writes

$$2 \times \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\pi/2} \frac{\pi}{4} ((\sin(\theta_A) \cos(\phi) \sin(\theta) + \cos(\theta_A) \cos(\theta))^2 + 1) \sin(\theta)^{2n+1} dm = \frac{\pi (3n + 4 - n (\xi_D \cdot \xi_{A_i})^2)}{8n + 12}. \quad (13.2.26)$$

Again, we have to sum this for each shadowing axis. For a $D_{\sin}(m, \xi_D, n)$ distribution, we obtain:

$$\boxed{\int \sigma_t(\omega) d\omega = \rho \int A(\omega) \int D_{\sin}(m, \xi_D, n) \langle m \cdot \omega \rangle dm d\omega = \rho \frac{\pi ((3n + 4) \text{tr}(S_A) - n(\omega^T S_A \omega))}{8n + 12}.} \quad (13.2.27)$$

Closed-form expression of $\int \sigma_{ss}(\omega)$ for cosine lobe $D_{\cos}(m, \xi_D, n)$

In the case of a distribution $D_{\cos}(m, \xi_D, n)$, we look for a closed-form expression of the integral

$$\int \sigma_{ss}(\omega) d\omega = \rho\alpha_{ss}(\lambda) \int A(\omega) \int A(\omega') D_{\cos}(m, \xi_D, n) \langle \omega \cdot m \rangle dm d\omega, \quad (13.2.28)$$

with $\omega' = -\omega + 2m(\omega \cdot m)$. We invert integrals and we obtain

$$\int D_{\cos}(m, \xi_D, n) \int A(\omega) A(\omega') \langle \omega \cdot m \rangle d\omega dm. \quad (13.2.29)$$

We expand each shadowing term using Equation 13.2.1 and we obtain 9 terms of the form:

$$\int D_{\cos}(m, \xi_D, n) \int A(\omega) A(\omega') \langle \omega \cdot m \rangle d\omega dm = \sum_{i,j \in \{1,2,3\}} \int D_{\cos}(m, \xi_D, n) a_i a_j \int (\omega \cdot \xi_{A_i})^2 (\omega' \cdot \xi_{A_j})^2 \langle \omega \cdot m \rangle d\omega. \quad (13.2.30)$$

We first consider the case $i = j$, and then the case $i \neq j$.

Case $i = j$

Without loss of generality, we work in the reference frame in which $m = [1, 0, 0]$ and $\xi_{A_i} = [\cos(\phi_{A_i}), \sin(\phi_{A_i}), 0]$. We find

$$\int (\omega \cdot \xi_{A_i})^2 ((-\omega + 2m(\omega \cdot m)) \cdot \xi_{A_i})^2 \langle \omega \cdot m \rangle d\omega \quad (13.2.31)$$

$$= \int_{\phi=-\pi/2}^{\pi/2} \int_{\theta=0}^{\pi} (\cos(\phi_{A_i}) \cos(\phi) + \sin(\phi_{A_i}) \sin(\phi))^2 (\cos(\phi_{A_i}) \cos(\phi) - \sin(\phi_{A_i}) \sin(\phi))^2 \cos(\phi) \sin(\theta)^6 d\omega d\theta \quad (13.2.32)$$

$$= \frac{1}{24} \pi (15(m \cdot \xi_{A_i})^4 - 10(m \cdot \xi_{A_i})^2 + 3). \quad (13.2.33)$$

We integrate this result with $D_{\cos}(m, \xi_D, n)$ (recall that we want a closed-form expression for Equation 13.2.28) and we find

$$\begin{aligned} & \int D_{\cos}(m, \xi_D, n) \frac{1}{24} \pi (15(m \cdot \xi_{A_i})^4 - 10(m \cdot \xi_{A_i})^2 + 3) dm \\ &= \pi \frac{15n(n-1)(\xi_D \cdot \xi_{A_i})^4 - 10n(n-2)(\xi_D \cdot \xi_{A_i})^2 + 3n^2 + 7n + 10}{24n^2 + 96n + 90}. \end{aligned} \quad (13.2.34)$$

Case $i \neq j$

We choose a reference frame such that $m = [0, 0, 1]^T$, $A_i = [\sin(\theta_{A_i}), 0, \cos(\theta_{A_i})]^T$ and

$$A_j = [\cos(\phi_{A_j}) \sin(\theta_{A_j}), \sin(\phi_{A_j}) \sin(\theta_{A_j}), \cos(\theta_{A_j})]^T.$$

We have

$$\xi_{A_i} \perp \xi_{A_j} \Rightarrow \sin(\theta_{A_i}) \cos(\phi_{A_j}) \sin(\theta_{A_j}) + \cos(\theta_{A_j}) \cos(\theta_{A_i}) = 0.$$

We develop and integrate as before, and find

$$\int (\omega \cdot \xi_{A_i})^2 ((-\omega + 2m(\omega \cdot m)) \cdot \xi_{A_j})^2 \langle \omega \cdot m \rangle d\omega = \frac{\pi}{24} (15(m \cdot \xi_{A_i})^2 (m \cdot \xi_{A_j})^2 + (m \cdot \xi_{A_i})^2 + (m \cdot \xi_{A_j})^2 + 1). \quad (13.2.35)$$

Finally, we obtain

$$\begin{aligned} & \int D_{\cos}(m, \xi_D, n) \frac{\pi}{24} (15(m \cdot \xi_{A_i})^2 (m \cdot \xi_{A_j})^2 + (m \cdot \xi_{A_i})^2 + (m \cdot \xi_{A_j})^2 + 1) dm \\ &= \pi \frac{15C_1 C_2 n(n-1) + (n^2 + 10n)(C_1 + C_2) + n^2 + 5n + 10}{24n^2 + 96n + 90} \end{aligned} \quad (13.2.36)$$

with $C_1 = (\xi_{A_i} \cdot \xi_D)^2$ and $C_2 = (\xi_{A_j} \cdot \xi_D)^2$.

Sum of all terms

Given equations 13.2.30, 13.2.34 and 13.2.36, we can write the closed-form expression of $\int \sigma_{ss}(\omega)$ in the case of a cosine lobe $D_{\cos}(m, \xi_D, n)$. After simplification, we obtain:

$$\int \sigma_{ss}(\omega) d\omega = \frac{\pi \rho \alpha_{ss}(\lambda)}{24n^2 + 96n + 90} \left(15n(n-1) (\xi_D^T S_A \xi_D)^2 - 10n(n-2) \xi_D^T S_A S_A \xi_D + 2n(n+10) (\xi_D^T S_A \xi_D \operatorname{tr}(S_A) - \xi_D^T S_A S_A \xi_D) + (3n^2 + 7n + 10) \operatorname{tr}(S_A S_A) + (n^2 + 5n + 10) (\operatorname{tr}(S_A)^2 - \operatorname{tr}(S_A S_A)) \right).$$

(13.2.37)

Closed-form expression of $\int \sigma_{ss}(\omega)$ for sine lobe $D_{\sin}(m, \xi_D, n)$

The derivation is almost the same as for cosine lobes.

Case $i = j$

We have shown (Sec. 13.2.7) that

$$\int (\omega \cdot \xi_{A_i})^2 ((-\omega + 2m(\omega \cdot m)) \cdot \xi_{A_i})^2 \langle \omega \cdot m \rangle d\omega = \frac{\pi}{24} (15(m \cdot \xi_{A_i})^4 - 10(m \cdot \xi_{A_i})^2 + 3) \quad (13.2.38)$$

Then

$$\begin{aligned} \int D_{\sin}(m, \xi_D, n) \frac{\pi}{24} (15(m \cdot \xi_{A_i})^4 - 10(m \cdot \xi_{A_i})^2 + 3) dm \\ = \pi \frac{(45n^2 - 45n)(\xi_D \cdot \xi_{A_i})^4 + (-50n^2 + 10n)(\xi_D \cdot \xi_{A_i})^2 + 29n^2 + 91n + 80}{192n^2 + 768n + 720}. \end{aligned} \quad (13.2.39)$$

Case $i \neq j$

We have found previously (Sec. 13.2.7) that

$$\int (\omega \cdot \xi_{A_i})^2 ((-\omega + 2m(\omega \cdot m)) \cdot \xi_{A_j})^2 \langle \omega \cdot m \rangle d\omega = \frac{1}{24} \pi (15(m \cdot \xi_{A_i})^2 (m \cdot \xi_{A_j})^2 + (m \cdot \xi_{A_i})^2 + (m \cdot \xi_{A_j})^2 + 1) \quad (13.2.40)$$

We obtain

$$\begin{aligned} \int D_{\sin}(m, \xi_D, n) \frac{\pi}{24} (15(m \cdot \xi_{A_i})^2 (m \cdot \xi_{A_j})^2 + (m \cdot \xi_{A_i})^2 + (m \cdot \xi_{A_j})^2 + 1) dm \\ = \pi \frac{45C_1 C_2 n(n-1) - (C_1 + C_2)n(19n + 25) + 31n^2 + 105n + 80}{192n^2 + 768n + 720}. \end{aligned} \quad (13.2.41)$$

with $C_1 = (\xi_{A_1} \cdot \xi_D)^2$ and $C_2 = (\xi_{A_2} \cdot \xi_D)^2$.

Sum of all terms

Again, we sum the terms from cases $i = j$ and cases $i \neq j$, and after simplification we obtain the closed-form expression of $\int \sigma_{ss}$ for sine lobes $D_{\sin}(m, \xi_D, n)$:

$$\int \sigma_{ss}(\omega) d\omega = \frac{\pi \rho \alpha_{ss}(\lambda)}{192n^2 + 768n + 720} \left(45n(n-1) (\xi_D^T S_A \xi_D)^2 + 10n(1-5n) \xi_D^T S_A S_A \xi_D \right. \\ \left. - 2n(19n+25) (\xi_D^T S_A \xi_D \operatorname{tr}(S_A) - \xi_D^T S_A S_A \xi_D) \right. \\ \left. + (29n^2 + 91n + 80) \operatorname{tr}(S_A S_A) + (31n^2 + 105n + 80) (\operatorname{tr}(S_A)^2 - \operatorname{tr}(S_A S_A)) \right).$$

(13.2.42)

13.2.8 Sampling the distribution of visible normals for a cosine lobe $D_{\cos}(m, \xi_D, n)$

We address here the problem of sampling the distribution of visible microflake normals [HDCD15] given that the microflake distribution is a cosine lobe $D_{\cos}(m, \xi_D, n)$. This sampling procedure is useful when using trigonometric lobes in the standard microflake model or in the simplified self-shadowing model. Sampling visible normal distributions is actually the same problem as sampling the specular phase function. We want to sample the probability density function

$$D_{\omega_i}(\omega_o) = \frac{D_{\cos}(\omega_h, \xi_D, n)}{4 \int D_{\cos}(m, \xi_D, n) \langle \omega_i \cdot m \rangle dm} \quad (13.2.43)$$

with $\omega_h = \frac{\omega_i + \omega_o}{\|\omega_i + \omega_o\|}$. It is more convenient to work in the space of microflake normals, and to sample normals instead of outgoing directions. Using the Jacobian of the transformation from microflake normals to outgoing directions [WMLT07], the *pdf* for sampling microflake normals writes

$$D_{\omega_i}(m) = \frac{D_{\cos}(m, \xi_D, n) \langle \omega_i \cdot m \rangle}{\int D_{\cos}(m, \xi_D, n) \langle \omega_i \cdot m \rangle dm}. \quad (13.2.44)$$

We could not find a procedure for sampling directly this function. We propose a sampling procedure based on rejection sampling (Sec. B).

Using rejection sampling for sampling visible normals of $D_{\cos}(m, \xi_D, n)$

We have found that it is possible to sample the function

$$\frac{D_{\cos}(m, \xi_D, n) (\omega_i \cdot m)^2}{\int D_{\cos}(m, \xi_D, n) (\omega_i \cdot m)^2 dm}$$

which is relatively similar to $D_{\omega_i}(m)$ (Eq. 13.2.44) on the hemisphere for which $D_{\omega_i}(m)$ is not null. We propose a rejection sampling method based on this function. We introduce the normalized *pdf*

$$D_{\omega_i}^s(m) = \frac{D_{\cos}(m, \xi_D, n) ((\omega_1 \cdot m)^2 + 1/4)}{\int D_{\cos}(m, \xi_D, n) ((\omega_1 \cdot m)^2 + 1/4) dm}$$

because we have

$$|\omega_1 \cdot m| \leq (\omega_1 \cdot m)^2 + 1/4$$

so that we can write

$$\frac{D_{\cos}(m, \xi_D, n) |\omega_i \cdot m|}{\int D_{\cos}(m, \xi_D, n) |\omega_i \cdot m| dm} \leq M D_{\omega_i}^s(m)$$

with

$$M = \frac{\int D_{\cos}(m, \xi_D, n) ((\omega_1 \cdot m)^2 + 1/4) dm}{\int D_{\cos}(m, \xi_D, n) |m \cdot \omega_i| dm}.$$

Using rejection sampling, we can generate samples from the *pdf*

$$\frac{D_{\cos}(m, \xi_D, n)|\omega_i \cdot m|}{\int D_{\cos}(m, \xi_D, n)|\omega_i \cdot m| dm}$$

by generating samples from $D_{\omega_i}^s(m)$ and accepting samples with probability

$$\frac{D_{\cos}(m, \xi_D, n)|\omega_i \cdot m|}{\int D_{\cos}(m, \xi_D, n)|\omega_i \cdot m| dm} \frac{1}{MD_{\omega_i}^s(m)} = \frac{|\omega_i \cdot m|}{(\omega_i \cdot m)^2 + 1/4}.$$

Once we get a sample m , we invert its direction if $\omega_i \cdot m < 0$ and we obtain a sample for D_{ω_i} .

Sampling $D_{\omega_i}^s(m)$

We show here how to sample the *pdf*

$$D_{\omega_i}^s(m) = \frac{D_{\cos}(m, \xi_D, n)((\omega_i \cdot m)^2 + 1/4)}{\int D_{\cos}(m, \xi_D, n)((\omega_i \cdot m)^2 + 1/4) dm}.$$

We choose a reference frame such that $\xi_D = [0, 0, 1]^T$ and $\omega = [\sin(t_i), 0, \cos(t_i)]^T$, and find

$$D_{\omega_i}^s(\phi, \theta) = \cos(\theta)^{2n} ((\cos(t_i) \cos(\phi) \sin(\theta) + \sin(t_i) \cos(\theta))^2 + 1/4) \frac{(2n+1)(2n+3)}{\pi(8n \cos(t_i)^2 + 2n+7)}.$$

The marginal *pdf* for θ is given by

$$f_{\theta}(\theta) = \int D_{\omega_i}^s(\phi, \theta) \sin(\theta) d\phi = \frac{(2n+3)(2n+1) \cos(\theta)^{2n} \sin(\theta)}{8 \cos(t_i)^{2n} + 2n+7} \left(\cos(\theta)^2 (3 \cos(t_i)^2 - 1) - \cos(t_i)^2 + \frac{3}{2} \right).$$

For sampling this function, we rely on inverse transform sampling, meaning that we sample a random number $U \in (0, 1)$ and solve for θ the equation

$$U = \int_0^{\theta} f_{\theta}(x) dx.$$

We find that

$$\int_0^{\theta} f_{\theta}(x) dx = \frac{-\cos(\theta)^{2n+1}}{8 \cos(t_i)^{2n} + 2n+7} \left((2n+1) (3 \cos(t_i)^2 - 1) \cos(\theta)^2 + (2n+3) \left(-\cos(t_i)^2 + \frac{3}{2} \right) \right).$$

In our implementation, we solve this function numerically using Brent's method, although other methods could also be used. Once θ has been sampled, ϕ must be sampled from the *pdf*

$$\frac{D_{\omega_i}^s(\phi, \theta)}{f_{\theta}(\theta)}.$$

This leads to the following equation for sampling ϕ :

$$\begin{aligned} V &= \int_0^{\phi} \frac{D_{\omega_i}^s(x, \theta)}{f_{\theta}(\theta)} dx \\ &= \frac{2 \sin(t_i)^2 \sin(\theta)^2 (\cos(\phi) \sin(\phi) + \phi) + 8 \sin(\theta) \cos(\theta) \cos(t_i) \sin(t_i) \sin(\phi) + 4 \cos(\theta)^2 \cos(t_i)^2 \phi + \phi}{2\pi(6 \cos(\theta)^2 \cos(t_i)^2 - 2 \cos(\theta)^2 - 2 \cos(t_i)^2 + 3)}, \end{aligned} \tag{13.2.45}$$

V being another random number in $(0, 1)$. Again, we invert this equation numerically with Brent's method, and we obtain a sample $m = \begin{bmatrix} \cos(\phi) \sin(\theta) \\ \sin(\phi) \sin(\theta) \\ \cos(\theta) \end{bmatrix}$.

13.2.9 Sampling the distribution of visible normals for a sine lobe $D_{\sin}(m, \xi_D, n)$

For sine lobes, we use the fact that $D_{\sin}(m, \xi_D, n)$ can be written as a sum of cosine lobes $D_{\cos}(m, \xi_D, n)$ (Sec. 13.2.4). We first sample one of the cosine lobes proportionally to their projected area in direction ω_i , and then sample a visible normal from this lobe using the procedure derived in the previous paragraph (Sec. 13.2.8).

13.2.10 Sampling the single scattering phase function f_{ss} for $D_{\cos}(m, \xi_D, n)$

In Sec. 13.2.8 and 13.2.9, we have derived sampling procedures for the distribution of visible normals. Now, we derive similar sampling procedures for the single scattering phase function f_{ss} . The single scattering phase function is given by

$$f_{ss}(\omega_i \rightarrow \omega_o) = \frac{\rho\alpha_{ss}(\lambda)A(\omega_o)A(\omega_i)D(\omega_h)}{4\sigma_s(\omega_i)}.$$

We use $D_{\omega_i}^s$ (Eq. 13.2.8) in the space of outgoing directions

$$D_{\omega_i}^s(\omega_o) = \frac{D(\omega_h)((\omega_h \cdot \omega_i)^2 + 1/4)}{4|\omega_h \cdot \omega_i| \int D(\omega_h)((\omega_h \cdot \omega_i)^2 + 1/4) dm}$$

and write

$$f_{ss}(\omega_i \rightarrow \omega_o) = \frac{\rho\alpha_{ss}(\lambda)A(\omega_o)A(\omega_i)D(\omega_h)}{4\sigma_s(\omega_i)} \quad (13.2.46)$$

$$= D_{\omega_i}^s(\omega_o) \frac{\rho\alpha_{ss}(\lambda)A(\omega_o)A(\omega_i)}{\sigma_s(\omega_i)} \frac{|\omega_h \cdot \omega_i| \int D(\omega_h)((\omega_h \cdot \omega_i)^2 + 1/4) dm}{((\omega_h \cdot \omega_i)^2 + 1/4)} \quad (13.2.47)$$

$$\leq D_{\omega_i}^s(\omega_o) \frac{\rho\alpha_{ss}(\lambda)A_{max}A(\omega_i) \int D(\omega_h)((\omega_h \cdot \omega_i)^2 + 1/4) dm}{\sigma_s(\omega_i)} \quad (13.2.48)$$

$$\leq D_{\omega_i}^s(\omega_o)M_A \quad (13.2.49)$$

with $A_{max} = \max_{\omega} A(\omega)$ and

$$M_A = \frac{A_{max} \int D(m)((m \cdot \omega_i)^2 + 1/4) dm}{\int A(-\omega_i + 2m(m \cdot \omega_i))D(m)|m \cdot \omega_i| dm}.$$

Using this relation between $D_{\omega_i}^s$ and f_{ss} , we can generate samples from f_{ss} by generating samples from $D_{\omega_i}^s$ and accepting them with probability

$$\frac{f_{ss}(\omega_i \rightarrow \omega_o)}{MD_{\omega_i}^s(\omega)} = \frac{A(\omega_o)|\omega_h \cdot \omega_i|}{A_{max}((\omega_h \cdot \omega_i)^2 + 1/4)}. \quad (13.2.50)$$

13.2.11 Sampling the single scattering phase function f_{ss} for $D_{\sin}(m, \xi_D, n)$

In the case of a sine distribution $D_{\sin}(m, \xi_D, n)$, we rely on the sampling procedure for visible normals proposed in Sec. 13.2.9. This method generates samples from the *pdf*

$$D_{\omega_i}(\omega_o) = \frac{D(\omega_h)}{4 \int D(m)|m \cdot \omega_i| dm}.$$

We can write

$$f_{ss}(\omega_i \rightarrow \omega_o) = \frac{\rho\alpha_{ss}(\lambda)A(\omega_i)A(\omega_o)D(\omega_h)}{4\sigma_s(\omega_i)} \quad (13.2.51)$$

$$\leq D_{\omega_i}(\omega_o) \frac{\rho\alpha_{ss}(\lambda)A_{max}A(\omega_i) \int D(m)|m \cdot \omega_i| dm}{\sigma_s(\omega_i)} \quad (13.2.52)$$

$$(13.2.53)$$

so we can sample f_{ss} for a sine distribution $D_{\sin}(m, \xi_D, n)$ by generating samples ω_o from $D_{\omega_i}(\omega_o)$ and accepting them with probability

$$\frac{pdf_1(\omega_o)}{Mpdf_2(\omega_o)} = \frac{\rho\alpha_{ss}(\lambda)A(\omega_i)A(\omega_o)D(\omega_h)}{4\sigma_s(\omega_i)} \frac{4 \int D(m)|m \cdot \omega_i| dm}{D(\omega_h)} \frac{\sigma_s(\omega_i)}{\rho\alpha_{ss}(\lambda)A_{max}A(\omega_i) \int D(m)|m \cdot \omega_i| dm} \quad (13.2.54)$$

$$= \frac{A(\omega_o)}{A_{max}}. \quad (13.2.55)$$

13.2.12 Sampling the multiple scattering phase function f_{ms}

Our multiple scattering phase function is given by

$$f_{ms}(\omega_o) = \frac{\sigma_t(\omega_o) - \sigma_s(\omega_o)/\alpha_{ss}(\lambda)}{\int \sigma_t(\omega) - \int \sigma_s(\omega)/\alpha_{ss}(\lambda)}. \quad (13.2.56)$$

We observe that in the case of isotropic self-shadowing, this phase function reduces to the *pdf*

$$\frac{A(1-A) \int D(m)\langle m \cdot \omega_o \rangle dm}{\int A \int D(m)\langle m \cdot \omega_o \rangle dmd\omega - \int A \int AD(m)\langle m \cdot \omega_o \rangle dmd\omega} = \frac{\int D(m)\langle m \cdot \omega_o \rangle dm}{\int \int D(m)\langle m \cdot \omega_o \rangle dmd\omega}$$

which is proportional to the projected area of microflakes in direction ω_o . Given a single microflake with normal m , we can sample a direction ω proportionally to the projected area of the microflake, $|m \cdot \omega|$, by sampling a cosine lobe around direction m . Similarly, we can sample the projected area of a cloud of microflakes by sampling first a normal from the distribution of normals D , and then by sampling a cosine lobe around this normal. In Sec. 13.2.13 and 13.2.14, we show how to sample normals for D_{\cos} and D_{\sin} .

We derive a rejection sampling procedure for f_{ms} based on this *pdf*:

$$\begin{aligned} \frac{\sigma_t(\omega_o) - \sigma_s(\omega_o)/\alpha_{ss}(\lambda)}{\int \sigma_t(\omega) - \int \sigma_s(\omega)/\alpha_{ss}(\lambda)} &= \frac{A(\omega_o) \int (1 - A(-\omega_o + 2m(m \cdot \omega_o)))D(m)\langle m \cdot \omega_o \rangle dm}{\int A(\omega) \int (1 - A(-\omega + 2m(m \cdot \omega)))D(m)\langle m \cdot \omega \rangle dm d\omega} \quad (13.2.57) \\ &< \frac{\int D(m)\langle m \cdot \omega_o \rangle dm}{\int \int D(m)\langle m \cdot \omega_o \rangle dmd\omega_o} \frac{A_{max}(1 - A_{min}) \int \int D(m)\langle m \cdot \omega_o \rangle dmd\omega_o}{\int A(\omega) \int (1 - A(-\omega + 2m(m \cdot \omega)))D(m)\langle m \cdot \omega \rangle dm d\omega}. \end{aligned} \quad (13.2.58)$$

We find the following probability of acceptance:

$$\frac{A(\omega) \int (1 - A(-\omega + 2m(m \cdot \omega)))D(m)\langle m \cdot \omega \rangle dm}{A_{max}(1 - A_{min}) \int D(m)\langle m \cdot \omega_o \rangle dm}. \quad (13.2.59)$$

This probability tends to 1 when the self-shadowing is almost anisotropic, and it tends to be smaller when A_{min} and A_{max} are very different, meaning that this sampling procedure is less efficient in such case.

13.2.13 Sampling a normal from D_{\cos}

The distribution

$$D_{\cos}(\omega, \xi, n) = (\omega \cdot \xi)^{2n} \frac{2n+1}{4\pi}$$

can be sampled easily in the reference frame in which $\xi = [0, 0, 1]$, using inverse transform sampling. We first sample the azimuth ϕ uniformly in $(0, 2\pi)$. Then, the marginal distribution for θ is

$$\int_0^{2\pi} \cos(\theta)^{2n} \frac{2n+1}{4\pi} \sin(\theta) d\phi = \frac{2n+1}{2} \cos(\theta)^{2n} \sin(\theta).$$

The cumulative distribution function writes

$$\int_0^x \frac{2n+1}{2} \cos(\theta)^{2n} \sin(\theta) d\theta = \frac{1}{2} (1 - \cos(x)^{2n+1}).$$

This function can be inverted:

$$U = \frac{1}{2} (1 - \cos(x)^{2n+1}) \Rightarrow \cos(x)^{2n+1} = 1 - 2U,$$

If $U \geq 1/2$:

$$x = \arccos \left((1 - 2U)^{\frac{1}{2n+1}} \right). \quad (13.2.60)$$

If $U \leq 1/2$:

$$x = \pi - \arccos \left((2U - 1)^{\frac{1}{2n+1}} \right). \quad (13.2.61)$$

Therefore, θ can be sampled by sampling U in $(0, 1)$ and then using Eq. 13.2.60 or 13.2.61. Finally, the sample

$$m = \begin{bmatrix} \cos(\phi) \sin(\theta) \\ \sin(\phi) \sin(\theta) \\ \cos(\theta) \end{bmatrix}$$

must be rotated back in the original reference frame of $D_{\cos}(\omega, \xi, n)$.

13.2.14 Sampling a normal from D_{\sin}

Again, we use the fact that $D_{\sin}(m, \xi_D, n)$ can be written as a sum of cosine lobes $D_{\cos}(m, \xi_D, n)$ (Sec. 13.2.4). We first sample one of the cosine lobes randomly and follow the procedure derived in Sec. 13.2.13.

13.2.15 Sampling a cosine lobe

Sampling simple cosine lobes $\cos(\theta)$ is very common in path tracing. We recall one of the existing procedures here. Given a lobe axis m , sampling a cosine lobe around this direction can be done easily in the frame in which $m = [0, 0, 1]$. In this frame, the cosine lobe on the upper hemisphere in spherical coordinates writes

$$\frac{\cos(\theta)}{\pi} \mathbb{1}_{\theta \in (0, \frac{\pi}{2})}(\theta).$$

We first sample ϕ uniformly in $(0, 2\pi)$. The marginal *pdf* for θ writes

$$\int_0^{2\pi} \frac{\cos(\theta)}{\pi} \sin(\theta) d\phi = 2 \cos(\theta) \sin(\theta).$$

Then the cumulative distribution function writes

$$\int_0^x 2 \cos(\theta) \sin(\theta) d\theta = \sin(x)^2.$$

So θ can be sampled using a random sample U in $(0, 1)$ and the inverse of the cumulative function:

$$\theta = \arcsin \left(\sqrt{U} \right).$$

13.2.16 Case of isotropic distribution $D(\omega) = \frac{1}{4\pi}$

We derived above the closed-form expressions and sampling procedures of our self-shadowing model for sine and cosine distributions. We now consider the case of an isotropic microflake distribution

$$D(\omega) = \frac{1}{4\pi}$$

which is much simpler. The attenuation coefficient writes

$$\sigma_t(\omega_i) = \rho A(\omega_i) \frac{1}{4\pi} \int \langle \omega_i \cdot \omega \rangle d\omega = \frac{\rho A(\omega_i)}{4}$$

and the single scattering coefficient

$$\sigma_{ss}(\omega_i) = \rho \alpha_{ss}(\lambda) A(\omega_i) \int A(\omega_o) \frac{1}{4\pi} \langle \omega_i \cdot \omega \rangle d\omega \quad (13.2.62)$$

$$= \rho \alpha_{ss}(\lambda) A(\omega_i) \frac{1}{4\pi} \frac{1}{4} \int A(\omega_o) d\omega_o \quad (13.2.63)$$

$$= \rho \alpha_{ss}(\lambda) A(\omega_i) \frac{1}{4\pi} \frac{1}{4} \frac{4\pi}{3} \text{tr}(S_A) \quad (13.2.64)$$

$$= \rho \alpha_{ss}(\lambda) A(\omega_i) \frac{\text{tr}(S_A)}{12}. \quad (13.2.65)$$

The single scattering phase function writes

$$f_{ss}(\omega_i \rightarrow \omega_o) = \frac{A(\omega_i) A(\omega_o) \frac{1}{4\pi}}{A(\omega_i) \frac{1}{4\pi} \int A(\omega) d\omega} = \frac{3A(\omega_o)}{4\pi \text{tr}(S_A)}.$$

This phase function is proportional to the self-shadowing function in the outgoing direction $A(\omega_o)$. The function A can be written as a sum of three \cos^2 lobes as shown in equation 13.2.1. Using this expression, sampling $A(\omega_o)$ can be done by sampling one of the lobe proportionally to the eigenvalues a_1 , a_2 and a_3 , and then sampling a \cos^2 lobe (almost the same as in Section 13.2.13) around the corresponding eigenvector. The multiple scattering phase function writes

$$f_{ms}(\omega_o) = \frac{\sigma_t(\omega_o) - \sigma_s(\omega_o)/\alpha_{ss}(\lambda)}{\int \sigma_t - \int \sigma_s/\alpha_{ss}(\lambda)}$$

with

$$\int \sigma_t(\omega) d\omega = \rho \frac{1}{4} \int A(\omega) d\omega = \frac{\pi \text{tr}(S_A)}{3}$$

and

$$\int \sigma_{ss}(\omega) d\omega = \rho \alpha_{ss}(\lambda) \frac{\text{tr}(S_A)}{12} \int A(\omega) d\omega = \rho \alpha_{ss}(\lambda) \frac{\text{tr}(S_A)}{12} \frac{4\pi \text{tr}(S_A)}{3} = \frac{\pi \text{tr}(S_A)^2}{9}$$

so we have

$$f_{ms}(\omega_o) = \frac{A(\omega_o) \frac{1}{4} - A(\omega_o) \frac{\text{tr}(S_A)}{12}}{\frac{\pi}{3} \text{tr}(S_A) - \frac{\pi \text{tr}(S_A)^2}{9}} = \frac{3A(\omega_o)}{\text{tr}(S_A) 4\pi} = f_{ss}(\omega_i \rightarrow \omega_o). \quad (13.2.66)$$

The multiple scattering phase function f_{ms} is the same as the single scattering phase function f_{ss} in the case of an isotropic microflake normal distribution.

13.2.17 Combining trigonometric lobes

In previous sections, we have derived expressions for σ_t , σ_{ss} , σ_{ms} , f_{ss} , f_{ms} and sampling procedures for cosine, sine and isotropic microflake distributions. We now provide details for combining various lobes, that is, when the microflake normal distribution D can be written

$$D(\omega) = \sum w_i D_i(\omega)$$

with $\sum w_i = 1$. Several expressions are linear in D , so that we obtain:

$$\sigma_t(\omega) = \rho A(\omega) \int \sum w_i D_i(m) \langle m \cdot \omega \rangle dm = \sum w_i \sigma_t^i(\omega)$$

$$\begin{aligned}
\sigma_{ss}(\omega) &= \rho\alpha_{ss}(\lambda)A(\omega) \int A(-\omega + 2m(m \cdot \omega)) \sum w_i D_i(m) dm = \sum w_i \sigma_{ss}^i(\omega) \\
\int \sigma_t(\omega) d\omega &= \int \sum w_i \sigma_t^i(\omega) d\omega = \sum w_i \int \sigma_t^i(\omega) d\omega \\
\int \sigma_{ss}(\omega) d\omega &= \int \sum w_i \sigma_{ss}^i(\omega) d\omega = \sum w_i \int \sigma_{ss}^i(\omega) d\omega
\end{aligned}$$

The single scattering phase function is not a linear combination of phase functions derived from each lobes, because of the normalization factor:

$$\begin{aligned}
f_{ss}(\omega_i \rightarrow \omega_o) &= \frac{\rho\alpha_{ss}(\lambda)A(\omega_i)A(\omega_o) \sum w_i D_i(\omega_h)}{4 \sum w_i \sigma_{ss}^i(\omega)} \\
&= \sum \frac{w_i \sigma_{ss}^i(\omega)}{\sum w_i \sigma_{ss}^i(\omega)} \frac{\rho\alpha_{ss}(\lambda)A(\omega_i)A(\omega_o)D_i(\omega_h)}{4\sigma_{ss}^i(\omega)} \\
&= \sum \frac{w_i \sigma_{ss}^i(\omega)}{\sum w_i \sigma_{ss}^i(\omega)} f_{ss}^i(\omega_i \rightarrow \omega_o).
\end{aligned}$$

The weights depend on direction ω_i . Intuitively, this means that if one lobe has a small projected area in direction ω_i , or if there is more self-shadowing for this lobe because reflected rays are such that $A(\omega_o)$ is small, then this lobe contributes less to the combined phase function from direction ω_i . Similarly, the combined multiple scattering phase function writes:

$$f_{ms}(\omega_o) = \frac{\sigma_t(\omega_o) - \sigma_{ss}(\omega_o)/\alpha_{ss}(\lambda)}{\int \sigma_t(\omega) d\omega - \int \sigma_{ss}(\omega)/\alpha_{ss}(\lambda) d\omega} \quad (13.2.67)$$

$$= \frac{\sum w_i (\sigma_t^i(\omega_o) - \sigma_{ss}^i(\omega_o)/\alpha_{ss}(\lambda))}{\sum w_i (\int \sigma_t^i(\omega) d\omega - \int \sigma_{ss}^i(\omega)/\alpha_{ss}(\lambda) d\omega)} \quad (13.2.68)$$

$$= \sum \frac{w_i (\int \sigma_t^i(\omega) d\omega - \int \sigma_{ss}^i(\omega)/\alpha_{ss}(\lambda) d\omega)}{\sum w_i (\int \sigma_t^i(\omega) d\omega - \int \sigma_{ss}^i(\omega)/\alpha_{ss}(\lambda) d\omega)} \frac{\sigma_t^i(\omega_o) - \sigma_{ss}^i(\omega_o)/\alpha_{ss}(\lambda)}{\int \sigma_t^i(\omega) d\omega - \int \sigma_{ss}^i(\omega)/\alpha_{ss}(\lambda) d\omega} \quad (13.2.69)$$

$$= \sum \frac{w_i (\int \sigma_t^i(\omega_o) - \int \sigma_{ss}^i(\omega_o)/\alpha_{ss}(\lambda))}{\sum w_i (\int \sigma_t^i(\omega) d\omega - \int \sigma_{ss}^i(\omega)/\alpha_{ss}(\lambda) d\omega)} f_{ms}^i(\omega_o). \quad (13.2.70)$$

Sampling these phase functions from direction ω_i can be done by computing view dependent weights of each lobes, and then sampling one function f_{ss}^i or f_{ms}^i accordingly.

13.3 Implementing the simplified self-shadowing model with the SGGX distribution

In this section, we discuss the implementation of our simplified self-shadowing model (Sec. 12.2) using the SGGX distribution. The attenuation coefficient is given by

$$\sigma_t(\omega) = A\rho \int D(\omega_m) \langle \omega \cdot \omega_m \rangle d\omega_m.$$

Using SGGX microflake distributions, this writes

$$\sigma_t(\omega) = A\rho\sqrt{\omega^T S \omega}$$

with S the SGGX matrix. Similarly, the single scattering coefficient writes

$$\sigma_{ss}(\omega) = \alpha_{ss}(\lambda)A\sigma_t(\omega) = \alpha_{ss}(\lambda)A^2\rho\sqrt{\omega^T S \omega}.$$

The single scattering phase function writes

$$f_{ss}(\omega \rightarrow \omega') = \frac{\rho\alpha_{ss}(\lambda)A^2D(\omega_h)}{4\sigma_{ss}(\omega)} = \frac{D(\omega_h)}{4\sqrt{\omega^T S \omega}}$$

which is the SGGX specular phase function [HDCD15]. For the microscopic multiple scattering, we have

$$\sigma_{ms}(\omega) = \alpha_{ms}(\lambda)\sigma_t(\omega)(1-A) = \alpha_{ms}(\lambda)\rho A(1-A)\sqrt{\omega^T S \omega} \quad (13.3.1)$$

$$f_{ms}(\omega') = \frac{\sigma_{ms}(\omega')}{\int \sigma_{ms}(\omega'') d\omega''} = \frac{\sqrt{\omega'^T S \omega'}}{\int \sqrt{\omega''^T S \omega''} d\omega''}. \quad (13.3.2)$$

Unfortunately, there is no closed-form expression for $\int \sqrt{\omega^T S \omega}$. As this integral does not depend on the orientation of the SGGX distribution, but only on the eigenvalues of the SGGX matrix, we precompute numerically this integral for several sets of eigenvalues. Assuming that the SGGX is normalized so that its largest eigenvalue is 1, we pre-compute $\int \sqrt{\omega^T S \omega}$ for several pairs of smaller eigenvalues a_2 and a_3 . Appendix D shows the values we used in our implementation.

The multiple scattering phase function $f_{ms}(\omega_o)$ is proportional to the projected area of microflakes in direction ω_o . As discussed in Sec. 13.2.12, sampling this function could be done by sampling a normal from distribution D and then sampling a cosine lobe around this normal. However, there is no efficient way of sampling a normal from SGGX distributions. Available methods are based on rejection sampling and can be inefficient in some configurations. We observed that the distribution

$$\frac{\omega^T S \omega}{\int \omega'^T S \omega' d\omega'}$$

is rather close to $f_{ms}(\omega_o)$, and simple to sample, since it is actually the same distribution as the multiple scattering phase function in the case of anisotropic self-shadowing and isotropic microflake distribution (Sec. 13.2.16). Based on this function, we can derive a rejection sampling method. We know that

$$\sqrt{\omega^T S \omega} \leq \omega^T S \omega + \frac{1}{4}$$

so that

$$\frac{\sqrt{\omega^T S \omega}}{\int \sqrt{\omega'^T S \omega'} d\omega'} \leq \frac{\omega^T S \omega + 1/4}{\int \sqrt{\omega'^T S \omega'} d\omega'} \quad (13.3.3)$$

$$\leq \frac{\omega^T S \omega + 1/4}{\int (\omega'^T S \omega' + 1/4) d\omega'} \frac{\int (\omega'^T S \omega' + 1/4) d\omega'}{\int \sqrt{\omega'^T S \omega'} d\omega'}. \quad (13.3.4)$$

We can sample

$$\frac{\omega^T S \omega + 1/4}{\int (\omega'^T S \omega' + 1/4) d\omega'}$$

and accept samples with probability

$$\frac{\sqrt{\omega^T S \omega}}{\int \sqrt{\omega'^T S \omega'} d\omega'} \frac{\int (\omega'^T S \omega' + 1/4) d\omega'}{\omega^T S \omega + 1/4} \frac{\int \sqrt{\omega'^T S \omega'} d\omega'}{\int (\omega'^T S \omega' + 1/4) d\omega'} = \frac{\sqrt{\omega^T S \omega}}{\omega^T S \omega + 1/4}.$$

Chapter 14

Volume downsampling with our microscopic self-shadowing model

In previous chapters, we introduced a new volume model called the microscopic self-shadowing model (Chap. 12). We provided all the derivations for implementing this model and its simplified version so that it can be used for efficient rendering (Chap. 13).

As mentioned in chapter 12, the main motivation for this model was the introduction of new parameters for the anisotropic attenuation and the amount of multiple scattering, in order to prefilter blocks of high-resolution voxels. In this chapter, we provide algorithms for estimating the parameters of the microscopic self-shadowing model for each low-resolution voxel. We presented the work of Zhao, Wu et al. in Sec. 11.3.3 and discussed its limitations raised by the optimization of scattering parameters with image-space errors on the entire volume. In this chapter, we show that optimizing parameters locally overcomes some of these limitations, and that it is also more efficient.

14.1 Downsampling algorithms

In this section, we introduce new algorithms for downsampling standard microflake volumes using our self-shadowing model. We assume that input voxels contain one density value, parameters for the microflake distribution and a specular albedo. We describe two algorithms that we used for generating results in Sec. 14.2: **Aniso** uses anisotropic self-shadowing functions in each low-resolution voxel, and **Iso** relies on our simplified self-shadowing model (Sec. 12.2) for which each low-resolution voxel has less parameters. We also describe in this section two naïve algorithms that use the standard microflake model in output voxels: **Linear** performs linear prefiltering for densities and albedos, and **Transp** performs linear prefiltering for albedos but computes density parameters that preserve the local transparency of the input volume. These naïve algorithms are used for comparison in Sec. 14.2.

Fig. 14.1 illustrates our downsampling pipeline and summarizes our input and output models for each algorithm. Each of our output voxels have a unique single scattering albedo, which could be insufficient for downsampling datasets with strong correlations between microflake distributions and albedos as shown in Sec. 11.2.2. We discuss this case in Sec. 14.2.4.

14.1.1 Overview

In the four algorithms described in this section, each block of input voxels (Fig. 14.1) is downsampled into one low-resolution voxel independently of other low-resolution voxels. The parameters of each low-resolution voxel are computed using different strategies summarized in Table 14.1.

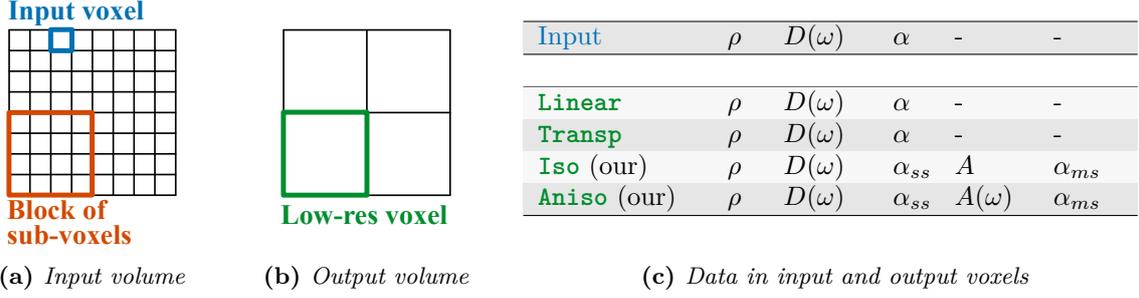


Figure 14.1: Our input voxels (a) have parameters for the standard microflakes model (density ρ , microflake distribution $D(\omega)$ and albedo α). Each low-resolution voxel (b) approximates a cube of input voxels (or block of input voxels). In this paper, we compare four downsampling algorithms (c). **Linear** and **Transp** use the standard microflake model. **Iso** and **Aniso** use our self-shadowing model and output voxels store additional parameters in this case, for the self-shadowing function A and for the multiple scattering albedo α_{ms} .

| | ρ | α_{ss} | A | α_{ms} |
|---------------|---------------------|---------------------|-------------|---------------|
| Linear | Linear (Eq. 14.1.1) | Linear (Eq. 14.1.2) | - | - |
| Transp | Mean (Eq. 14.1.5) | Linear (Eq. 14.1.2) | - | - |
| Aniso | Max (Eq. 14.1.7) | Linear (Eq. 14.1.2) | Sec. 14.1.4 | Sec. 14.1.4 |
| Iso | Mean (Eq. 14.1.6) | Linear (Eq. 14.1.2) | Sec. 14.1.4 | Sec. 14.1.4 |

Table 14.1: Summary of the algorithms described in this chapter.

We do not contribute to microflake prefiltering. Algorithms described in this section are valid for any number of SGGX or trigonometric lobes in input and output voxels. In our implementation, we used a single SGGX lobe in each low-resolution voxel, or an equivalent distribution using trigonometric lobes (Fig. 13.1), and we prefiltered normal distributions using linear SGGX prefiltering [HDCD15].

14.1.2 Algorithm Linear

This is the naïve linear algorithm: given ρ_i and α_i , densities and albedos of N_{iv} input voxels (Fig. 14.1), the parameters of the corresponding low-resolution voxel are computed using

$$\rho = \frac{1}{N_{iv}} \sum \rho_i \quad (14.1.1)$$

and

$$\alpha = \frac{\sum \alpha_i \rho_i}{\sum \rho_i}. \quad (14.1.2)$$

14.1.3 Algorithm Transp

This algorithm first computes transparency of the block of input voxel in the three axis of the voxel grid: $T(\omega_X)$, $T(\omega_Y)$ and $T(\omega_Z)$. These quantities are computed exactly from directional transparencies of each input voxel. Given the size of input voxels L_{iv} , the transparency of one input voxel in direction ω_i is given by

$$\exp \left(-L_{iv} \rho_i \int D_i(m) \langle \omega_i \cdot m \rangle dm \right). \quad (14.1.3)$$

Similarly, the transparency of the low-resolution voxel in direction ω_i is given by

$$\exp(-L\sigma_t(\omega_i)) = \exp \left(-L\rho \int D(m) \langle \omega_i \cdot m \rangle dm \right) \quad (14.1.4)$$

with L the size of the low-resolution voxel. We compute density parameters that preserve the transparency in each direction ω_i , and we average them:

$$\rho = \frac{1}{3} \sum_{i \in \{X, Y, Z\}} \frac{-\log(T(\omega_i))}{L \int D(m) \langle \omega_i \cdot m \rangle dm}. \quad (14.1.5)$$

14.1.4 Downsampling using our self-shadowing model (Iso and Aniso)

Given a prefiltered microflake distribution D , our algorithms first compute the directional transparencies of the block of input voxels as in algorithm **Transp**. Then, they compute a temporary density value ρ_0 and temporary parameters for a self-shadowing function A_0 . These temporary parameters are used for estimating the amount of self-shadowing in the low-resolution voxel. The amount of self-shadowing is also computed in the block of input voxels (Fig. 14.1). The final density ρ and self-shadowing parameters are computed so that the amount of self-shadowing is the same in the block of input voxels and the low-res voxel. Single scattering albedos are downsampled linearly using Equation 14.1.2, and the computation the multiple scattering albedo is discussed in Sec. 14.1.4.

Computing temporary parameters ρ_0 and A_0 from directional transparencies

Given the size of low-res voxels L , the transparency of a low-res voxel in direction ω_i is given by

$$\exp(-L\sigma_t(\omega_i)) = \exp\left(-L\rho_0 A_0(\omega_i) \int D(m) \langle \omega_i \cdot m \rangle dm\right).$$

Algorithm **Iso** only uses the simplified microflake model in which $A_0(\omega) = A_0, \forall \omega$. In this case, we set $A_0 = 1$ and compute ρ for preserving the average transparency as in algorithm **Transp**:

$$\rho_0 = \frac{1}{3} \sum \frac{-\log(T(\omega_i))}{L \int D(m) \langle \omega_i \cdot m \rangle dm}. \quad (14.1.6)$$

Algorithm **Aniso** uses the anisotropic self-shadowing function introduced in Section 13.2.1. It computes ρ_0 using

$$\rho_0 = \max_{i \in \{X, Y, Z\}} \left(\frac{-\log(T(\omega_i))}{L \int D(m) \langle \omega_i \cdot m \rangle dm} \right), \quad (14.1.7)$$

and then it computes the self-shadowing parameters with

$$A_0(\omega) = \omega^T \begin{bmatrix} s_X & 0 & 0 \\ 0 & s_Y & 0 \\ 0 & 0 & s_Z \end{bmatrix} \omega \quad (14.1.8)$$

with

$$s_i = \frac{-\log(T(\omega_i))}{\rho_0 L \int D(m) \langle \omega_i \cdot m \rangle dm}, \quad i \in \{X, Y, Z\} \quad (14.1.9)$$

so that directional transparencies of the low-res voxel match exactly $T(\omega_x)$, $T(\omega_y)$ and $T(\omega_z)$, the transparencies of the block of input voxels.

Estimating self-shadowing

At this stage, the low-resolution voxel has a correct transparency but an incorrect self-shadowing probability. For preserving self-shadowing effects, we estimate the amount of single scattering in the block of input voxels and in the low-res voxel, and we compute final parameters accordingly. More precisely, we estimate the mean amount of energy that leaves the block of input voxels after exactly one scattering event. For one incoming ray r with direction ω_r , intersecting the block of input voxels in x_{\min} and x_{\max} as shown in Fig. 14.2, the amount of single scattering writes

$$P_{ss}(r) = \int_{x_{\min}}^{x_{\max}} \sigma_s(x, \omega_r) O(x_{\min}, \omega_r, x) \int_{S^2} f(x, \omega_r \rightarrow \omega) O(x, \omega) d\omega dx \quad (14.1.10)$$

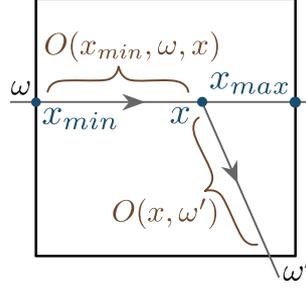


Figure 14.2: Notations for Eq. 14.1.10, for a ray in a block of input voxels or in a low-res voxel.

with

$$O(x_1, \omega, x_2) = \exp\left(-\int_{x_1}^{x_2} \sigma_t(s, \omega) ds\right) \quad (14.1.11)$$

and $O(x_1, \omega)$ a similar transmittance probability between x_1 and the limit of the block of input voxels, in direction ω (Fig. 14.2). The quantity $P_{ss}(r)$ is dimensionless but wavelength dependent. We estimate the mean amount of single scattering in the block of input voxels by casting N_{ray} rays all around the block of input voxels. In our implementation, we chose rays with directions parallel to the mains axis of the voxel grid. For each ray, we compute $P_{ss}(r_i)$ and we average the results:

$$M_{ss}^{input} = \frac{1}{N_{ray}} \sum_{rays} P_{ss}(r_i). \quad (14.1.12)$$

Similarly, we estimate the quantity M_{ss} , the amount of single scattering in the low-res voxel, using the prefiltering distribution D , ρ_0 , $A_0(\omega)$ and the corresponding σ_{ss} and f_{ss} (instead of σ_s and f) in Eq. 14.1.10.

Computing final density and self-shadowing parameters

Thanks to our model, we can control the amount of self-shadowing in the low-res voxel while preserving the voxel transparency, using $\rho = \frac{\rho_0}{\gamma}$ and $A(\omega) = \gamma A_0(\omega)$ with $\gamma \in (0, 1]$. Indeed, for any value γ , using ρ and $A(\omega)$ instead of ρ_0 and $A_0(\omega)$ in the low-res voxel does not impact the attenuation coefficient:

$$\sigma_t(\omega) = \rho A(\omega) \int D(m) \langle \omega \cdot m \rangle dm = \rho_0 A_0(\omega) \int D(m) \langle \omega \cdot m \rangle dm.$$

On the contrary, the single scattering coefficient is multiplied by γ :

$$\begin{aligned} \sigma_{ss}(\omega) &= \rho A(\omega) \int A(\omega) D(m) \langle \omega \cdot m \rangle dm \\ &= \gamma \rho_0 A_0(\omega) \int A_0(\omega) D(m) \langle \omega \cdot m \rangle dm. \end{aligned}$$

This means that the amount of self-shadowing in the low-res voxel (estimated from Eq. 14.1.10) would now be γM_{ss} . Given M_{ss}^{input} and M_{ss} , we compute the value γ that minimizes the error over various wavelengths:

$$\gamma = \arg \max_g \sum_{\lambda_i} \|M_{ss}^{input}(\lambda_i) - g M_{ss}(\lambda_i)\|^2 \quad (14.1.13)$$

λ_i being wavelengths. We used RGB albedos in our implementation.

Computation of the multiple scattering albedo σ_{ms}

Our self-shadowing model requires a multiple scattering albedo σ_{ms} that characterizes the color of light scattered at least two times at the *microscopic* scale due to self-shadowing. At rendering, multiple

scattering occurs at the microscopic scale with our model, but it can also occur several times in the low-res voxel. We need to estimate σ_{ms} such that the *effective* albedo α_e of the low-res voxel, that is, the resulting albedo of the voxel taking into account multiple scattering with microscopic and non-microscopic scattering events, is the same as the effective albedo α_e^{input} of the corresponding block of input voxels.

We first estimate the effective albedo α_e^{input} in the block of input voxels, that is, the average color of light when it leaves the block after 1 or more scattering events. We cast rays through the block of input voxels as for single scattering estimations (Fig. 14.1). For each ray, we compute a light path until the ray leaves the block, exactly as a volume path tracer would do. Then, α_e^{input} is computed averaging ray throughputs.

Now we want to find α_{ms} such that $\alpha_e = \alpha_e^{\text{input}}$. Unfortunately, there is no simple way to derive exactly α_e from α_{ms} (Sec. 11.3.1), and we want to avoid iterative optimizations because they are time consuming. We approximate α_e using:

$$\alpha_e \approx \sum P_{sh}(n)(P_{ss}\alpha_{ss} + (1 - P_{ss})\alpha_{ms})^n \quad (14.1.14)$$

with $P_{sh}(n)$ the probability for a ray of having exactly n scattering events before leaving the voxel ($\sum P_{sh}(n) = 1$) and P_{ss} the probability of self-shadowing at the microscopic scale. We cast rays in the low-res voxel and estimate probabilities $P_{sh}(n)$ for each n (up to a given value), which do not depend on σ_{ms} . For P_{ss} , we use the following approximation:

$$P_{ss} \approx \frac{\int \sigma_{ss}}{\alpha_{ss}(\lambda) \int \sigma_t}. \quad (14.1.15)$$

Finally, we can estimate α_{ms} by solving numerically the equation

$$\alpha_e^{\text{input}} = \sum P_{sh}(n)(P_{ss}\alpha_{ss} + (1 - P_{ss})\alpha_{ms})^n \quad (14.1.16)$$

for each wavelength.

We evaluated this method by computing the relative errors of the RGB values of the effective albedos in low-resolution voxels. For each low-resolution voxel, we compared the effective albedo (estimated with ray casting and using the multiple scattering albedo obtained with our approximation) with the effective albedo of the corresponding block of input voxels (also estimated with ray casting). In coarse LODs (grid resolution divided by 32 in each dimension), we obtained the relative errors shown in Fig. 14.3. Our approximations tend to underestimate the effective albedo, meaning that our multiple scattering albedos in low-resolution voxels are slightly too dark on average. The accuracy depends on the amount of shadowing and the density in the voxels. Our approach could be improved with a better understanding of these errors.

Interestingly, cases with the highest errors in the effective albedos do not correspond to the highest visual errors at rendering (See Fig. 14.4). The average albedo is important, but phase functions also contribute a lot to appearance, especially when the lighting is not uniform.

14.2 Implementation and results

14.2.1 Precomputation time

The precomputation time depends on the number of samples used for estimating self-shadowing probabilities and multiple scattering albedos (Sec. 14.1.4 and 14.1.4). In our implementation, we adapted the number of samples to the complexity of the block of input voxels. For LOD i (*i.e.* 2^i times smaller than input volume in each dimension), blocks of input voxels viewed from one side have a complexity of 4^i voxels. We cast 40×4^i rays for each non-empty low-res voxels so that our LODs are not subject to noise. Our precomputation times are shown in Table 14.2.

In their work, Zhao, Wu et al. give precomputation time of respectively 12 and 40 CPU core hours for the bunny (LOD 2) and the hairy ball (LOD 3). We obtained accurate results for these model with

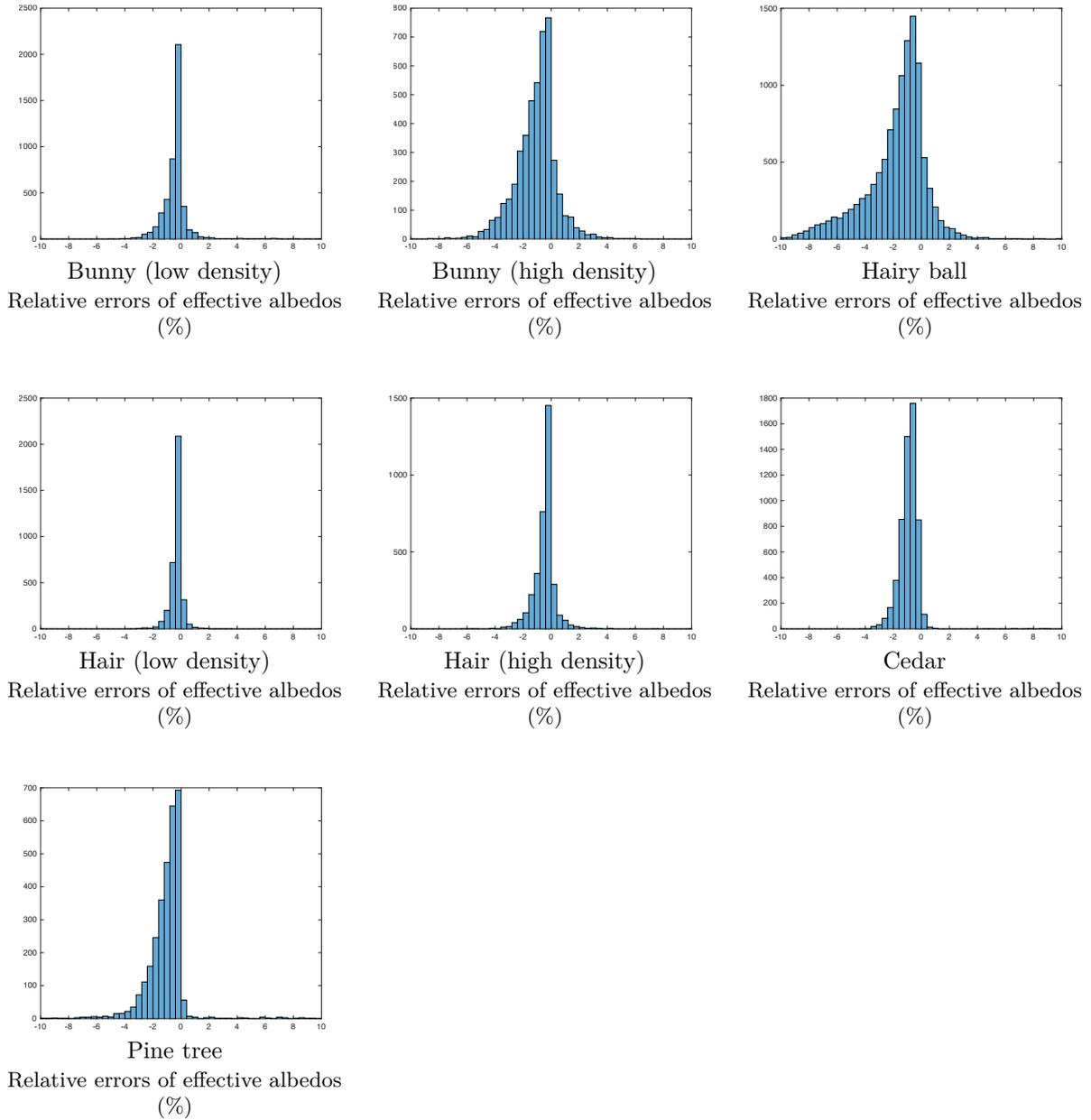


Figure 14.3: *Relative errors between measured effective albedos in our low-resolution voxels and reference effective albedos measured in the corresponding blocks of input voxels. Charts show relative errors for R, G and B values, in %. Each chart corresponds to one dataset used in Fig. 14.4. Vertical axes give the count of a given relative error in the low-res voxels of the LOD (empty voxels are not taken into account).*

respectively 0.97 and 3.2 CPU core hours. As our algorithm downsamples each block of input voxels independently, the parallelization of our method is straightforward.

14.2.2 Storage savings

Compared to the standard microflake model, our model requires additional data per voxel: 2 additional values for our trigonometric distributions compared to SGGX, 3 for multiple scattering albedos (RGB) and 6 or 1 additional values for the shadowing function using respectively our anisotropic self-shadowing model or our simplified model. Despite its additional parameters, our downsampling method allows for

| | Bunny | hair | Cedar | Pine | Hairy ball |
|----------------------|----------------|-------------|--------------|-------------|-------------------|
| Size of input vol. | $\simeq 500^3$ | 600^3 | 680^3 | 680^3 | 680^3 |
| Aniso / LOD 1 | 1.1 | 1.4 | 0.1 | 0.13 | 6.8 |
| Aniso / LOD 3 | 0.92 | 0.6 | 0.14 | 0.17 | 3.2 |
| Aniso / LOD 5 | 0.64 | 0.33 | 0.17 | 0.15 | 1.2 |

Table 14.2: Our precomputation times measured on a Intel Xeon Processor E5-2630 v3 (in CPU core hour).

huge memory savings as shown in Table 14.3.

| | Linear | Iso (SGGX) | Aniso (triglobe) |
|-------------------------------|---------------|-------------------|-------------------------|
| Per voxel values | 10 | 14 | 21 |
| Size of LOD1 <i>wrt</i> input | 12.5% | 17.5% | 26.3% |
| Size of LOD3 <i>wrt</i> input | 0.20% | 0.27% | 0.41% |

Table 14.3: Memory requirements of our prefiltering methods.

In average, for the LODs shown in Fig. 14.4, the render time for naïve LODs is around 11% of the render time for high-resolution volumes, and around 13% for our LODs with algorithm **Aniso**. This means that our LODs also decrease render time despite the additional cost compared to rendering naïve LODs.

14.2.3 Results

Fig. 14.4 compares our LODs with input volumes and results from naïve algorithms described in Section 14.1. Our downsampling method supports semi-transparent inputs with low-density voxels. LODs are only slightly more accurate in this case because naïve methods already perform well when there is no strong self-shadowing effects (Fig. 14.4a and 14.4f). However, our LODs are much more accurate than naïve LODs when input volumes have relatively dense voxels (Fig. 14.4b to 14.4e). Because we preserve local transparency, our results have correct silhouettes even when input datasets have intricate shapes (Fig. 14.4d and 14.4e). Our LODs can be computed at arbitrary scales and their appearance is consistent (Fig. 14.5).

Using anisotropic self-shadowing functions allows for preserving exactly directional transparency in low-res voxels independently of the microflake distribution, for instance when the microflake distribution is isotropic as shown in Fig. 12.1a. However, we found in our experiments that using anisotropic self-shadowing does not improve significantly the accuracy in practice, even for datasets with dense aligned voxels such as the hairy bunny (Fig. 14.4b). This means that our simplified model has sufficient accuracy in most downsampling applications, and that accuracy is more limited by albedos and phase functions than by accurate view-dependent attenuation.

We compared our LODs with results provided by Zhao, Wu et al. [ZWDR16]. Fig. 14.6 and Fig. 14.7 show LODs of the dense hairy bunny and the velvet datasets. Both methods preserve large-scale albedos, unlike naïve methods (14.5, 14.4b). Silhouettes of our LODs are more accurate because we preserve the local transparency in low-resolution voxels instead of prefiltering density parameters linearly.

14.2.4 Limitations

The main limitation of our work is that the self-shadowing functions we use do not model accurately self-shadowing effects in dense voxels. Indeed, our functions are symmetric ($A(\omega) = A(-\omega)$) while self-shadowing in a dense voxel is very strong forward and much smaller backward. Moreover, our multiple

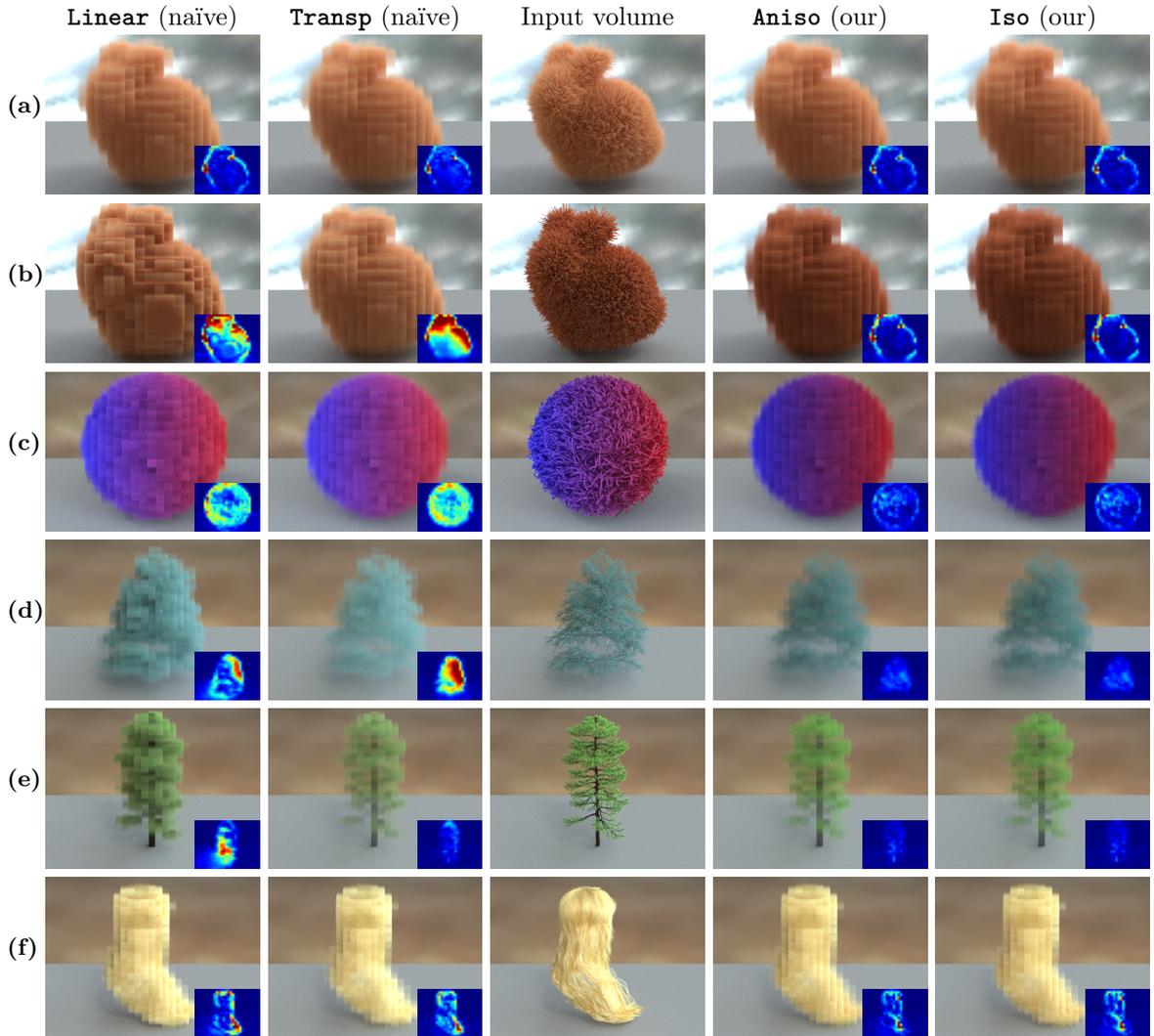


Figure 14.4: Comparison of volumes downsampled using the four algorithms described in Sec. 14.1. Low-res volumes are LOD 5, meaning that the number of voxel has been divided by 8^3 compared to input volumes. Insets show relative L_1 errors in linear RGB, computed on low-resolution pictures. (a): low-density bunny. (b): high density bunny ($\times 20$ compared to (a)). (c): dense hairy ball with anisotropic microflake distributions. (d): cedar foliage with homogeneous albedo. (e): pine tree with dark trunk. (f): hair with anisotropic microflake distribution. Our algorithm increases significantly accuracy of LODs when input volumes have dense voxels ((b) to (e)).

scattering phase function is relatively diffuse, while light scattered multiple times in a dense voxel mainly leaves this voxel backward — because light cannot reach the opposite side of the voxel. Hence, our phase functions lack accuracy when input datasets are very dense and heterogeneous. This can be seen in Figure 14.7 where the LOD from [ZWDR16] better preserves velvet-like reflections at grazing angles because they use dense low-resolution voxels. Fig. 14.8 shows that our LODs lack accuracy for the hair datasets because input voxels are too dense. Our self-shadowing model allows for preserving the mean amount of local self-shadowing, but preserving both the local transparency and an accurate scattering behavior remains an open problem for dense and heterogeneous input volumes.

We did not address the problem of colored multi-fiber datasets, for which it is important to store multiple albedos and lobes in low-resolution voxels. We believe that our method can be extended to such case using self-shadowing estimations for each lobe, but a rigorous study remains to be done. We did not work on animated datasets with time-varying scattering parameters in input voxels. Precomputing low-resolution volumes at each time step would reduce the benefits of using LODs in some applications.

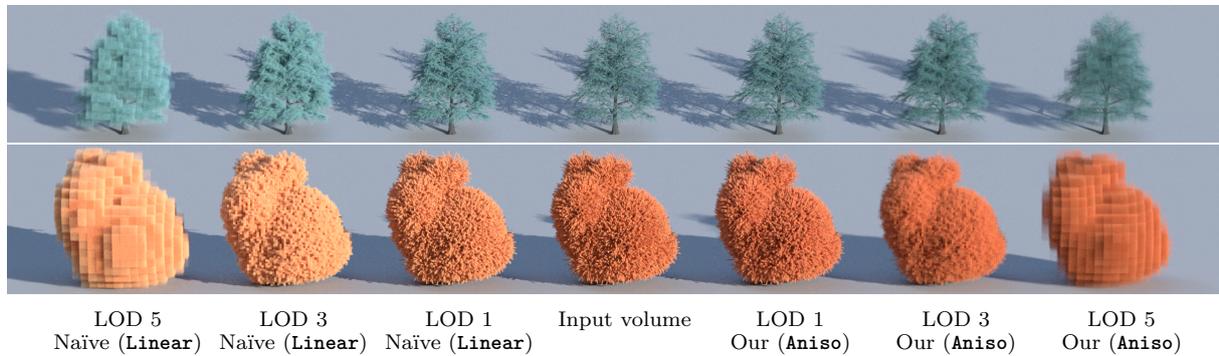


Figure 14.5: Comparison between naïve downsampling of microflake volumes and our method (**Aniso**). Rendered without trilinear interpolation.

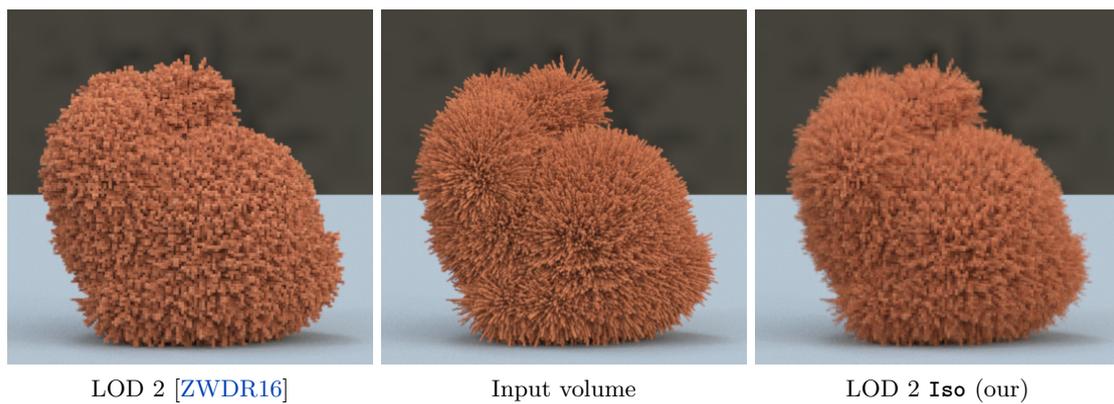


Figure 14.6: Dense hairy bunny. Both methods preserve the macroscopic appearance of the input volume. Silhouettes are slightly more accurate in our LOD because we do not prefilter density linearly.

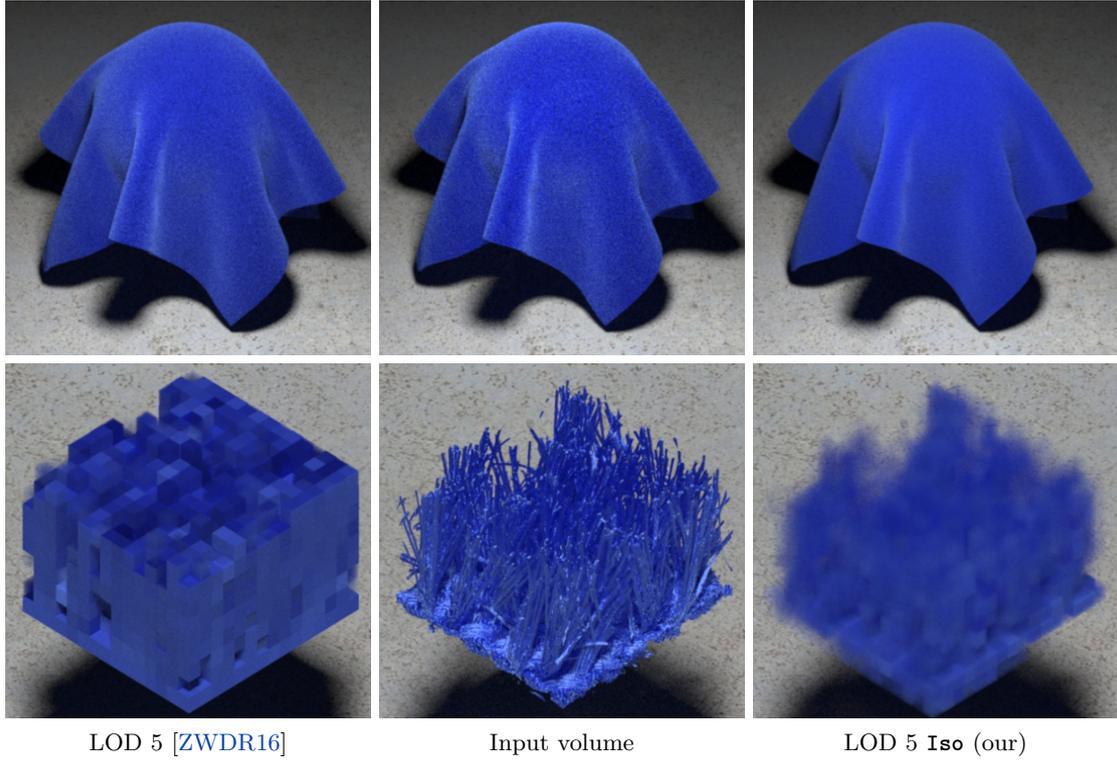


Figure 14.7: Close-up vs large-scale view of the velvet dataset. Our LODs have accurate transparency (bottom right). Because they have dense low-resolution voxels (top left), LODs from [ZWDR16] preserve more accurately fabrics-like appearance at large scales, especially at grazing angles. Preserving both the transparency and the scattering behavior of the input volume remains an open problem.

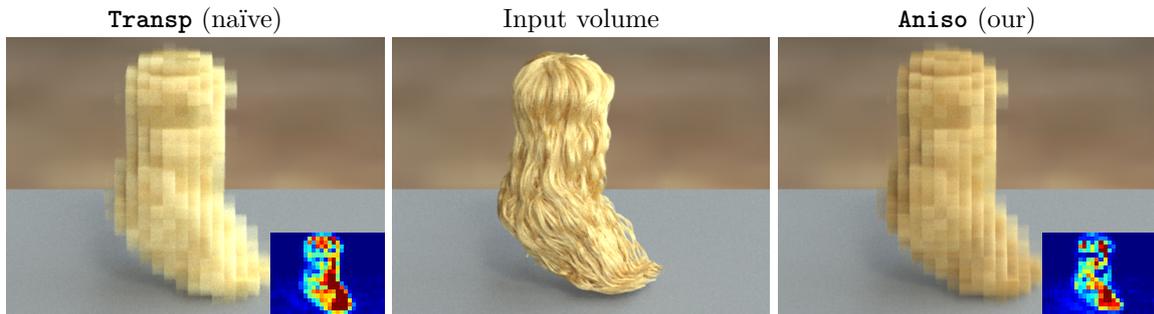


Figure 14.8: Same hair dataset as in Fig. 14.4f, with density multiplied by 10. This is a challenging case because of very anisotropic microflake distributions and complex self-shadowing effects due to dense input voxels. Our LODs are slightly better than naïve methods for this datasets but lacks accuracy (loss of bright anisotropic reflections). This is because our self-shadowing model does not preserve well appearance of very dense voxels (Sec. 14.2.4).

14.3 Summary of the chapter

We introduced two new downsampling algorithms that rely on our microflake model (Chap. 12). The most accurate algorithm is based on anisotropic self-shadowing functions and can be implemented using trigonometric distributions as discussed in chapter 13. The second algorithm relies on our simplified model (isotropic self-shadowing functions). These downsampling algorithms estimate parameters for low-resolution voxels corresponding to blocks of high-resolution voxels. The main steps of the algorithms for each block of voxels are the following:

- Microflakes normal distributions are prefiltered as in previous work [HDCD15].
- The anisotropic occlusion of the block of voxels is estimated. Then, parameters for the corresponding low-resolution voxel are obtained such that the local transparency of the medium is preserved.
- The self-shadowing parameters of the low-resolution voxel are estimated based on numerical estimations of the self-shadowing probabilities in the block of voxels.
- Finally, a multiple scattering albedo is estimated for the low-resolution voxel, such that the average effective albedo is preserved.

From our results, we can draw several conclusions:

- First, our anisotropic self-shadowing model and our simplified model produce LODs with similar quality for all our test assets. Our simplified model should be used by default because it requires less parameters and can be rendered more efficiently.
- This means that accurate anisotropic occlusion in low-resolution voxels is not necessary in practice, it is sufficient to preserve the average occlusion of the block of input voxels. To the contrary, the quality of LODs highly depends on the accuracy of the effective albedos and phase functions.
- We show that estimating scattering parameters locally is efficient and accurate. We compare our results with previous work based on image-space optimization and we show that our method generates LODs with similar quality.
- The main limitation of our work is that our model is still unable to preserve accurately the effective phase functions. This limits the quality of our LODs in some cases. Previous methods also fail in preserving at the same time the occlusion, the effective albedos and the effective phase functions in low-resolution volumes and this remains an open problem.
- We did not extend our method to the case of multi-lobe microflake distributions, and downsampling such media with our method would be an interesting direction for future work.

14.4 Résumé du chapitre

Nous avons introduit deux nouveaux algorithmes pour le préfiltrage de volumes de microflakes (Chap. 12). L'algorithme le plus précis utilise des fonctions d'ombrage local anisotropes et peut être implémenté avec les distributions trigonométriques présentées dans le chapitre 13. Le deuxième algorithme repose sur notre modèle simplifié et des fonctions d'ombrage isotropes. Ces algorithmes estiment des paramètres de réflectance pour chaque voxel de basse résolution correspondant à un bloc de voxels de haute résolution. Les principales étapes de ces algorithmes pour chaque bloc de voxels sont les suivantes :

- Les distributions de normales de microflakes sont préfiltrées comme dans les travaux précédents [HDCD15].
- L'occlusion anisotrope du bloc de voxels est mesurée. Ensuite, des paramètres pour le voxel de basse résolution correspondant sont obtenus de manière à conserver la transparence locale du milieu.

- Des paramètres d'ombrage sont estimés à partir d'estimations numériques des probabilités d'ombrage dans le bloc de voxels.
- Finalement, un albedo de diffusion multiple est estimé pour le voxel de basse résolution, de manière à ce que l'apparence moyenne du milieu soit préservée.

A partir des résultats obtenus, nous pouvons tirer les conclusions suivantes :

- Tout d'abord, nos modèles d'ombrage anisotrope et isotrope génèrent des LODs de qualité similaire pour toutes nos scènes de test. Notre modèle isotrope devrait être utilisé par défaut puisqu'il requiert moins de paramètres et qu'il peut être rendu plus efficacement.
- Nous en concluons que préserver précisément la transparence anisotrope d'un milieu n'est pas nécessaire en pratique, il est suffisant de préserver la transparence moyenne du bloc de voxels. Au contraire, la qualité des LODs dépend grandement de la précision des albedos effectifs et des fonctions de phase.
- Nous montrons qu'estimer des paramètres de réflectance localement est à la fois efficace et précis. Nous comparons nos résultats avec une méthode existante qui repose sur de l'optimisation à base d'erreurs en espace image et nous montrons que notre méthode produit des résultats de qualité similaire.
- La principale limitation de notre méthode est que notre modèle ne peut pas représenter très précisément les fonctions de phases effectives des blocs de voxels. Cela limite la qualité de nos LODs dans certains cas. Les méthodes précédentes échouent également à préserver à la fois la transparence et l'apparence des milieux et cela reste problématique.
- Nous n'avons pas travaillé sur le cas des tissus multi-fibres pour lesquels plusieurs lobes sont nécessaires par voxels. Préfiltrer de tels milieux avec notre modèle est une piste intéressante pour des travaux futurs.

Part IV

Conclusion

Chapter 15

Conclusion

15.1 Important ideas

In this thesis, we have explained why prefiltering 3D assets is an important problem for the efficiency of production rendering pipelines, and for efficient rendering in general: it allows for reducing memory usage, loading times, the cost of ray intersections, the cost of shading the materials and the noise in rendered images. We highlighted some important requirements of LOD methods for their use in production pipelines such as accuracy, robustness, drastic reduction of memory usage, seamless transitions and the use of standard rendering primitives.

Our main conclusions about our hybrid LOD approach (Part II) were the following:

- We used a volumetric model for prefiltering distant complex geometry such as foliage. We showed that volumes are memory efficient and appropriate for representing sub-resolution intricate geometry.
- Seamless transitions from meshes to volumes are possible and allow for combining the strengths of these two representations for LOD rendering. In our all-scale hybrid approach, each prefiltering method is used in its domain of validity.
- We showed that the separation of macrosurfaces and sub-resolution intricate geometry can be done automatically using a mesh analysis. This is a first step towards automatic and robust prefiltering of arbitrary assets.
- We showed that it is possible to prefilter the materials of surfaces during the geometric simplification without any surface parametrization. This enlarges the scope of material prefiltering techniques that were limited to texture space.

In Part III, we presented our contributions on downsampling high-resolution heterogeneous volumes. Our main conclusions were the following:

- It is very important that low-resolution volumes have a correct local transparency. We explained why linear density downsampling has insufficient accuracy, and we proposed an alternative method that generates better results, with correct silhouettes and shadows.
- A correct average transparency in low-resolution volumes is sufficient: preserving accurately the local anisotropic occlusion of the input volume does not improve the quality of the LODs according to our experiments.
- We showed that scattering parameters and in particular albedos can be optimized locally, instead of global optimizations based on image-space errors as in previous work. Local estimation of parameters is much faster and it allows for downsampling very easily spatially-varying volumes, which was a difficult case for previous methods.

- We extended the microflake model with additional parameters that are useful in downsampling applications. These parameters allow for controlling independently the microflake distribution, the anisotropic occlusion in the medium and the amount of self-shadowing, which plays an important role in the apparent color of the medium.

15.2 Open problems

Prefiltering complex assets is a long-term research problem and it is far from being solved entirely. In this thesis, we investigated new strategies and proposed new tools that overcome some limitations of previous work. Nevertheless, many problems remain to be studied:

- Prefiltering methods in surface space are still rather limited: existing techniques mostly rely on either heavy bidirectional texture functions, normal maps prefiltering, some hypothesis of decorrelation between the geometry and the reflectance of the surface, or surfaces with specific micro-appearance. Very few methods address the problem of prefiltering accurately non-flat surfaces. Moreover, most methods focus on simple surface reflectance models (Lambertian BRDF, Phong lobes, specular microfacets models), whereas reflectance models used in production are complex combinations of base reflectance models which include layered models, subsurface scattering, transmission through the surface, emission, etc. Prefiltering production assets with such reflectance models is not well studied.
- As discussed in several chapters of this thesis, adding LOD methods to production pipelines raises many practical problems. Some of these problems are specific to each pipeline, but some problems such as asset overlapping or the efficient implementation of seamless transitions are more general and far from being entirely investigated.
- Animating LOD representations is a rather understudied problem. In this thesis, as in previous work on aggregate detail prefiltering, we relied on volumetric approximations that lack flexibility in the case of animated assets. In particular, distant plants and furry characters can be efficiently rendered using volumetric representations but they are often animated in production scenes, which limits the scope of current prefiltering methods or adds additional precomputation costs that should be studied.
- As an alternative to volume representations, we considered the use of a small number of sub-voxel polygons for representing sub-resolution volume-like appearances. This strategy seems promising to us for production environment, especially for animated and overlapping assets, but it remains to be investigated and evaluated with the same quality criteria: correct mean occlusion, effective albedos and effective phase functions. In particular, estimating and preserving the probability of local shadowing seems mandatory in order to avoid overly bright LODs and our work on prefiltering volumes would have to be adapted to this polygon-based representation.
- Our new volume model extends the range of available models for representing complex appearances. We believe that new volume models should be investigated for the accurate approximation of sub-resolution geometry. As demonstrated by Khungurn et al. [KSZ*15], deriving phase functions from fiber scattering models improves the accuracy of volumetric approximations for rendering fabrics. More work should be done on approximating hair and foliage in the same spirit, not only for computing multiple scattering but also for rendering distant assets without explicit geometry. For example, the microflake model could be used with transmissive microflakes for representing leaves more accurately.
- Glints on scratched surfaces, hair and granular materials such as snow have been recently investigated and rendered. Although the existing methods remain costly compared to materials, this work is an important step towards perfect photorealism. In this thesis, we always prefiltered normal distributions aggressively for ensuring low storage requirements and we did not attempt to preserve glints in our low-resolution assets. We believe that the support of sparkling appearances is an interesting future work, in particular for the accurate prefiltering of foliage with volumetric models.

We hope that this thesis will inspire future work on appearance-preserving prefiltering for improving the efficiency of rendering pipelines and the quality of rendered virtual worlds.

Chapter 16

Conclusion

16.1 Idées importantes

Dans cette thèse, nous avons expliqué en quoi le préfiltrage des objets 3D est un problème important pour l'efficacité du rendu dans les studios de production et de manière générale pour le rendu de haute qualité : le préfiltrage permet de réduire la taille mémoire des scènes, les temps de chargement, les coûts d'intersection des rayons lumineux, les coûts de calcul des matériaux et le bruit dans les images. Nous avons souligné certaines contraintes de qualité que doivent satisfaire les méthodes de niveaux de détail pour être utilisées en production, et plus particulièrement la fiabilité, la précision, l'efficacité mémoire, les transitions invisibles entre les différentes résolutions et l'usage de primitives de rendu standards.

Nos principales conclusions concernant notre travail sur les LOD hybrides (Partie II) ont été les suivantes :

- Nous avons utilisé des modèles volumiques pour préfiltrer et faire le rendu de géométries complexes comme du feuillage. Nous avons montré que les représentations volumiques sont efficaces du point de vue de la mémoire et appropriées pour le préfiltrage de certaines géométries complexes sous-résolution.
- Les transitions invisibles entre les maillages et les volumes sont possibles et permettent d'allier les avantages de ces deux représentations pour le rendu d'objets préfiltrés. Avec notre approche hybride, chaque représentation et technique de préfiltrage est utilisée dans son domaine de validité.
- Nous avons montré que la détection de macrosurfaces et de géométrie complexe sous-résolution peut être faite automatiquement à l'aide d'une analyse de maillage. Cette méthode est un premier pas vers le préfiltrage automatique et fiable des objets 3D complexes.
- Nous avons montré qu'il est possible de préfiltrer les matériaux des surfaces dont la géométrie est simplifiée par des algorithmes de simplification de maillage, le tout sans avoir besoin d'une paramétrisation de toute la surface. Cela permet d'étendre la portée de certaines méthodes de niveaux de détail qui étaient auparavant réservées à du préfiltrage en espace texture.

Dans la partie III, nous avons présenté nos contributions pour le préfiltrage des volumes hétérogènes de haute résolution. Nos principales conclusions ont été les suivantes :

- Il est très important que les volumes préfiltrés aient une transparence locale correcte. Nous avons montré pourquoi le filtrage linéaire des paramètres de densité n'est pas assez précis. Nous avons proposé une méthode alternative qui génère de meilleurs résultats, avec des silhouettes plus précises et des ombres correctes.
- Assurer une transparence moyenne précise est suffisant en terme de qualité : nos tests ont montré

que la conservation de la transparence anisotrope locale ne permet pas de gagner significativement en précision dans les volumes préfiltrés.

- Nous avons montré que les paramètres de réflectance et en particulier les albedos peuvent être optimisés localement, à la place de méthodes d'optimisation globales s'appuyant sur des erreurs en espace image comme c'était le cas dans les travaux précédents. L'estimation locale de paramètres est plus rapide et permet le préfiltrage des milieux qui varient spatialement, ce qui était auparavant un cas difficile.
- Nous avons étendu le modèle volumique de microflakes avec des paramètres supplémentaires qui sont utiles pour préfiltrer des volumes. Ces paramètres permettent de contrôler de manière indépendante la distribution de normales des microflakes, l'atténuation anisotrope du milieu, et la quantité d'ombrage local qui influence beaucoup la couleur perçue du milieu.

16.2 Questions ouvertes

Le préfiltrage des objets 3D complexes est un problème de recherche de longue haleine qui est encore loin d'être résolu entièrement. Dans cette thèse, nous avons proposé de nouvelles stratégies et de nouveaux outils qui permettent de dépasser certaines limitations des travaux précédents. Néanmoins, de nombreuses questions restent en suspens :

- Les méthodes de préfiltrage de l'apparence des surfaces sont toujours relativement limitées : les méthodes existantes reposent principalement sur des BTFs coûteuses, sur du préfiltrage de distributions de normales, sur des hypothèses fortes d'absence de corrélation entre la géométrie de la surface et sa réflectance, ou sur des modèles de microsursaces particuliers. Peu de méthodes permettent de préfiltrer précisément les surfaces non planes. De plus, la plupart des techniques existantes concernent des modèles de réflectance simples (modèle lambertien, lobes de Phong, BRDF à microfacettes spéculaires), alors que dans les studios de production les modèles utilisés sont des combinaisons complexes de matériaux de base qui peuvent inclure des modèles de réflectance multicouches, de la diffusion dans la surface, de la transmission à travers la surface, de l'émission, etc. Le préfiltrage de ces matériaux n'est pas encore bien étudié.
- Comme mentionné dans plusieurs chapitres, implémenter des méthodes de niveaux de détail dans les studios de production pose de nombreux problèmes pratiques. Certains de ces problèmes sont spécifiques à chaque studio, mais d'autres sont plus généraux, comme les problèmes liés au chevauchement d'objets voisins ou l'implémentation efficace des transitions invisibles entre les différentes résolutions. Ces problèmes pourraient être étudiés plus précisément.
- L'animation de représentations préfiltrées est un problème sous-étudié. Dans cette thèse, comme dans la majorité des travaux précédents sur le préfiltrage de géométrie complexe, nous avons utilisé des représentations volumiques qui sont assez peu adaptées aux objets animés. En particulier, les plantes et les personnages à fourrure peuvent être efficacement rendus avec des volumes mais ils sont souvent animés dans les scènes de production, ce qui limite la portée des méthodes de préfiltrage ou ajoute des surcoûts de précalcul. Cela devrait être étudié plus en profondeur.
- Nous avons commencé à étudier l'utilisation de quelques polygones sous-résolution comme représentation alternative aux volumes pour préfiltrer de la géométrie complexe. Cette stratégie semble prometteuse pour les studios de production, en particulier pour les objets animés ou se chevauchant. Il reste à étudier la qualité de cette solution pour la conservation des probabilités d'occlusion et de l'apparence. L'estimation et la conservation des probabilités d'ombrage local semble nécessaire pour éviter de générer des objets préfiltrés trop clairs, et nos travaux sur le filtrage de volumes devons être adaptés à cette représentation alternative.
- Le nouveau modèle de volume que nous avons introduit étend le nombre de modèles disponibles pour le préfiltrage d'apparences complexes. Nous pensons que de nouveaux modèles devraient être étudiés et s'ajouter au notre et accroître la portée et la précision du préfiltrage par des volumes. Comme l'ont montré Khungurn et al. [KSZ*15], utiliser des fonctions de phase issues des modèles de réflexion de fibres permet d'améliorer la qualité du rendu de tissus. Une étude plus approfondie

pourrait être menée dans le même esprit sur l'approximation volumique de cheveux et de feuillages distants. Par exemple, dans le cas des feuilles, le modèle à microflakes pourrait être utilisé avec des lobes de transmission pour un préfiltrage plus précis.

- Le scintillement des surfaces rayées, des cheveux et des matériaux granulaires comme la neige a été récemment étudié et rendu. Bien que les techniques actuelles sont onéreuses comparées aux autres modèles de matériaux, elles sont un pas en avant vers le photoréalisme parfait des images rendues. Dans cette thèse, nous avons toujours préfiltré les distributions de normales de manière agressive pour s'assurer de réduire la taille mémoire des objets, et nous n'avons pas essayé de conserver des apparences scintillantes dans les objets préfiltrés. Nous pensons qu'il s'agit d'une piste intéressante pour des travaux futurs, en particulier pour le préfiltrage des feuillages avec des représentations volumiques.

Nous espérons que cette thèse inspirera d'autres travaux sur le préfiltrage des apparences complexes pour l'amélioration de l'efficacité des studios de production et la qualité des mondes virtuels rendus.

Appendices

A Macrosurface analysis: finding the topology of I

The topology of I , as defined in chapter 7, can be computed with a mesh exploration, starting from the face containing the point x . We make sure that only triangles connected to x *inside* S are explored. We do not need the exact shape of I but just its topology.

```
Function IsOnMacroSurface (inputTri, sphere)
```

```
  l = ∅;  
  trisToExplore = inputTri;  
  exploredTris = ∅;  
  while not IsVoid (trisToExplore) do  
    currentTri = PopFirst (trisToExplore);  
    Explore (currentTri, l, exploredTris);  
    Add (currentTri, exploredTris);  
  end  
  return HasSingleConnectedComponent (l);
```

```
End
```

```
Function Explore (currentTri, l, exploredTris, trisToExplore, sphere)
```

```
  for e in edges of currentTri do  
    if e intersect sphere or e is inside sphere then  
      adjTri = triangle adjacent to currentTri at e;  
      if adjTri is not in exploredTris and adjTri is not in trisToExplore then  
        Add (trisToExplore, adjTri);  
      end  
    end  
  end  
  /* sphere ∩ currentTri could be 1, 2 or 3 sections of a circle, or void. Only the  
     ends of each part is needed, because we just need the topology of l.      */  
  its = ComputeIntersections (currentTri, sphere);  
  Add (its, l);
```

```
End
```

B Rejection sampling

Probability distribution functions (*pdf*) are ubiquitous in physically-based rendering, and procedures that generate samples from such distribution are often needed. In particular, generating samples from phase functions tends to decrease noise in volume path tracing.

Unfortunately, it is not always possible to find an efficient sampling procedure for given a *pdf*. The most common method, known as *inverse transform sampling*, requires an expression of the inverse F^{-1} of the cumulative distribution function of the *pdf*:

$$F(u) = \int_{x=-\infty}^u pdf(x) dx,$$

but it is not always possible to find closed-form expressions for F^{-1} . Some authors have found sampling procedures using the fact that their *pdf* comes from a geometric problem – for instance, sampling visible normals of an ellipsoid viewed from a given direction [HDCD15, DWMG15]. This strategy cannot be applied to all *pdfs*.

Rejection sampling is a very general method that can generate samples from any normalized *pdf*. For sampling the distribution pdf_1 , it is possible to generate samples from any other distribution pdf_2 and to accept or reject these samples with a probability such that accepted samples follow exactly the distribution pdf_1 . Rejection sampling is very powerful in the sense that it allows for sampling arbitrary distributions, but the algorithm is only efficient if pdf_2 is *close enough* to pdf_1 . The more pdf_2 is different from pdf_1 , the more samples will be rejected by the algorithm, leading to a very inefficient sampling procedure. For sampling pdf_1 using pdf_2 , rejection sampling needs a scalar M such that, for all x ,

$$pdf_1(x) \leq M pdf_2(x)$$

Then, the sampling procedure is the following:

1. Generate a sample x from pdf_2 .
2. Generate a random number U in $(0,1)$.
If $U \leq \frac{pdf_1(x)}{M pdf_2(x)}$, accept the sample x (end of the procedure).
Else, go back to step 1.

The value M is the average number of iterations needed before accepting one sample, and $1/M$ is the average probability of accepting one sample from pdf_2 . It is thus possible to predict how efficient is a sampling procedure – in average.

C Derivation of general closed-form expressions for $\sigma_t(\omega)$

In the case of a \cos^{2n} distribution, $\sigma_t(\omega)$ is given by

$$\sigma_t(\omega) = \rho A(\omega) \int D_{\cos}(m, \xi_D, n) \langle m \cdot \omega \rangle dm = \rho \frac{A(\omega)}{N_{\cos}(n)} \int (m \cdot \xi_D)^{2n} \langle m \cdot \omega \rangle dm.$$

Without loss of generality, we choose a convenient spherical parametrization such that $\omega = [\cos(\phi_i), \sin(\phi_i), 0]$ and $\xi_D = [1, 0, 0]$. We have:

$$\begin{aligned} \sigma_t(\omega) &= \rho \frac{A(\omega)}{N_{\cos}(n)} \int \left(\begin{bmatrix} \cos(\phi) \sin(\theta) \\ \sin(\phi) \sin(\theta) \\ \cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right)^{2n} \left| \begin{bmatrix} \cos(\phi) \sin(\theta) \\ \sin(\phi) \sin(\theta) \\ \cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} \cos(\phi_i) \\ \sin(\phi_i) \\ 0 \end{bmatrix} \right| \sin(\theta) d\phi d\theta \\ &= \rho \frac{A(\omega)}{N_{\cos}(n)} \int_{\phi=-\pi/2-\phi_i}^{\pi/2-\phi_i} \int_{\theta=0}^{\pi} (\cos(\phi) \sin(\theta))^{2n} (\cos(\phi) \sin(\theta) \cos(\phi_i) + \sin(\phi) \sin(\theta) \sin(\phi_i)) \sin(\theta) d\phi d\theta \\ &= \rho \frac{A(\omega)}{N_{\cos}(n)} \int_{\phi=-\pi/2-\phi_i}^{\pi/2-\phi_i} \int_{\theta=0}^{\pi} \cos(\phi)^{2n} (\cos(\phi) \cos(\phi_i) + \sin(\phi) \sin(\phi_i)) \sin(\theta)^{2n+2} d\phi d\theta \\ &= \rho \frac{A(\omega)}{N_{\cos}(n)} \frac{\sqrt{\pi} \Gamma(\frac{3}{2} + n)}{\Gamma(2 + n)} \int_{\phi=-\pi/2-\phi_i}^{\pi/2-\phi_i} \cos(\phi)^{2n} (\cos(\phi) \cos(\phi_i) + \sin(\phi) \sin(\phi_i)) d\phi \\ &= \rho \frac{A(\omega)}{N_{\cos}(n)} \frac{\sqrt{\pi} \Gamma(\frac{3}{2} + n)}{\Gamma(2 + n)} \left(\cos(\phi_i) \int_{\phi=-\pi/2-\phi_i}^{\pi/2-\phi_i} \cos(\phi)^{2n+1} d\phi + \sin(\phi_i) \int_{\phi=-\pi/2-\phi_i}^{\pi/2-\phi_i} \cos(\phi)^{2n} \sin(\phi) d\phi \right) \\ &= \rho \frac{A(\omega)}{N_{\cos}(n)} \frac{\sqrt{\pi} \Gamma(\frac{3}{2} + n)}{\Gamma(2 + n)} \left(\cos(\phi_i) \int_{\phi=-\pi/2-\phi_i}^{\pi/2-\phi_i} \cos(\phi)^{2n+1} d\phi - \frac{2 \sin(\phi_i)^{2n+2}}{2n+1} \right). \end{aligned} \quad (C.1)$$

Now the integral $\int \cos(x)^m dx$ can be written as a finite sum using

$$\int \cos(x)^m dx = \frac{1}{m} \cos(x)^{m-1} \sin(x) + \frac{m+1}{m} \int \cos(x)^{m-2} dx.$$

It is also possible to write it with the Gauss hypergeometric function. Given that $0 \leq \phi_i \leq \pi/2$, on the interval $(-\frac{\pi}{2} - \phi_i, -\frac{\pi}{2})$ we perform a change of variable with $X = \cos(\phi)$, and we obtain

$$\int_{\phi=-\pi/2-\phi_i}^{-\pi/2} \cos(\phi)^{2n+1} d\phi = \int_{X=\cos(-\pi/2-\phi_i)}^0 \frac{X^{2n+1}}{\sqrt{1-X^2}} dX \quad (C.2)$$

$$= \int_{X=-\sin(\phi_i)}^0 \frac{X^{2n+1}}{\sqrt{1-X^2}} dX. \quad (C.3)$$

Using the change of variable $Y = \frac{X^2}{-\sin(\phi_i)}$, with $\left\| \frac{\partial X}{\partial Y} \right\| = \frac{\sin(\phi_i)}{2\sqrt{Y}}$ we get

$$\int_{X=-\sin(\phi_i)}^0 \frac{X^{2n+1}}{\sqrt{1-X^2}} dX = \int_0^1 \frac{Y^{n+\frac{1}{2}} \sin(\phi_i)^{2n+1} \sin(\phi_i)}{\sqrt{1-Y \sin(\phi_i)^2} 2\sqrt{Y}} dY \quad (C.4)$$

$$= -\frac{\sin(\phi_i)^{2n+2}}{2} \int_1^0 \frac{Y^n}{\sqrt{1-Y \sin(\phi_i)^2}} dY. \quad (C.5)$$

Using

$$\frac{\Gamma(b)\Gamma(c-b)}{\Gamma(c)} {}_2F_1(a, b, c, z) = \int_0^1 \frac{t^{b-1}(1-t)^{c-b-1}}{(1-tz)^a} dt$$

we can write

$$-\frac{\sin(\phi_i)^{2n+2}}{2} \int_1^0 \frac{Y^{n+\frac{1}{2}}}{\sqrt{1-Y\sin(\phi_i)^2}} dY = -\frac{\sin(\phi_i)^{2n+2}}{2} \frac{\Gamma(b)\Gamma(c-b)}{\Gamma(c)} {}_2F_1(a, b, c, z) \quad (\text{C.6})$$

with $a = \frac{1}{2}$, $b = n + 1$, $c = n + 2$ and $z = \sin(\phi_i)^2$. We have

$$\int_{\phi=-\pi/2-\phi_i}^{-\pi/2} \cos(\phi)^{2n+1} d\phi = -\frac{\sin(\phi_i)^{2n+2}}{2} \frac{\Gamma(n+1)\Gamma(1)}{\Gamma(n+2)} {}_2F_1\left(\frac{1}{2}, n+1, n+2, \sin(\phi_i)^2\right) \quad (\text{C.7})$$

$$= -\frac{\sin(\phi_i)^{2n+2}}{2n+2} {}_2F_1\left(\frac{1}{2}, n+1, n+2, \sin(\phi_i)^2\right). \quad (\text{C.8})$$

Now, we integrate between $-\pi/2$ and 0:

$$\int_{\phi=-\pi/2}^0 \cos(\phi)^{2n+1} d\phi = \int_0^1 \frac{X^{2n+1}}{\sqrt{1-X^2}} dX \quad (\text{C.9})$$

$$= \int_0^1 \frac{Y^{n+\frac{1}{2}}}{\sqrt{1-Y}} \frac{1}{2\sqrt{Y}} dY \quad (\text{C.10})$$

$$= \frac{1}{2} \int_0^1 \frac{Y^n}{\sqrt{1-Y}} dY \quad (\text{C.11})$$

$$= \frac{\Gamma(b)\Gamma(c-b)}{2\Gamma(c)} {}_2F_1(a, b, c, 1) \quad (\text{C.12})$$

with $a = \frac{1}{2}$, $b = n + 1$, $c = n + 2$. We have

$$\int_{\phi=-\pi/2}^0 \cos(\phi)^{2n+1} d\phi = \frac{1}{2} B\left(\frac{1}{2}, n+1\right) \quad (\text{C.13})$$

with B the beta function¹. Finally, using the same approach,

$$\int_{\phi=0}^{\pi/2-\phi_i} \cos(\phi)^{2n+1} d\phi = \int_{\phi=0}^{\pi/2} \cos(\phi)^{2n+1} d\phi + \int_{\phi=\pi/2}^{\pi/2-\phi_i} \cos(\phi)^{2n+1} d\phi \quad (\text{C.14})$$

$$= \int_1^0 \frac{X^{2n+1}}{-\sqrt{1-X^2}} dX + \int_0^{\cos(\pi/2-\phi_i)} \frac{X^{2n+1}}{-\sqrt{1-X^2}} dX \quad (\text{C.15})$$

$$= \frac{1}{2n+2} B\left(\frac{1}{2}, n+1\right) - \frac{\sin(\phi_i)^{2n+2}}{2n+2} {}_2F_1\left(\frac{1}{2}, n+1, n+2, \sin(\phi_i)^2\right). \quad (\text{C.16})$$

We can add the three terms and get

$$\int_{\phi=-\pi/2-\phi_i}^{\pi/2-\phi_i} \cos(\phi)^{2n+1} d\phi = B\left(\frac{1}{2}, n+1\right) - \frac{\sin(\phi_i)^{2n+2}}{n+1} {}_2F_1\left(\frac{1}{2}, n+1, n+2, \sin(\phi_i)^2\right). \quad (\text{C.17})$$

¹https://en.wikipedia.org/wiki/Beta_function

We proved that

$$\sigma_i(\omega) = \rho \frac{A(\omega)}{N_{\cos}(n)} \frac{\sqrt{\pi} \Gamma\left(\frac{3}{2} + n\right)}{\Gamma(2 + n)} \left(\cos(\phi_i) \left(B\left(\frac{1}{2}, n + 1\right) - \frac{\sin(\phi_i)^{2n+2}}{n + 1} {}_2F_1\left(\frac{1}{2}, n + 1, n + 2, \sin(\phi_i)^2\right) \right) - \frac{2 \sin(\phi_i)^{2n+2}}{2n + 1} \right).$$

(C.18)

As explained before, we don't use this expression in our implementation because more efficient expressions can be found for each particular value of n , when n is small.

D Tabulated values of $\int \sqrt{\omega^T S \omega}$

| $\alpha_2 \setminus \alpha_3$ | 0.05 | 0.1 | 0.15 | 0.2 | 0.25 | 0.3 | 0.35 | 0.4 | 0.45 | 0.5 | 0.55 | 0.6 | 0.65 | 0.7 | 0.75 | 0.8 | 0.85 | 0.9 | 0.95 | 1.0 | |
|-------------------------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|--|
| 0.05 | 6.9847E | | | | | | | | | | | | | | | | | | | | |
| 0.1 | 7.24191 | 7.48864 | | | | | | | | | | | | | | | | | | | |
| 0.15 | 7.4704E | 7.7068E | 7.92133 | | | | | | | | | | | | | | | | | | |
| 0.2 | 7.6792E | 7.9089E | 8.1193 | 8.31161 | | | | | | | | | | | | | | | | | |
| 0.25 | 7.8758E | 8.10067 | 8.3039E | 8.4932E | 8.6726E | | | | | | | | | | | | | | | | |
| 0.3 | 8.0621E | 8.2800E | 8.48044 | 8.66547 | 8.8418E | 9.0083 | | | | | | | | | | | | | | | |
| 0.35 | 8.2367 | 8.4513E | 8.6477E | 8.8310E | 9.00371 | 9.16961 | 9.32714 | | | | | | | | | | | | | | |
| 0.4 | 8.4059 | 8.6167E | 8.8094E | 8.99007 | 9.1601 | 9.3241E | 9.4798E | 9.6308E | | | | | | | | | | | | | |
| 0.45 | 8.5678E | 8.7760E | 8.96504 | 9.1433E | 9.3114E | 9.4728E | 9.62734 | 9.7765E | 9.9207E | | | | | | | | | | | | |
| 0.5 | 8.7241 | 8.9295E | 9.1162 | 9.29157 | 9.4584E | 9.6168E | 9.76871 | 9.9177E | 10.061E | 10.199E | | | | | | | | | | | |
| 0.55 | 8.8738E | 9.07574 | 9.2619E | 9.4387E | 9.6012E | 9.758 | 9.90851 | 10.0557 | 10.198E | 10.3334 | 10.4677 | | | | | | | | | | |
| 0.6 | 9.0211E | 9.2217E | 9.40381 | 9.5761E | 9.7371E | 9.8940E | 10.0441 | 10.189 | 10.3284 | 10.464E | 10.597E | 10.728E | | | | | | | | | |
| 0.65 | 9.1663E | 9.36294 | 9.54264 | 9.7123E | 9.87291 | 10.027E | 10.176E | 10.319E | 10.459 | 10.5934 | 10.724E | 10.852E | 10.978E | | | | | | | | |
| 0.7 | 9.30461 | 9.49 | 9.6785E | 9.8452E | 10.0051 | 10.157E | 10.304E | 10.447E | 10.5851 | 10.718E | 10.848E | 10.976E | 11.101E | 11.2231 | | | | | | | |
| 0.75 | 9.44041 | 9.6315E | 9.80971 | 9.9749 | 10.133 | 10.288E | 10.431 | 10.572 | 10.708E | 10.841E | 10.971E | 11.097E | 11.220E | 11.341E | 11.462E | | | | | | |
| 0.8 | 9.57287 | 9.76284 | 9.9390E | 10.102 | 10.26 | 10.409E | 10.554E | 10.693E | 10.8301 | 10.9622 | 11.0911 | 11.215E | 11.3384 | 11.458E | 11.576E | 11.691E | | | | | |
| 0.85 | 9.70291 | 9.89037 | 10.0654 | 10.2271 | 10.3821 | 10.5327 | 10.675 | 10.814E | 10.9487 | 11.0797 | 11.207E | 11.332E | 11.454E | 11.572E | 11.690E | 11.805 | 11.917E | | | | |
| 0.9 | 9.8239 | 10.015E | 10.188E | 10.349E | 10.5037 | 10.651E | 10.794E | 10.931E | 11.065E | 11.195E | 11.322E | 11.446E | 11.567E | 11.6861 | 11.802E | 11.9167 | 12.028E | 12.1387 | | | |
| 0.95 | 9.9553E | 10.1387 | 10.309E | 10.4694 | 10.6227 | 10.7694 | 10.910E | 11.047E | 11.1807 | 11.3094 | 11.436E | 11.558E | 11.679 | 11.7971 | 11.912E | 12.028E | 12.137E | 12.2471 | 12.354E | | |
| 1.0 | 10.076E | 10.258E | 10.4287 | 10.587E | 10.7391 | 10.8857 | 11.028E | 11.162 | 11.293 | 11.4217 | 11.546E | 11.669E | 11.789E | 11.908E | 12.020E | 12.133E | 12.244E | 12.3537 | 12.460E | 12.5664 | |

Table 1: Precomputed values for $\int \sqrt{\omega^T S \omega}$ for eigenvalues of S being 1, a_2 and a_3 .

Bibliography

- [AA03] ADAMSON A., ALEXA M.: Ray tracing point set surfaces. In *2003 Shape Modeling International*. (May 2003), pp. 272–279.
- [AA04] ADAMSON A., ALEXA M.: Approximating bounded, nonorientable surfaces from points. In *Proceedings Shape Modeling Applications, 2004*. (June 2004), pp. 243–252.
- [ABCO*01] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Point set surfaces. In *Proceedings of the Conference on Visualization '01* (Washington, DC, USA, 2001), VIS '01, IEEE Computer Society, pp. 21–28.
- [AFFC16] ANDERSEN T. G., FALSTER V., FRISVAD J. R., CHRISTENSEN N. J.: Hybrid fur rendering: combining volumetric fur with explicit hair strands. *The Visual Computer* 32, 6 (Jun 2016), 739–749.
- [AGP*04] ALEXA M., GROSS M., PAULY M., PFISTER H., STAMMINGER M., ZWICKER M.: Point-based computer graphics. In *ACM SIGGRAPH 2004 Course Notes* (New York, NY, USA, 2004), SIGGRAPH '04, ACM.
- [AK04] AMENTA N., KIL Y. J.: Defining point-set surfaces. In *ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), SIGGRAPH '04, ACM, pp. 264–270.
- [ALSS99] ALLIEZ P., LAURENT N., SANSON H., SCHMITT F.: Mesh approximation using a volume-based metric. In *Proceedings of the 7th Pacific Conference on Computer Graphics and Applications* (1999), IEEE Computer Society, pp. 292–.
- [ASCE02] ASPERT N., SANTA-CRUZ D., EBRAHIMI T.: MESH: measuring errors between surfaces using the hausdorff distance. In *Proceedings of the ICME '02* (2002), pp. 705–708 vol.1.
- [BA05] BÓO M., AMOR M.: High-performance architecture for anisotropic filtering. *J. Syst. Archit.* 51, 5 (May 2005), 297–314.
- [Bak11] BAKER D.: Spectacular specular — lean and clean specular highlights. In *Game Developer Conference* (2011).
- [BFK*16] BRAINERD W., FOLEY T., KRAEMER M., MORETON H., NIESSNER M.: Efficient gpu rendering of subdivision surfaces using adaptive quadtrees. *ACM Trans. Graph.* 35, 4 (July 2016), 113:1–113:12.
- [BL08] BURLEY B., LACEWELL D.: Ptex: Per-face texture mapping for production rendering. In *Proceedings of the Nineteenth Eurographics Conference on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2008), EGSR '08, Eurographics Association, pp. 1155–1164.
- [Bli77] BLINN J. F.: Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph.* 11, 2 (July 1977), 192–198.
- [Bli78] BLINN J. F.: Simulation of wrinkled surfaces. *SIGGRAPH Comput. Graph.* 12, 3 (Aug. 1978), 286–292.
- [Bli82] BLINN J.: Light reflection function for simulation of clouds and dusty surfaces. In *Siggraph 1982* (January 1982), Association for Computing Machinery, Inc.

- [BLP*12] BOMMES D., LÉVY B., PIETRONI N., PUPPO E., A C. S., TARINI M., ZORIN D.: State of the art in quad meshing. In *Eurographics STARS* (2012).
- [BM93] BECKER B. G., MAX N. L.: Smooth transitions between bump rendering algorithms. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1993), SIGGRAPH '93, ACM, pp. 183–190.
- [BN12] BRUNETON E., NEYRET F.: A survey of nonlinear prefiltering methods for efficient and accurate surface shading. *IEEE Transactions on Visualization and Computer Graphics* 18, 2 (Feb. 2012), 242–260.
- [BRS96] BAJAJ C., R. SCHIKORE D.: Error-bounded reduction of triangle meshes with multivariate data. vol. 2656, pp. 2656 – 2656 – 12.
- [BSBK02] BOTSCH M., STEINBERG S., BISCHOFF S., KOBBELT L.: Openmesh - a generic and efficient polygon mesh data structure. In *OpenSG Symposium* (2002).
- [Bun05] BUNNELL M.: *Dynamic Ambient Occlusion and Indirect Lighting*, vol. 2. 01 2005.
- [Bur12] BURLEY B.: Physically-based shading at disney. In *ACM SIGGRAPH 2012 Courses* (New York, NY, USA, 2012), ACM, pp. 1–7.
- [Cat74] CATMULL E.: *A subdivision algorithm for computer display of curved surfaces*. PhD thesis, University of Utah, Dec. 1974.
- [CBTB16] CHIANG M. J.-Y., BITTERLI B., TAPPAN C., BURLEY B.: A practical and controllable hair and fur model for production path tracing. *Computer Graphics Forum* 35, 2 (2016), 275–283.
- [CC78] CATMULL E., CLARK J.: Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design* 10, 6 (1978), 350 – 355.
- [CCC87] COOK R. L., CARPENTER L., CATMULL E.: The reyes image rendering architecture. *SIGGRAPH Comput. Graph.* 21, 4 (Aug. 1987), 95–102.
- [CCL14] CALUDE C. S., COULL A., LEWIS J. P.: Can we solve the pipeline problem? In *Proceedings of the Fourth Symposium on Digital Production* (New York, NY, USA, 2014), DigiPro '14, ACM, pp. 25–27.
- [CGA16] *CGAL User and Reference Manual*, 4.9 ed. CGAL Editorial Board, 2016. <http://doc.cgal.org/4.9/Manual/packages.html>.
- [Cha50] CHANDRASEKHAR S.: *Radiative Transfer*. Oxford Univ. Press, 1950.
- [CHPR07] COOK R. L., HALSTEAD J., PLANCK M., RYU D.: Stochastic simplification of aggregate detail. *ACM Trans. Graph.* (2007).
- [Chr10] CHRISTENSEN P. H.: Global illumination across industries point-based global illumination for movie production. *SIGGRAPH 2010 Course: Global Illumination Across Industries* (2010).
- [CKB16] CHIANG M. J.-Y., KUTZ P., BURLEY B.: Practical and controllable subsurface scattering for production path tracing. In *ACM SIGGRAPH 2016 Talks* (New York, NY, USA, 2016), SIGGRAPH '16, ACM, pp. 49:1–49:2.
- [Cla76] CLARK J. H.: Hierarchical geometric models for visible surface algorithms. *Commun. ACM* 19, 10 (Oct. 1976), 547–554.
- [CMO97] COHEN J., MANOCHA D., OLANO M.: Simplifying polygonal models using successive mappings. In *Visualization '97, Proceedings* (Oct 1997), pp. 395–402.
- [CMS98] CIGNONI P., MONTANI C., SCOPIGNO R.: A comparison of mesh simplification algorithms. *Computers & Graphics* 22, 1 (1998), 37 – 54.
- [CNS*11] CRASSIN C., NEYRET F., SAINZ M., GREEN S., EISEMANN E.: Interactive indirect illumination using voxel cone tracing. *Computer Graphics Forum* 30, 7 (2011), 1921–1930.

- [COM98] COHEN J., OLANO M., MANOCHA D.: Appearance-preserving simplification. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (1998), SIGGRAPH '98, pp. 115–122.
- [Coo84] COOK R. L.: Shade trees. *SIGGRAPH Comput. Graph.* 18, 3 (Jan. 1984), 223–231.
- [Cro84] CROW F. C.: Summed-area tables for texture mapping. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1984), SIGGRAPH '84, ACM, pp. 207–212.
- [CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: Measuring error on simplified surfaces. *Computer Graphics Forum* 17, 2 (1998), 167–174.
- [CVM*96] COHEN J., VARSHNEY A., MANOCHA D., TURK G., WEBER H., AGARWAL P., BROOKS F., WRIGHT W.: Simplification envelopes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 119–128.
- [DCSD02] DEUSSEN O., COLDITZ C., STAMMINGER M., DRETTAKIS G.: Interactive visualization of complex plant ecosystems. In *Proceedings of the Conference on Visualization '02* (Washington, DC, USA, 2002), VIS '02, IEEE Computer Society, pp. 219–226.
- [Dew08] DEWEY M. W.: *Automated Quadrilateral Coarsening by Ring Collapse*. Master's thesis, Brigham Young University, 2008.
- [dFH*11] D'EON E., FRANCOIS G., HILL M., LETTERI J., AUBRY J.: An energy-conserving hair reflectance model. *Computer Graphics Forum* 30, 4 (2011), 1181–1187.
- [DHI*13] DUPUY J., HEITZ E., IEHL J.-C., POULIN P., NEYRET F., OSTROMOUKHOV V.: Linear efficient antialiased displacement and reflectance mapping. *ACM Trans. Graph.* 32, 6 (Nov. 2013), 211:1–211:11.
- [DI11] D'EON E., IRVING G.: A quantized-diffusion model for rendering translucent materials. *ACM Trans. Graph.* 30, 4 (July 2011), 56:1–56:14.
- [DKT98] DEROSE T., KASS M., TRUONG T.: Subdivision surfaces in character animation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 85–94.
- [DLG90] DYN N., LEVINE D., GREGORY J. A.: A butterfly subdivision scheme for surface interpolation with tension control. *ACM Trans. Graph.* 9, 2 (Apr. 1990), 160–169.
- [dMH13] D'EON E., MARSCHNER S., HANIKA J.: Importance sampling for physically-based hair fiber models. In *SIGGRAPH Asia 2013 Technical Briefs* (New York, NY, USA, 2013), SA '13, ACM, pp. 25:1–25:4.
- [DS78] DOO D., SABIN M.: Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design* 10, 6 (1978), 356 – 360.
- [DSC09] DANIELS II J., SILVA C. T., COHEN E.: Localized quadrilateral coarsening. In *Proceedings of the Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2009), SGP '09, Eurographics Association, pp. 1437–1444.
- [DSSC08] DANIELS J., SILVA C. T., SHEPHERD J., COHEN E.: Quadrilateral mesh simplification. *ACM Trans. Graph.* 27, 5 (Dec. 2008), 148:1–148:9.
- [DvGNK99] DANA K. J., VAN GINNEKEN B., NAYAR S. K., KOENDERINK J. J.: Reflectance and texture of real-world surfaces. *ACM Trans. Graph.* 18, 1 (Jan. 1999), 1–34.
- [DWMG15] DONG Z., WALTER B., MARSCHNER S., GREENBERG D. P.: Predicting appearance from measured microgeometry of metal surfaces. *ACM Trans. Graph.* 35, 1 (Dec. 2015), 9:1–9:13.
- [Ede06] EDELSBRUNNER H.: *Geometry and Topology for Mesh Generation*. Cambridge University Press, New York, NY, USA, 2006.

- [EM99] ERIKSON C., MANOCHA D.: GAPS: General and automatic polygonal simplification. In *Proceedings of the 1999 Symposium on Interactive 3D Graphics* (New York, NY, USA, 1999), I3D '99, ACM, pp. 79–88.
- [FF88] FOURNIER A., FIUME E.: Constant-time filtering with space-variant kernels. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1988), SIGGRAPH '88, ACM, pp. 229–238.
- [Fou92a] FOURNIER A.: *Filtering Normal Maps and Creating Multiple Surfaces*. Tech. Rep. TR-92-41, Department of Computer Science, University of British Columbia, Vancouver, BC, Canada, 1992.
- [Fou92b] FOURNIER A.: Normal distribution functions and multiple surfaces. In *Graphics Interface '92 Workshop on Local Illumination* (Vancouver, BC, Canada, May 1992), pp. 45–52.
- [GG07] GUENNEBAUD G., GROSS M.: Algebraic point set surfaces. In *ACM SIGGRAPH 2007 Papers* (New York, NY, USA, 2007), SIGGRAPH '07, ACM.
- [GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (1997), SIGGRAPH '97, pp. 209–216.
- [GH98] GARLAND M., HECKBERT P. S.: Simplifying surfaces with color and texture using quadric error metrics. In *Proceedings of the Conference on Visualization '98* (1998), pp. 263–269.
- [GM05] GOBBETTI E., MARTON F.: Far voxels: A multiresolution framework for interactive rendering of huge complex 3d models on commodity graphics platforms. *ACM Trans. Graph.* 24, 3 (July 2005), 878–885.
- [Gro09] GROSS M.: Point based graphics: State of the art and recent advances. In *ACM SIGGRAPH 2009 Courses* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 18:1–18:68.
- [Gué99] GUÉZIEC A.: Locally toleranced surface simplification. *IEEE Transactions on Visualization and Computer Graphics* 5, 2 (Apr. 1999), 168–189.
- [GZB*13] GKIOULEKAS I., ZHAO S., BALA K., ZICKLER T., LEVIN A.: Inverse volume rendering with material dictionaries. *ACM Trans. Graph.* 32, 6 (Nov. 2013), 162:1–162:13.
- [Har96] HART J. C.: Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer* 12, 10 (Dec 1996), 527–545.
- [HCJ13] HABEL R., CHRISTENSEN P. H., JAROSZ W.: *Classical and Improved Diffusion Theory for Subsurface Scattering*. Tech. rep., Disney Research Zürich, June 2013.
- [HD15] HEITZ E., DUPUY J.: *Implementing a Simple Anisotropic Rough Diffuse Material with Stochastic Evaluation*. Tech. rep., 2015.
- [HDCD15] HEITZ E., DUPUY J., CRASSIN C., DACHSBACHER C.: The SGGX microflake distribution. *ACM Trans. Graph.* 34, 4 (July 2015), 48:1–48:11.
- [HDD*92] HOPPE H., DEROSE T., DUCHAMP T., McDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. *SIGGRAPH Comput. Graph.* 26, 2 (July 1992), 71–78.
- [HDD*93] HOPPE H., DEROSE T., DUCHAMP T., McDONALD J., STUETZLE W.: Mesh optimization. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1993), SIGGRAPH '93, ACM, pp. 19–26.
- [HDD*94] HOPPE H., DEROSE T., DUCHAMP T., HALSTEAD M., JIN H., McDONALD J., SCHWEITZER J., STUETZLE W.: Piecewise smooth surface reconstruction. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1994), SIGGRAPH '94, ACM, pp. 295–302.
- [Hec89] HECKBERT P. S.: *Fundamentals of texture mapping and image warping*. Tech. rep., 1989.
- [Hei14a] HEITZ E.: *Appearance of multi-scale surfaces*. Theses, Université de Grenoble, Sept. 2014.

- [Hei14b] HEITZ E.: Understanding the masking-shadowing function in microfacet-based brdfs. *Journal of Computer Graphics Techniques (JCGT)* 3, 2 (June 2014), 48–107.
- [Her03] HERY C.: Renderman, theory and practice. In *ACM SIGGRAPH 2003 Course Notes* (2003), SIGGRAPH '03, pp. 73–88.
- [HG41] HENYEV L. G., GREENSTEIN J. L.: Diffuse radiation in the Galaxy. *The Astrophysical Journal* 93 (Jan. 1941), 70–83.
- [HN12] HEITZ E., NEYRET F.: Representing Appearance and Pre-filtering Subpixel Data in Sparse Voxel Octrees. In *EGGH-HPG'12 - Eurographics conference on High Performance Graphics* (2012), pp. 125–134.
- [HNPN13] HEITZ E., NOWROUZEZAHRAI D., POULIN P., NEYRET F.: Filtering color mapped textures and surfaces. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2013), I3D '13, ACM, pp. 129–136.
- [HNPN14] HEITZ E., NOWROUZEZAHRAI D., POULIN P., NEYRET F.: Filtering Non-Linear Transfer Functions on Surfaces. *IEEE Transactions on Visualization and Computer Graphics* 20, 7 (July 2014), 996–1008.
- [Hop96] HOPPE H.: Progressive meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 99–108.
- [Hop97] HOPPE H.: View-dependent refinement of progressive meshes. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 189–198.
- [Hop99] HOPPE H.: New quadric metric for simplifying meshes with appearance attributes. In *Proceedings of the 10th IEEE Visualization 1999 Conference (VIS '99)* (Washington, DC, USA, 1999), VISUALIZATION '99, IEEE Computer Society, pp. –.
- [HR13] HAŠAN M., RAMAMOORTHI R.: Interactive albedo editing in path-traced volumetric materials. *ACM Transactions on Graphics* (2013).
- [HSRG07] HAN C., SUN B., RAMAMOORTHI R., GRINSPUN E.: Frequency domain normal map filtering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)* 26, 3 (2007), 28:1–28:12.
- [ICG86] IMMEL D. S., COHEN M. F., GREENBERG D. P.: A radiosity method for non-diffuse environments. *SIGGRAPH Comput. Graph.* 20, 4 (Aug. 1986), 133–142.
- [Ish78] ISHIMARU A.: *Wave propagation and scattering in random media*. Academic Press, 1978.
- [Jak10] JAKOB W.: Mitsuba renderer, 2010.
- [JAM*10a] JAKOB W., ARBREE A., MOON J. T., BALA K., MARSCHNER S.: A radiative transfer framework for rendering materials with anisotropic structure. *ACM Trans. Graph.* 29, 4 (July 2010), 53:1–53:13.
- [Jam10b] JAMRIŠKA O.: Interactive ray tracing of distance fields.
- [JC98] JENSEN H. W., CHRISTENSEN P. H.: Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 311–320.
- [JHY*14] JAKOB W., HAŠAN M., YAN L.-Q., LAWRENCE J., RAMAMOORTHI R., MARSCHNER S.: Discrete stochastic microfacet models. *ACM Trans. Graph.* 33, 4 (July 2014), 115:1–115:10.
- [JMLH01] JENSEN H. W., MARSCHNER S. R., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 511–518.

- [Kaj85] KAJIYA J. T.: Anisotropic reflection models. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1985), SIGGRAPH '85, ACM, pp. 15–21.
- [Kaj86] KAJIYA J. T.: The rendering equation. *SIGGRAPH Comput. Graph.* 20, 4 (Aug. 1986), 143–150.
- [Kan14] KANYUK P.: Level of detail in an age of gi: Rethinking crowd rendering. In *ACM SIGGRAPH 2014 Talks* (New York, NY, USA, 2014), SIGGRAPH '14, ACM, pp. 6:1–6:1.
- [KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association, pp. 61–70.
- [KFF*15] KELLER A., FASCIONE L., FAJARDO M., GEORGIEV I., CHRISTENSEN P., HANIKA J., EISENACHER C., NICHOLS G.: The path tracing revolution in the movie industry. In *ACM SIGGRAPH 2015 Courses* (New York, NY, USA, 2015), SIGGRAPH '15, ACM, pp. 24:1–24:7.
- [KHPL16] KAPLANYAN A. S., HILL S., PATNEY A., LEFOHN A.: Filtering distributions of normals for shading antialiasing. In *Proceedings of High Performance Graphics* (Aire-la-Ville, Switzerland, Switzerland, 2016), HPG '16, Eurographics Association, pp. 151–162.
- [Kin97] KINNEY P.: Cleanup: Improving quadrilateral finite element meshes.
- [KK89] KAJIYA J. T., KAY T. L.: Rendering fur with three dimensional textures. *SIGGRAPH Comput. Graph.* 23, 3 (July 1989), 271–280.
- [KKF*02] KLEIN J., KROKOWSKI J., FISCHER M., WAND M., WANKA R., MEYER AUF DER HEIDE F.: The randomized sample tree: A data structure for interactive walkthroughs in externally stored virtual environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2002), VRST '02, ACM, pp. 137–146.
- [KLS96] KLEIN R., LIEBICH G., STRASSER W.: Mesh reduction with error control. In *Proceedings of the 7th Conference on Visualization '96* (Los Alamitos, CA, USA, 1996), VIS '96, IEEE Computer Society Press, pp. 311–318.
- [Kob96] KOBBELT L.: Interpolatory subdivision on open quadrilateral nets with arbitrary topology. *Computer Graphics Forum* 15, 3 (1996), 409–420.
- [KSZ*15] KHUNGURN P., SCHROEDER D., ZHAO S., BALA K., MARSCHNER S.: Matching real fabrics with micro-appearance models. *ACM Trans. Graph.* 35, 1 (Dec. 2015), 1:1–1:26.
- [KTO11] KONTKANEN J., TABELLION E., OVERBECK R. S.: Coherent out-of-core point-based global illumination. In *Proceedings of the Twenty-second Eurographics Conference on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2011), EGSR '11, Eurographics Association, pp. 1353–1360.
- [KVH84] KAJIYA J. T., VON HERZEN B. P.: Ray tracing volume densities. *SIGGRAPH Comput. Graph.* 18, 3 (Jan. 1984), 165–174.
- [LBBS08] LACEWELL D., BURLEY B., BOULOS S., SHIRLEY P.: Raytracing prefiltered occlusion for aggregate geometry. In *IEEE Symposium on Interactive Raytracing 2008* (2008).
- [LBG16] LAMBERT T., BÉNARD P., GUENNEBAUD G.: Multi-Resolution Meshes for Feature-Aware Hardware Tessellation. *Computer Graphics Forum* (2016).
- [LE97] LUEBKE D., ERIKSON C.: View-dependent simplification of arbitrary polygonal environments. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 199–208.
- [LK10a] LAINE S., KARRAS T.: Efficient sparse voxel octrees. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2010), I3D '10, ACM, pp. 55–63.

- [LK10b] LAINE S., KARRAS T.: *Efficient Sparse Voxel Octrees – Analysis, Extensions, and Implementation*. NVIDIA Technical Report NVR-2010-001, NVIDIA Corporation, Feb. 2010.
- [LKR*96] LINDSTROM P., KOLLER D., RIBARSKY W., HODGES L. F., FAUST N., TURNER G. A.: Real-time, continuous level of detail rendering of height fields. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 109–118.
- [LN17] LOUBET G., NEYRET F.: Hybrid mesh-volume LoDs for all-scale pre-filtering of complex 3D assets. *Comput. Graph. Forum* 36, 2 (May 2017), 431–442.
- [LN18] LOUBET G., NEYRET F.: A new microflake model with microscopic self-shadowing for accurate volume downsampling. *Computer Graphics Forum* 37, 2 (2018), 1–11.
- [Loo87] LOOP C.: *Smooth subdivision surfaces based on triangles*. Master's thesis, University of Utah, Department of Mathematics, 1987.
- [LPD14] LIKTOR G., PAN M., DACHSBACHER C.: Fractional Reyes-Style Adaptive Tessellation for Continuous Level of Detail. *Computer Graphics Forum* 33, 7 (2014), 191–198.
- [LS08] LOOP C., SCHAEFER S.: Approximating catmull-clark subdivision surfaces with bicubic patches. *ACM Trans. Graph.* 27, 1 (Mar. 2008), 8:1–8:11.
- [LSNCn09] LOOP C., SCHAEFER S., NI T., CASTAÑO I.: Approximating subdivision surfaces with gregory patches for hardware tessellation. In *ACM SIGGRAPH Asia 2009 Papers* (New York, NY, USA, 2009), SIGGRAPH Asia '09, ACM, pp. 151:1–151:9.
- [LT98] LINDSTROM P., TURK G.: Fast and memory efficient polygonal simplification. In *Proceedings of the Conference on Visualization '98* (Los Alamitos, CA, USA, 1998), VIS '98, IEEE Computer Society Press, pp. 279–286.
- [LT00] LINDSTROM P., TURK G.: Image-driven simplification. *ACM Trans. Graph.* 19, 3 (July 2000), 204–241.
- [LW93] LAFORTUNE E. P., WILLEMS Y. D.: Bi-directional path tracing. In *PROCEEDINGS OF THIRD INTERNATIONAL CONFERENCE ON COMPUTATIONAL GRAPHICS AND VISUALIZATION TECHNIQUES (COMPUGRAPHICS '93)* (1993), pp. 145–153.
- [LWC*02] LUEBKE D., WATSON B., COHEN J. D., REDDY M., VARSHNEY A.: *Level of Detail for 3D Graphics*. Elsevier Science Inc., New York, NY, USA, 2002.
- [Mar12] MARSCHNER S.: Volumetric path tracing, March 2012.
- [MGH*05] MONAGAN M. B., GEDDES K. O., HEAL K. M., LABAHN G., VORKOETTER S. M., MCCARRON J., DEMARCO P.: *Maple 10 Programming Guide*. Maplesoft, Waterloo ON, Canada, 2005.
- [MJC*03] MARSCHNER S. R., JENSEN H. W., CAMMARANO M., WORLEY S., HANRAHAN P.: Light scattering from human hair fibers. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 780–791.
- [MKMW97] MAX N., KEATING B., MOBLEY C., WU E.-H.: *Plane-Parallel Radiance Transport for Global Illumination in Vegetation*. Springer Vienna, Vienna, 1997, pp. 239–250.
- [MM06] MOON J. T., MARSCHNER S. R.: Simulating multiple scattering in hair using a photon mapping approach. *ACM Trans. Graph.* 25, 3 (July 2006), 1067–1074.
- [MN88] MITCHELL D. P., NETRAVALI A. N.: Reconstruction filters in computer-graphics. *SIGGRAPH Comput. Graph.* 22, 4 (June 1988), 221–228.
- [Mor01] MORETON H.: Watertight tessellation using forward differencing. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware* (New York, NY, USA, 2001), HWWS '01, ACM, pp. 25–32.

- [MPFJ99] MCCORMACK J., PERRY R., FARKAS K. I., JOUPPI N. P.: Feline: Fast elliptical lines for anisotropic texture mapping. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1999), SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., pp. 243–250.
- [MPG*16] MÜLLER T., PAPAS M., GROSS M., JAROSZ W., NOVÁK J.: Efficient rendering of heterogeneous polydisperse granular media. *ACM Trans. Graph.* (2016), 168:1–168:14.
- [MPH*15] MENG J., PAPAS M., HABEL R., DACHSBACHER C., MARSCHNER S., GROSS M., JAROSZ W.: Multi-scale modeling and rendering of granular materials. *ACM Trans. Graph.* 34, 4 (July 2015), 49:1–49:13.
- [MWM07] MOON J. T., WALTER B., MARSCHNER S. R.: Rendering discrete random media using precomputed scattering solutions. EGSR'07, pp. 231–242.
- [MWM08] MOON J. T., WALTER B., MARSCHNER S.: Efficient multiple scattering in hair using spherical harmonics. *ACM Trans. Graph.* 27, 3 (Aug. 2008), 31:1–31:7.
- [Ney95] NEYRET F.: A General and Multiscale Model for Volumetric Textures. In *Graphics Interface* (Toronto, Canada, 1995), pp. 83–91.
- [Ney98] NEYRET F.: Modeling, animating, and rendering complex scenes using volumetric textures. *IEEE Transactions on Visualization and Computer Graphics* 4, 1 (Jan. 1998), 55–70.
- [NKF*16] NIESSNER M., KEINERT B., FISHER M., STAMMINGER M., LOOP C., SCHÄFER H.: Real-time rendering techniques with hardware tessellation. *Comput. Graph. Forum* 35, 1 (Feb. 2016), 113–137.
- [NL13] NIESSNER M., LOOP C.: Analytic displacement mapping using hardware tessellation. *ACM Trans. Graph.* 32, 3 (July 2013), 26:1–26:9.
- [NLMD12] NIESSNER M., LOOP C., MEYER M., DEROSE T.: Feature-adaptive gpu rendering of catmull-clark subdivision surfaces. *ACM Trans. Graph.* 31, 1 (Feb. 2012), 6:1–6:11.
- [OB10] OLANO M., BAKER D.: Lean mapping. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2010), I3D '10, ACM, pp. 181–188.
- [ON97] OLANO M., NORTH M.: *Normal distribution mapping*. Tech. rep., 07 1997.
- [Ope]
- [OVBP11] OVREIU E., VALETTE S., BUZULOIU V., PROST R.: Mesh simplification using an accurate measured quadratic error. In *Signals, Circuits and Systems (ISSCS)* (2011).
- [Pan14] PANTELEEV A.: Practical real-time voxel-based global illumination for current GPUs. Presented at SIGGRAPH '14 - Best of GTC talks, 2014.
- [PES15] PRUS M., EISENACHER C., STAMMINGER M.: Level-of-detail for production-scale path tracing. In *Vision, Modeling & Visualization* (2015), Bommes D., Ritschel T., Schultz T., (Eds.), The Eurographics Association.
- [PH08] P. H. C.: *Point-Based Approximate Color Bleeding*. Tech. rep., 01 2008.
- [PMA14] PALMER S., MAURER E., ADAMS M.: Using sparse voxel octrees in a level-of-detail pipeline for rio 2. In *ACM SIGGRAPH 2014 Talks* (New York, NY, USA, 2014), SIGGRAPH '14, ACM, pp. 70:1–70:1.
- [PR97] PETERS J., REIF U.: The simplest subdivision scheme for smoothing polyhedra. *ACM Trans. Graph.* 16, 4 (Oct. 1997), 420–431.
- [PZvBG00] PFISTER H., ZWICKER M., VAN BAAR J., GROSS M.: Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (2000), SIGGRAPH '00, pp. 335–342.

- [RB85] REEVES W. T., BLAU R.: Approximate and probabilistic algorithms for shading and rendering structured particle systems. *SIGGRAPH Comput. Graph.* 19, 3 (July 1985), 313–322.
- [RB93] ROSSIGNAC J., BORREL P.: Multi-resolution 3D approximations for rendering complex scenes. In *Modeling in Computer Graphics* (Berlin, Heidelberg, 1993), Falcidieno B., Kunii T. L., (Eds.), Springer Berlin Heidelberg, pp. 455–465.
- [Ros97] ROSSIGNAC J.: Geometric simplification and compression. In *Multiresolution Surface Modeling, Course Notes #25* (1997), SIGGRAPH'97.
- [SBS08] STATEN M. L., BENZLEY S., SCOTT M.: A methodology for quadrilateral finite element mesh coarsening. *Engineering with Computers* 24, 3 (Sept. 2008), 241–251.
- [Sch97] SCHILLING A.: Towards real-time photorealistic rendering: Challenges and solutions. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware* (New York, NY, USA, 1997), HWWS '97, ACM, pp. 7–15.
- [SFS05] SCHEIDEGGER C. E., FLEISHMAN S., SILVA C. T.: Triangulating point set surfaces with bounded error. In *Proceedings of the Third Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2005), SGP '05, Eurographics Association.
- [SKS96] SCHILLING A., KNITTEL G., STRASSER W.: Texram: A smart memory for texturing. *IEEE Comput. Graph. Appl.* 16, 3 (May 1996), 32–41.
- [SKZ11] SCHRÖDER K., KLEIN R., ZINKE A.: A volumetric approach to predictive rendering of fabrics. EGSR '11, pp. 1277–1286.
- [Smi67] SMITH B.: Geometrical shadowing of a random rough surface. 668 – 671.
- [SPM*13] SCHÄFER H., PRUS M., MEYER Q., SUESSMUTH J., STAMMINGER M.: Multiresolution attributes for hardware tessellated objects. *IEEE Transactions on Visualization and Computer Graphics* 19, 9 (Sept 2013), 1488–1498.
- [SRK*15] SCHÄFER H., RAAB J., KEINERT B., MEYER M., STAMMINGER M., NIESSNER M.: Dynamic feature-adaptive subdivision. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2015), i3D '15, ACM, pp. 31–38.
- [SS95] SCHAUFLEER G., STÜRZLINGER W.: Generating multiple levels of detail from polygonal geometry models. In *Virtual Environments '95* (Vienna, 1995), Göbel M., (Ed.), Springer Vienna, pp. 33–41.
- [Sta98] STAM J.: Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 395–404.
- [SZL92] SCHROEDER W. J., ZARGE J. A., LORENSEN W. E.: Decimation of triangle meshes. *SIGGRAPH Comput. Graph.* 26, 2 (July 1992), 65–70.
- [TB94] TAYLOR D. C., BARRETT W. A.: An algorithm for continuous resolution polygonalizations of a discrete surface. In *Proceedings of Graphics Interface '94* (Toronto, Ontario, Canada, 1994), GI '94, Canadian Human-Computer Communications Society, pp. 33–42.
- [tBPHJ02] T. BREMER P., PORUMBESCU S. D., HAMANN B., JOY K. I.: Automatic semi-regular mesh construction from adaptive distance fields. In *In Proc. Curves and Surfaces 02* (2002).
- [TLQ*05] TAN P., LIN S., QUAN L., GUO B., SHUM H.-Y.: Multiresolution Reflectance Filtering. In *Eurographics Symposium on Rendering (2005)* (2005), Bala K., Dutre P., (Eds.), The Eurographics Association.
- [TLQ*08] TAN P., LIN S., QUAN L., GUO B., SHUM H.: Filtering and rendering of resolution-dependent reflectance models. *IEEE Transactions on Visualization and Computer Graphics* 14, 2 (Mar. 2008), 412–425.
- [Tok05] TOKSVIG M.: Mipmapping normal maps. 65–71.

- [TPC*10] TARINI M., PIETRONI N., CIGNONI P., PANOZZO D., PUPPO E.: Practical quad mesh simplification. *Computer Graphics Forum* 29, 2 (2010), 407–418.
- [TS06] TSAI Y.-T., SHIH Z.-C.: All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. In *Proceedings of SIGGRAPH'06* (2006), pp. 967–976.
- [Tur92] TURK G.: Re-tiling polygonal surfaces. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1992), SIGGRAPH '92, ACM, pp. 55–64.
- [Vea98] VEACH E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford, CA, USA, 1998. AAI9837162.
- [Wei17] WEISSTEIN E. W.: Trigonometric power formulas., 2017.
- [Wil83] WILLIAMS L.: Pyramidal parametrics. *SIGGRAPH Comput. Graph.* 17, 3 (July 1983), 1–11.
- [WK93] WANG S. W., KAUFMAN A. E.: Volume sampled voxelization of geometric primitives. In *Proceedings of the 4th Conference on Visualization '93* (Washington, DC, USA, 1993), VIS '93, IEEE Computer Society, pp. 78–84.
- [WLC*03] WILLIAMS N., LUEBKE D., COHEN J. D., KELLEY M., SCHUBERT B.: Perceptually guided simplification of lit, textured meshes. In *Proceedings of the 2003 Symposium on Interactive 3D Graphics* (New York, NY, USA, 2003), I3D '03, ACM, pp. 113–121.
- [WMB15] WANG B., MENG X., BOUBEKEUR T.: Wavelet point-based global illumination. In *Proceedings of the 26th Eurographics Symposium on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2015), EGSR '15, Eurographics Association, pp. 143–153.
- [WMLT07] WALTER B., MARSCHNER S. R., LI H., TORRANCE K. E.: Microfacet models for refraction through rough surfaces. In *Proceedings of EGSR'07* (2007), pp. 195–206.
- [WPW89] WYMAN D. R., PATTERSON M. S., WILSON B. C.: Similarity relations for the interaction parameters in radiation transport. *Appl. Opt.* 28, 24 (Dec 1989), 5243–5249.
- [WS05] WALD I., SEIDEL H. P.: Interactive ray tracing of point-based models. In *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005*. (June 2005), pp. 9–16.
- [WVH17] WRENNINGE M., VILLEMIN R., HERY C.: *Path Traced Subsurface Scattering using Anisotropic Phase Functions and Non-Exponential Free Flights*. Tech. rep., Pixar Technical Memo #17-07, 2017.
- [XESV97] XIA J. C., EL-SANA J., VARSHNEY A.: Adaptive real-time level-of-detail-based rendering for polygonal models. *IEEE Transactions on Visualization and Computer Graphics* 3, 2 (Apr. 1997), 171–183.
- [XSD*13] XU K., SUN W.-L., DONG Z., ZHAO D.-Y., WU R.-D., HU S.-M.: Anisotropic spherical gaussians. *ACM Trans. Graph.* 32, 6 (Nov. 2013), 209:1–209:11.
- [XWZB17] XU C., WANG R., ZHAO S., BAO H.: Real-time linear brdf mip-mapping. *Comput. Graph. Forum* 36, 4 (July 2017), 27–34.
- [YHJ*14] YAN L.-Q., HAŠAN M., JAKOB W., LAWRENCE J., MARSCHNER S., RAMAMOORTHY R.: Rendering glints on high-resolution normal-mapped specular surfaces. *ACM Trans. Graph.* 33, 4 (July 2014), 116:1–116:9.
- [YHMR16] YAN L.-Q., HAŠAN M., MARSCHNER S., RAMAMOORTHY R.: Position-normal distributions for efficient rendering of specular microstructure. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2016)* 35, 4 (2016).
- [YJR17] YAN L.-Q., JENSEN H. W., RAMAMOORTHY R.: An efficient and practical near and far field fur reflectance model. *ACM Trans. Graph.* 36, 4 (July 2017), 67:1–67:13.

- [YLM06] YOON S.-E., LAUTERBACH C., MANOCHA D.: R-lods: Fast lod-based ray tracing of massive models. *Vis. Comput.* 22, 9 (Sept. 2006), 772–784.
- [ZHRB13] ZHAO S., HAŠAN M., RAMAMOORTHY R., BALA K.: Modular flux transfer: Efficient rendering of high-resolution volumes with repeated structures. *ACM Trans. Graph.* 32, 4 (July 2013), 131:1–131:12.
- [ZJMB11] ZHAO S., JAKOB W., MARSCHNER S., BALA K.: Building volumetric appearance models of fabric using micro ct imaging. In *Proceedings of SIGGRAPH '11* (2011), pp. 44:1–44:10.
- [ZRB14] ZHAO S., RAMAMOORTHY R., BALA K.: High-order similarity relations in radiative transfer. *ACM Trans. Graph.* 33, 4 (July 2014), 104:1–104:12.
- [ZSS97] ZORIN D., SCHRÖDER P., SWELDENS W.: Interactive multiresolution mesh editing. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 259–268.
- [ZSW04] ZINKE A., SOBOTKA G., WEBER A.: Photo-realistic rendering of blond hair. In *in Vision, Modeling, and Visualization (VMV) 2004* (2004), IOS Press, pp. 191–198.
- [ZW06] ZINKE A., WEBER A.: Global illumination for fiber based geometries.
- [ZW07] ZINKE A., WEBER A.: Light scattering from filaments. *IEEE Transactions on Visualization and Computer Graphics* 13, 2 (Mar. 2007), 342–356.
- [ZWDR16] ZHAO S., WU L., DURAND F., RAMAMOORTHY R.: Downsampling scattering parameters for rendering anisotropic media. *ACM Trans. Graph.* 35, 6 (2016).

Abstract

We address the problem of rendering extremely complex virtual scenes with large amounts of detail. We focus on high-end off-line rendering algorithms based on path tracing that are intensively used in the special effects and 3D animation industries. The large amounts of detail required for creating believable virtual worlds raise important efficiency problems that paralyze production rendering pipelines and greatly complicate the work of 3D artists. We introduce new algorithms for prefiltering complex 3D assets while preserving their appearance at all scales, in order to reduce loading times, ray intersection costs, shading costs and Monte Carlo noise, without lowering the quality of rendered frames. Our main contributions are a new hybrid LOD approach that combines the benefits of meshes and volumetric representations for prefiltering complex 3D assets, as well as a new approach for prefiltering high-resolution heterogeneous participating media.

L'objectif de cette thèse est le rendu de scènes virtuelles extrêmement détaillées. Nous nous intéressons plus particulièrement aux algorithmes de rendu de haute qualité reposant sur du path tracing qui sont très largement utilisés dans l'industrie des effets spéciaux et pour calculer le rendu de films d'animations. Les grandes quantités de détails nécessaires à la modélisation de mondes virtuels crédibles soulèvent de sérieux problèmes d'efficacité du rendu qui paralysent les studios et compliquent grandement le travail des artistes. Nous introduisons de nouveaux algorithmes pour préfiltrer les objets 3D complexes à des échelles arbitraires, afin de réduire les temps de chargement, les coûts d'intersection des rayons lumineux, le calcul des matériaux et la quantité de bruit dans les images, le tout sans porter atteinte à la qualité du rendu. Nos contributions principales sont une nouvelle approche hybride de niveaux de détail qui combine les avantages des maillages et des représentations volumiques pour le préfiltrage des objets complexes à des échelles arbitraires, ainsi qu'une nouvelle approche de préfiltrage pour le cas des volumes hétérogènes de haute résolution.