

**THÈSE DE DOCTORAT DE L'UNIVERSITE DE
MARNE-LA-VALLÉE**
Spécialité
INFORMATIQUE

Présentée par : **M. THIERRY LEFEBVRE**
pour obtenir le grade de Docteur de L'Université de Marne-la-Vallée

Sujet de la thèse :

**Simulation d'éclairage réaliste de matériaux
hétérogènes en temps réel**

Soutenue le 22 Octobre 2007 devant la Commission d'examen composée de :

Jean-Pierre Jessel	rapporiteur
Bernard Péroche	rapporiteur
Andras Kemeny	co-directeur de thèse
Sylvain Michelin	co-directeur de thèse
Didier Arquès	examinateur
Laurent Lucas	examinateur

Résumé

Un simulateur de conduite est un environnement multi-sensoriel visant à reproduire avec la meilleure exactitude possible l'ensemble des stimuli présents dans la conduite réelle. Non seulement utiles pour la formation à la conduite, ils sont de véritables outils de recherche et de conception pour l'industrie automobile. Ils répondent à des besoins nombreux et variés tels que les études en ergonomie (interfaces homme-machine, visibilité du conducteur), la mise au point de systèmes d'aide à la conduite, la sûreté de fonctionnement, la mise au point de la dynamique des véhicules, ou encore de l'étude du comportement humain dans des situations accidentogènes. Pour toutes ces études, la reconstitution la plus réaliste possible des éléments visuels de l'environnement (la route, le paysage, le climat, les éclairages) est capitale.

A l'heure actuelle, les simulateurs de conduite développés par le centre technique de simulation (CTS) de Renault offrent quatre conditions climatiques de conduite : la conduite de jour par beau temps, la conduite de jour avec brouillard (sans simulation de l'éclairage du véhicule), la conduite de nuit avec simulation de l'éclairage du véhicule et la conduite de nuit par temps de brouillard.

Les travaux qui sont présentés dans ce document ont pour objectif d'améliorer le réalisme visuel pour la simulation de conduite de jour et de nuit. En particulier, nous nous focalisons sur la représentation de matériaux hétérogènes (asphalt, végétation, etc.), dont les propriétés de réflexion complexes ont un impact très important sur le réalisme de l'environnement virtuel simulé.

Pour cela, nos travaux se sont concentrés sur l'étude des BTF (Fonction de Texture Bi-directionnelle) adaptées au domaine de la simulation de conduite. Ces fonctions permettent de représenter un matériau par un ensemble de textures pour plusieurs conditions d'observation et d'éclairage. Cette méthode posant un certains nombres de contraintes (disponibilité des échantillons, quantité des données manipulées), nous avons implanté une méthode de génération de BTF synthétique ainsi qu'une technique de compression et de rendu temps réel pour la simulation d'éclairage réaliste de matériaux hétérogènes en temps réel.

Enfin, nous verrons que l'amélioration du réalisme visuel par cette méthode, qui fournit à la fois une augmentation du contraste et du nombre de détails sur les surfaces, n'est pas suffisante. Elle met en évidence les limites des systèmes de production d'images en temps réel (échantillonnage spatial et temporel) qui engendrent des artefacts visuels pénalisant ainsi le réalisme obtenu. Nous avons donc

mis en place une technique de filtrage temporel afin d'éliminer ces artefacts particulièrement présents dans le contexte de la simulation de conduite où l'observateur est principalement en mouvement.

Mots clés : simulation de conduite, simulateur, simulation d'éclairage, synthèse d'image temps réel, revêtements routiers, matériaux hétérogènes, BRDF, BTF.

Ces travaux ont été réalisés dans le cadre d'une thèse CIFRE (contrat ANRT 887/2003) entre RENAULT, Direction des Méthodes d'Ingénierie, au sein du Centre Technique de Simulation (CTS) et le Laboratoire de synthèse d'images de l'Université de Marne-la-Vallée, sous la codirection de M. Andras Kemeny (RENAULT) et de M. Sylvain Michelin (UMLV).

Abstract

A driving simulator is a multi-sensory environment aiming at reproducing stimuli present in real driving situation with the best realism. Not only useful for driver training, they are genuine tools of research and design for the car industry. They meet many needs such as studies in ergonomics (human-machine interfaces, visibility of the driver), the development of systems for driving assistance, the improvements of the dynamics of the vehicles, or studies concerning the human behavior in dangerous situations. For all these studies, the most realistic reconstitution of visual elements of the environment (road, the landscape, climate, lightings) is very important.

Today, driving simulators developed by the technical centre for simulation (CTS) of Renault offer several climatic conditions : you can drive at day time with clean weather, at day time with fog (without simulation of the lighting of the vehicle), at night time with simulation of the lighting of the vehicle projectors and at night time with presence of fog.

Our work aims to improve visual realism for driving simulation at day and night time. We focus on the representations of surfaces with heterogeneous materials (asphalt, vegetation, etc), having complex properties of reflectance because their look and feel have a very important impact on the realism of the simulated virtual environment.

To achieve this work, the study rely on the BTF (Bidirectional Texture Function) adapted to the field of driving simulation. These functions allow representing a material by a set of textures for several conditions of viewing and lighting. Since many constraints arise from this method (availability of the samples, size of the handled data), we set up a method for synthetic BTF generation as well as a technique of BTF compression and rendering algorithm for the realistic simulation of heterogeneous material lighting in real time.

Finally, we will show that improvements of visual realism provided by this method, which increases both contrast and number of details on surfaces, is not sufficient. It highlights the limits of images production systems (sampling rate for spatial and temporal space) which generates visual artifacts penalizing realism obtained by BTF simulation. We have developed a technique of temporal filtering in order to eliminate these disturbing artifacts in the context of driving simulation where the observer is mostly in movement.

Keywords : Driving simulation, driving simulator, lighting simulation, real-time image synthesis, road surface, heterogenous materials, weather conditions, BRDF, BTF.

This work was performed within the framework of an ANRT grant (887/2003) in Renault Technical Center for Simulation (CTS) and in the Image Synthesis Laboratory of Manr-la-Vallée University. This work was supervised by Dr. Andras Kemeny (Renault) and Dr. Sylvain Michelin (UMLV).

Remerciements

Ces travaux étant le fruit d'un travail collaboratif, je souhaite remercier toutes les personnes ayant contribuer à leurs réalisations.

A mes directeurs de thèse

Je remercie M. Andras Kemeny pour avoir initié ces travaux de recherche, sa confiance et de m'avoir intégré au sein de l'équipe de développement du Centre Technique de Simulation de Renault pour mon stage de DEA puis sous la forme d'un contrat CIFRE. Ces années ont été très enrichissantes tant sur la découverte du domaine de la simulation de conduite que sur son impact sur les enjeux industriels.

Je remercie M. Sylvain Michelin et M. Didier Arquès de m'avoir accueilli au sein du laboratoire de synthèse d'images de l'Université de Marne-la-Vallée ainsi que pour l'expertise et l'écoute dont ils ont fait preuve pendant la réalisation de ces travaux.

Aux membres du jury

Je remercie M. Bernard Péroche et M. Jean-Pierre Jessel d'avoir accepter d'être rapporteur et pour le temps qu'ils ont accordé à la relecture de mes travaux. Je remercie également M. Frédéric Mérienne et M. Laurent Lucas d'avoir examiner ces travaux en acceptant d'être membres du jury.

A mes collègues

Je remercie les chefs d'UET SDSI successifs, Alexandre Heidet, Loïc Joly et Vincent Rouelle pour leur implication dans ces travaux, et pour tous les enseignements que j'ai pu retirer en faisant partie intégrante de leur équipe.

Je remercie Pascal Lecocq pour son expertise sur la simulation d'éclairage et sa passion pour l'image de synthèse temps réel qu'il a su partager avec moi.

Je remercie les thésards et développeurs avec qui j'ai partagé ces années au CTS, Mehmet, Matthieu, Benoit, Damien, Vincent, Hakim et tous les autres pour la précieuse aide et le soutien qu'ils m'ont apporté dans la réalisation de ces travaux.

Je remercie aussi les autres membres du CTS, Gilles, Jean-charles, Christian, Serge, Philippe, Jean-christophe pour tous nos échanges et leurs qualités humaines durant ces années.

A ma famille

Je remercie de tout coeur ma femme, Anne-sophie, qui a toujours cru en moi, et qui m'a soutenu (on peut dire supporté) sans conditions chaque jour. Je remercie aussi ma fille, Hélia, d'avoir fait ses nuits rapidement.

Enfin, je remercie enfin mes parents, sans qui ces longues années d'études enrichissantes n'auraient peut être pas abouti à ces travaux de recherche.

A ma fille Hélia...

Table des matières

1	Introduction	17
1.1	Contexte	17
1.2	Objectifs de l'étude	18
1.3	Plan du document	18
I	Synthèse bibliographique	21
2	La lumière et le système visuel humain	23
2.1	Histoire de l'étude de la lumière	24
2.1.1	La théorie ondulatoire	24
2.1.2	La théorie corpusculaire	25
2.2	Radiométrie	25
2.3	Le système visuel humain	27
2.3.1	Anatomie de l'oeil	28
2.3.2	Traitement de l'information visuelle	30
2.3.3	Domaines de vision et fonction d'efficacité spectrale	31
2.4	Photométrie	32
2.4.1	Conversion des grandeurs radiométriques en grandeurs photométriques	32
2.4.2	Grandeurs photométriques	33
2.5	Colorimétrie	33
2.5.1	Définitions	33
2.5.2	Espace de couleur RGB (CIE 1931)	34
2.5.3	Espace de couleur CIE XYZ	35
2.5.4	Espace de couleur YCrCb	36
3	Les systèmes d'affichage	37
3.1	Introduction aux simulateurs de conduite	38
3.2	Les systèmes de vidéo-projection	39
3.2.1	Technologie CRT (Cathode Ray Tube)	39
3.2.2	Technologie LCD (Liquid Cristal Display)	40
3.2.3	Technologie DLP (Digital Light Processing)	41

3.3	Systèmes de visualisation	41
3.3.1	Écrans de projection	41
3.3.2	Casque de réalité virtuelle	42
3.4	Étude des systèmes d'affichage	43
3.4.1	Comparatifs des technologies de vidéo-projection	43
3.4.2	Limitations des systèmes d'affichage	46
3.5	Conclusion	49
4	La synthèse d'image	51
4.1	Modèle d'illumination local	52
4.1.1	Fonction de distribution de réflectance bidirectionnelle (BRDF)	52
4.1.2	Modèles de réflectance	54
4.2	Modèle d'illumination global	55
4.2.1	Lancer de rayon	55
4.2.2	Lancer de photon	58
4.2.3	Technique de radiosité	59
4.3	Le temps réel	61
4.3.1	Modèle de Phong	61
4.3.2	Pipeline graphique	62
4.3.3	Pipeline graphique programmable	63
4.3.4	Bump-mapping	66
4.4	Simulateur d'éclairage Renault	67
5	Les Fonctions de Textures Bidirectionnelles (BTF)	73
5.1	Modèles de réflectance	75
5.2	BTF	76
5.2.1	Définition	76
5.2.2	Acquisition de BTF	77
5.3	Méthodes de compression	79
5.3.1	Modèles analytiques	80
5.3.2	Décomposition en base linéaire par approche matricielle	83
5.4	Discussion	89
II	Rendu de matériaux hétérogènes en temps réel	91
6	Génération de BTF synthétique	93
6.1	Méthode de génération	94
6.1.1	Format de l'échantillonnage de la BTF	94
6.1.2	Synthèse d'images photoréalistes non temps réel	94
6.1.3	Données sources	94
6.1.4	Calcul des textures de la BTF	96
6.1.5	Transformation homographique	97

6.2	Résultats	99
6.3	Conclusion	99
7	Rendu temps réel de BTF pour la simulation de conduite	101
7.1	Étude préliminaire	102
7.1.1	Besoins et contraintes	102
7.1.2	Domaine de visualisation de surfaces hétérogènes	102
7.2	Rendu de surface de route par BRDF	105
7.2.1	Rendu avec Bump-mapping	105
7.3	Algorithme de rendu temps réel par BTF	106
7.3.1	Représentation des données et notations	106
7.3.2	Changement d'espace de couleur pour la compression	107
7.3.3	Pyramide Laplacienne	108
7.3.4	Compression par analyse en composantes principales	110
7.3.5	Représentation des variations globales du matériau par une fonction analytique	114
7.4	Implémentation temps réel	117
7.4.1	Paramétrisation des données pour le rendu	118
7.4.2	Algorithme de rendu	124
7.4.3	Détails d'implémentation	125
7.5	Résultats	125
7.5.1	Résultats en image	125
7.5.2	Performances	134
7.6	Validation	134
8	Visualisation de surfaces hétérogènes en mouvement	141
8.1	Artefacts visuels	142
8.1.1	Présentation du problème	142
8.1.2	Analyse du problème	143
8.2	Filtrage anti-aliasing temporel en temps réel	145
8.2.1	Algorithme de rendu	146
8.2.2	Détails de l'algorithme de rendu	146
8.3	Résultats	149
8.3.1	Résultats en image	149
8.3.2	Performances	150
8.3.3	Limitations	151
8.4	Conclusion	152
9	Discussion générale	155
9.1	Démarche et synthèse des résultats	156
9.2	Discussion et perspectives	158

A	Code GLSL du filtrage temporel	161
A.1	Code du Vertex Shader	161
A.2	Code du Fragment Shader	161
B	Optimisation non-linéaire : Méthode de Levenberg Marquardt	165
B.1	Définition	165
B.2	Code scilab	166
C	Code GLSL du rendu temps réel de BTF	169
C.1	Code du Vertex Shader	169
C.2	Code du Vertex Shader	170

Table des figures

1.1	Simulateur de conduite dynamique Ultimate	17
2.1	Chronologie des théories de la lumière	24
2.2	Spectre des ondes électromagnétiques et domaine visible	25
2.3	Angle solide	26
2.4	Les grandeurs de la radiométrie	26
2.5	Coupe latérale de l'oeil humain	28
2.6	Les photorécepteurs : cônes et bâtonnets	29
2.7	Projection des champs visuels sur les aires corticales. (Howard, 2002)	31
2.8	Efficacité lumineuse spectrale en vision scotopique et photopique	32
2.9	Fonctions colorimétriques du système RGB	34
2.10	Fonctions colorimétriques du système XYZ	35
2.11	Diagramme de chromacité CIE xy	36
3.1	Architecture logicielle de SCANeR II	39
3.2	Vidéo projecteur CRT	40
3.3	Vidéo projecteur LCD	40
3.4	Vidéo projecteur DLP	41
3.5	Systèmes de visualisation par vidéo-projection	42
3.6	Casque de réalité virtuelle à large champs de vision (SEOS) .	42
3.7	Gamut d'un écran CRT	45
3.8	Simulateur de conduite immersif	47
3.9	Simulateur de conduite dynamique CARDS	48
3.10	Conflit accomodation-convergence	48
4.1	Représentation de la BRDF	53
4.2	Principe du lancer de rayons	56
4.3	Simulation d'éclairage automobile	57
4.4	Simulation des phénomènes caustiques	58
4.5	Photon map	59
4.6	Facteur de forme	60
4.7	Rendu d'une scène avec la technique de radiosité	61
4.8	éclairage de Phong	62

4.9 Pipeline graphique OpenGL	62
4.10 Pipeline graphique OpenGL avec programmes GLSL	64
4.11 Rendu "cartoon rendering" temps réel par shader GLSL	66
4.12 Texture de perturbation de normales ("normal map")	67
4.13 Technique de perturbation de normales	67
4.14 Rendu temps réel avec la technique de bump-mapping	68
4.15 Rendu temps réel avec la technique de " <i>relief mapping</i> "	68
4.16 Simulateur d'éclairage Renault	69
4.17 Projecteur de Renault Scénic	69
4.18 Distribution d'éclairage d'un projecteur (feu de croisement) en fausses couleurs.	70
4.19 Technique de texture projetée	71
5.1 Paramètres nécessaires à la description de l'interaction lumière- matière	75
5.2 Hiérarchie des modèles de réflectance	76
5.3 Échantillons de BTF mesurée.	78
5.4 Système de mesure de BTF (Université de Bonn)	78
5.5 Méthode de compression JPEG	80
5.6 Comparaison d'une image (Lena) au format bmp (130 Ko, à gauche) et jpeg (40 Ko, à droite)	81
5.7 Principe de l'analyse en composantes principales (exemple en 2D)	85
6.1 Mesures de BRDF	95
6.2 Principe de génération de BTF synthétique	96
6.3 Transformation homographique	98
6.4 Échantillons de BTF synthétique (pavés)	100
7.1 surfaces intégrées dans un pixel	103
7.2 Domaine d'usage approprié pour une BRDF	103
7.3 Résolution au sol (distance de la route représentée pas un pixel)	104
7.4 Simulation d'éclairage de route bosselée (BRDF mesurée et <i>bump-mapping</i>)	106
7.5 Représentation schématique de l'ensemble des textures de la BTF d'origine	106
7.6 Images RVB et YCrCb décomposées	107
7.7 Représentation de la BTF dans l'espace de couleur YCrCb . .	108
7.8 Pyramide Laplacienne	109
7.9 Représentation de la BTF avec une pyramide Laplacienne . .	110
7.10 Ensemble des ABRDF d'un niveau de la pyramide Laplacienne	110
7.11 Schéma représentant les données normalisées issues de l'ana- lyse en composantes principales	111
7.12 Valeurs propres pour le canal Y (luminance)	112

7.13 Valeurs propres pour les canaux Cr et Cb (chromacité)	113
7.14 Analyse en composantes principales de la pyramide Laplacienne	114
7.15 Valeurs de $\mu_{k_{\max}}$ pour la composante de chromacité Cr	115
7.16 Données de variations globales du matériaux $\mu_{\lambda_{\max}}$	116
7.17 Modèle de Phong représentant les variations globales du ma- tériaux $\mu'_{\lambda_{\max}}$	117
7.18 Données issues de l'algorithme de compression pour le rendu .	118
7.19 Composante principale normalisée représentée dans une image 2D	119
7.20 Préservation du rapport de l'aire du sous-ensemble lors de la reparamétrisation.	119
7.21 Carte concentrique élevée.	120
7.22 Régions définissant une carte concentrique.	121
7.23 Étage d'une texture volumétrique S_i	122
7.24 Accès et interpolation de la texture volumétrique.	123
7.25 Rendu temps réel de BTF mesurée (échantillon "Impalla" me- suré par l'Université de Bonn).	126
7.26 Rendu temps réel de BTF mesurée sous plusieurs conditions (échantillon "Impalla").	127
7.27 Rendu temps réel de BTF synthétique de pavés sous plusieurs conditions d'éclairage	129
7.28 Rendu temps réel de BTF synthétique de pavés sous plusieurs conditions de vue	130
7.29 Rendu temps réel de BTF synthétique de pavés en vue rap- prochée	131
7.30 Rendu temps réel de BTF de velours (1)	132
7.31 Rendu temps réel de BTF de velours(2)	133
7.32 Comparaison entre la BTF source (colonne de gauche) et la BTF recomposée (colonne de droite) avec 8 composantes prin- cipales	135
7.33 Erreur relative RMSE pour chaque texture de la BTF avec 8 composantes principales	137
7.34 Erreur relative RMSE pour chaque texture de la BTF avec 4 composantes principales	139
7.35 Erreur relative RMSE en fonction du nombre de composantes principales utilisée pour la reconstruction de la BTF	140
8.1 Temps écoulé entre deux frames	142
8.2 Exemple de flou de déplacement (<i>motionblur</i>) dans une image capturée d'un film	144
8.3 Représentation en couleur des vecteurs vitesses	147
8.4 Optimisation des zones filtrées en fonction des vecteurs vitesses	147
8.5 Représentation du nombre d'échantillons en fonction de la vi- tesse	148

8.6	Filtrage dans les zones en bordure de l'image	149
8.7	Comparaison avec et sans filtrage	150
8.8	Comparaison avec et sans filtrage	151
8.9	Filtrage de texture de route très détaillée	151
8.10	Filtrage complet de la texture de route	153
9.1	Comparatif de l'évolution de la puissance CPU / GPU (<i>source : Nvidia</i>)	156

Chapitre 1

Introduction

1.1 Contexte

Depuis plusieurs années, le Centre Technique de Simulation (CTS) de Renault développe des simulateurs de conduite d'étude et de recherche (figure 1.1). Non seulement utiles pour la formation à la conduite, ils sont de véritables outils de recherche et de conception pour l'industrie automobile. Ils répondent à des besoins nombreux et variés tels que les études en ergonomie (interfaces homme-machine, visibilité du conducteur), la mise au point de systèmes d'aide à la conduite, la sûreté de fonctionnement, la mise au point de la dynamique des véhicules, l'évaluation de l'éclairage des projecteurs, ou encore de l'étude du comportement humain dans des situations accidentogènes.



FIG. 1.1 – Simulateur de conduite dynamique Ultimate

Pour toutes ces études, la reconstitution la plus réaliste possible des éléments visuels de l'environnement (la route, le paysage, le climat, les éclai-

rages) est capitale. A l'heure actuelle, les simulateurs de conduite de Renault offrent quatre conditions climatiques de conduite : la conduite de jour par beau temps, la conduite de jour avec brouillard (sans simulation de l'éclairage du véhicule), la conduite de nuit avec simulation de l'éclairage du véhicule [LP99] et la conduite de nuit par temps de brouillard [LP01].

1.2 Objectifs de l'étude

Dans un souci d'évolution et à la demande du bureau d'étude éclairage de Renault, le Centre Technique de Simulation souhaite offrir une meilleure représentativité du revêtement des surfaces composant les scènes de simulation et étendre la simulation d'éclairage actuelle aux éclairages urbains. L'objectif à terme est de disposer d'un outil permettant d'étudier des systèmes d'éclairage automobiles avancés dans différentes situations : chaussée sèche ou humide, zone urbaine, stratégies d'éclairage intelligent, etc.

Le but de cette thèse, effectuée en collaboration avec l'équipe de synthèse d'images de l'Université de Marne-la-Vallée, est d'étudier et de proposer des techniques en synthèse d'images permettant de simuler de manière réaliste et en temps réel ces phénomènes.

Cette étude porte en particulier sur la modélisation des surfaces (granulométrie, texture, réflectance, etc.) composant les scènes routières (asphalt, marquage au sol, végétation, etc.) par des algorithmes de rendu permettant d'aboutir à un résultat temps réel tout en garantissant la cohérence des résultats obtenus (validation).

1.3 Plan du document

Dans la première partie, nous réalisons une étude bibliographique fournit-
nant les bases de connaissance dans les domaines utiles à la compréhension
et à la réalisation de nos travaux.

Dans le premier chapitre, nous commençons par rappeler les études menées sur l'étude physique de la lumière et le système visuel humain.

Dans le deuxième chapitre, nous présentons les systèmes d'affichages utilisés dans le contexte de la simulation de conduite et insistons sur leurs caractéristiques.

Nous présentons dans le troisième chapitre le monde de la synthèse d'images (algorithmique, techniques temps réel, ...) et présentons comment elle peut être utile aux outils de conception que sont les simulateurs de conduite. Dans le quatrième chapitre, un état de l'art sur les méthodes de simulation d'éclairage de matériaux hétérogènes en temps réel, telles que les BRDF (Fonction de Distribution de Réflectance Bi-directionnelle) et les

BTF (Fonction de Texture Bi-directionnelle) sera réalisé.

Dans la deuxième partie, nous présentons nos travaux sur le réalisme visuel des images générées pour la simulation de conduite basé sur l'utilisation de BTF.

Nous présentons dans le cinquième chapitre une méthode de génération de BTF synthétique afin de posséder une base de donnée de matériaux pour tester des algorithmes de rendu temps réel par la technique de BTF que nous présentons dans le sixième chapitre. Nous verrons qu'il est nécessaire de tenir compte des caractéristiques des systèmes d'affichage afin de proposer un rendu d'image optimisé pour la simulation de conduite (septième chapitre) où l'observateur est principalement en mouvement.

Enfin, nous réalisons dans le huitième chapitre un bilan des apports de nos travaux et discutons des perspectives envisagées.

Première partie

Synthèse bibliographique

Chapitre 2

La lumière et le système visuel humain

Contrairement à la photographie qui est une reproduction du monde réel, la synthèse d'image a pour but la création d'images numériques par ordinateur. Elle repose non seulement sur une connaissance approfondie de la nature physique de la lumière, des lois régissant ses interactions avec son environnement, mais aussi des caractéristiques du système visuel humain.

En premier lieu, nous allons retracer les démarches scientifiques de l'histoire visant à décrire la nature physique de la lumière et nous présentons les grandeurs physiques appartenant au domaine de la radiométrie.

Ensuite, nous détaillons les caractéristiques du système visuel humain et les mécanismes de la perception visuelle afin de comprendre comment la lumière est transformée en sensation lumineuse et colorée.

Enfin, nous présentons les grandeurs de la photométrie dont l'objet est de mesurer la lumière par son effet sur le récepteur visuel ainsi que le domaine de la colorimétrie constituée par l'ensemble des données et moyens de calculs servant à spécifier les couleurs et leurs relations.

2.1 Histoire de l'étude de la lumière

Depuis déjà de nombreux siècles, l'homme cherche par différentes approches, à décrire la nature physique de la lumière. De nombreux et illustres physiciens (Huygens, Newton, Maxwell, Einstein, ...) se sont penchés sur le sujet et deux théories contradictoires ont longtemps coexistées (figure 2.1) : la théorie ondulatoire de la lumière introduite par Christian Huygens [AS84] en 1678 et la théorie corpusculaire de la lumière introduite par Isaac Newton en 1704. Ce n'est que deux siècles plus tard (1923), que Louis de Broglie, un des pionniers de la mécanique quantique, a proposé la mécanique ondulatoire en conciliant la dualité onde-corpuscule par l'unification de ces deux approches.

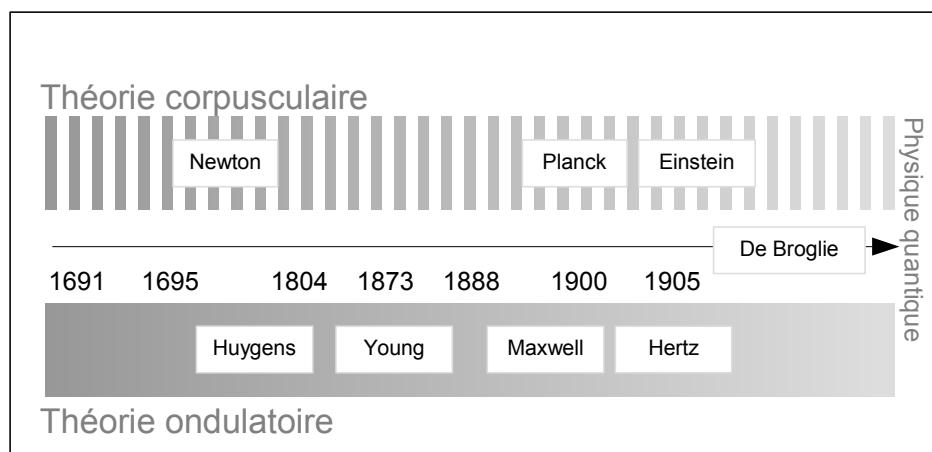


FIG. 2.1 – Chronologie des théories de la lumière

2.1.1 La théorie ondulatoire

La théorie ondulatoire de la lumière fut introduite par le Hollandais Christian Huygens (1629-1695) en 1678. Elle pris rapidement une place très importante dans la mesure où elle permettait de comprendre des phénomènes lumineux tels que la réflexion et la réfraction. Ce n'est toutefois qu'un siècle plus tard (1801) que le médecin anglais Thomas Young (1773-1829) va mettre en évidence la nature ondulatoire de la lumière en réalisant une expérience permettant d'observer des interférences. L'onde initiale, en passant par les fentes, se dédouble en deux ondes. Les franges d'interférences observées sur l'écran placé derrière, résultent de la superposition de ces ondes qui ne passent pas par le même chemin pour arriver en un point de l'écran (ondes déphasées). La variation spatiale d'intensité observée sur l'écran va dépendre du déphasage, et donc de la longueur d'onde. C'est le physicien écossais James Clerk Maxwell (1831-1879) qui précisera en 1865 que la nature de cette onde est électromagnétique. Les très connues équations de Maxwell vont permettre d'expliquer la vitesse de la lumière dans le vide (par le couplage de champs électriques et magnétiques variables) et de mettre en évidence qu'il existe

de la lumière invisible, c'est à dire que la plage des longueurs d'onde est bien plus vaste que le spectre visible (figure 2.2).

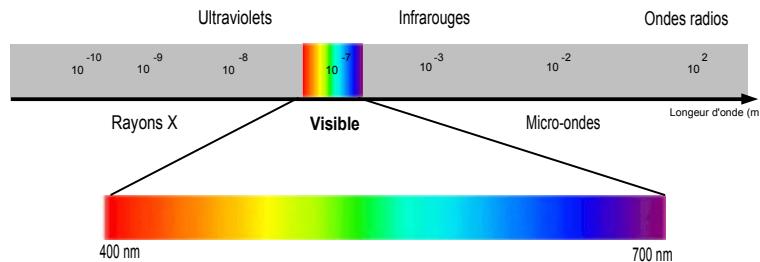


FIG. 2.2 – Spectre des ondes électromagnétiques et domaine visible

C'est vingt ans plus tard (1888) que le physicien Allemand Heinrich Hertz le démontrera de manière expérimentale en créant des ondes électromagnétiques dont la longueur d'onde était un million de fois plus grande que celle de la lumière visible, l'onde radio est née. A ce jour, on a pu observer des ondes électromagnétiques dont la longueur d'onde pouvait varier de 10^{-16}m à plusieurs milliers de kilomètres.

2.1.2 La théorie corpusculaire

En 1666, le physicien Anglais Isaac Newton (1642-1727) décompose la lumière solaire (blanche) avec un prisme et observe alors les couleurs que "contient" la lumière.

En 1900 Planck énonce la "théorie des quanta" selon laquelle, l'énergie des ondes électromagnétiques émise est absorbée de manière discontinue et indivisible.

C'est en 1905 qu' Albert Einstein, s'appuyant sur la "théorie des quanta" de Planck, mis en évidence l'existence des photons en proposant une explication de l'effet photoélectrique : lorsqu'un matériau (métal) reçoit de la lumière, il absorbe des photons possédant une énergie déterminée par la fréquence de la lumière, et dégage ainsi des électrons. Ce phénomène ne peut s'expliquer si l'on considère la lumière comme une onde, car si tel était le cas, la modification seule de l'intensité de la lumière permettrait de le provoquer.

2.2 Radiométrie

D'une manière générale, la radiométrie est la science qui consiste à mesurer les radiations électromagnétiques. Pour des rayonnements dont la propagation peut être représentée par les lois de l'optique géométrique (longueur d'onde (λ) » distance inter-atomique), les longueurs d'onde varient entre 1 nm et 10^5 nm. Les différentes grandeurs associées à la radiométrie font souvent référence à la notion d'angle solide dont voici la définition :

Angle Solide : La notion d'angle solide (figure 2.3) est semblable à la notion d'angle plan mais en trois dimensions. L'angle solide correspond au rapport entre

la surface délimitée par un ensemble de demi-droites intersectant une sphère de rayon r et le carré de ce même rayon. L'unité est le stéradian (sr).

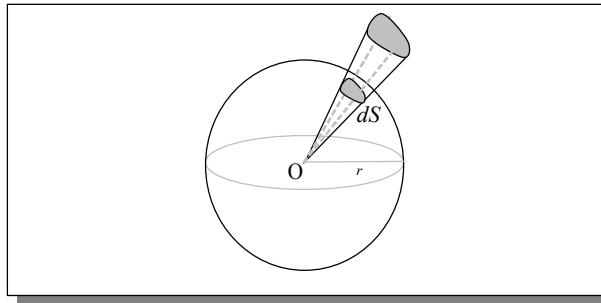


FIG. 2.3 – Angle solide

$$d\Omega = \frac{dS}{r^2}$$

Où dS est l'aire que découpe le cône sur une sphère de rayon r dont le centre (O) est au sommet du cône et r le rayon de la sphère (en mètre).

L'angle solide correspondant à tout l'espace autour d'un point vaut 4π Stéradians.

Énergie rayonnante : Elle correspond à l'énergie transportée (exprimée en Joule) par un faisceau de radiations électromagnétiques.

Flux : Le flux est la valeur instantanée d'un débit de rayonnement. Il s'exprime en lumen (lm) si cette grandeur intègre la sensibilité spectrale de l'oeil. Autrement on parle de puissance exprimée en Watt (W).

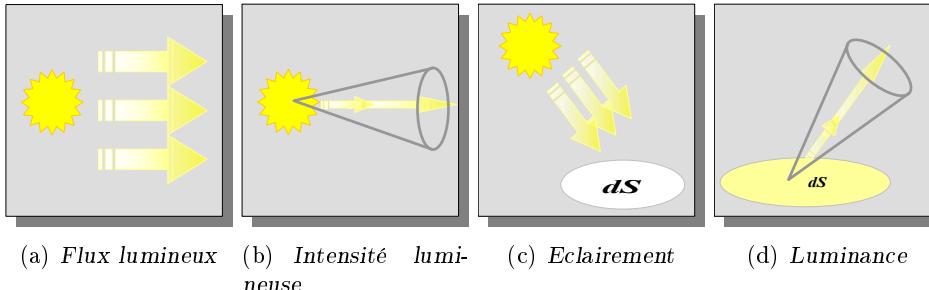


FIG. 2.4 – Les grandeurs de la radiométrie

Un flux de rayonnement de 1 W fourni, au maximum de la sensibilité spectrale de l'oeil (diurne, 555 nm), un flux lumineux de 683 lm.

Intensité lumineuse : L'intensité lumineuse, dont l'unité est la candela (cd), indique le flux lumineux émis par unité d'angle solide autour d'une direction donnée ω .

$$I = \frac{d\Phi}{d\Omega} \text{ exprimé en } W.sr^{-1}$$

Où $d\Phi$ est la puissance de la source lumineuse (W) et $d\Omega$ est l'angle solide pour la direction ω (sr).

Éclairement : L'éclairement s'exprime en Lux (lx), et correspond au flux reçu par une unité de surface.

$$E = \frac{d\Phi}{dS} \text{ exprimé en } W.m^{-2}$$

Où $d\Phi$ correspond à la puissance de la source lumineuse (W) et dS à l'unité de surface (m^2).

Emittance : L'émittance (appelée aussi radiance), s'exprime dans la même unité que l'éclairement (lux ou $W.m^{-2}$) mais correspond au flux émis par une unité de surface.

Luminance : La luminance caractérise le rayonnement d'une surface dans une direction donnée. Elle correspond au flux émis par angle solide et par surface depuis une direction d'émission.

$$L = \frac{d^2\Phi}{dS.d\Omega.\cos\theta} \text{ exprimé en } W.m^{-2}.sr^{-1}$$

avec :

- $d^2\Phi$: flux (W).
- $d\Omega$: angle solide (sr).
- dS : unité de surface (m^2).
- θ : angle entre la normale à la surface et la direction de diffusion.

Les grandeurs radiométriques présentées, pondérées par les fonctions d'efficacité spectrale de l'oeil, fournissent des grandeurs appartenant au domaine de la photométrie. Avant de détailler cette dernière, nous allons brièvement présenter le fonctionnement du système visuel humain.

2.3 Le système visuel humain

Le système visuel humain est un système fonctionnel qui traite les stimuli lumineux et permet la vision. Il est composé d'un système optique (composé de la cornée, du cristallin et du diaphragme irien) ainsi que d'un système nerveux (réteine, voies visuelles, cerveau). Son fonctionnement est très complexe et nécessiterait une description approfondie qui sort du cadre de notre étude, nous allons cependant présenter l'anatomie de l'oeil ainsi que les mécanismes de traitement de l'information visuelle de façon générale afin de connaître les principes de base régissant la vision humaine.

2.3.1 Anatomie de l'oeil

Les informations visuelles sont traitées dans un premier temps par l'oeil (figure 2.5). Il reçoit la lumière (ondes du visible dont la longueur d'onde varie entre 400 et 700 nm) qui traverse tout d'abord la cornée, membrane transparente extérieure de l'oeil. Elle est ensuite diaphragmée par la pupille formée par l'iris et dont la taille varie en fonction de l'accommodation. Enfin, la lumière est focalisée sur la rétine grâce au cristallin. Durant ce trajet, elle traverse différents milieux composant le globe oculaire (humour aqueuse et corps vitré).

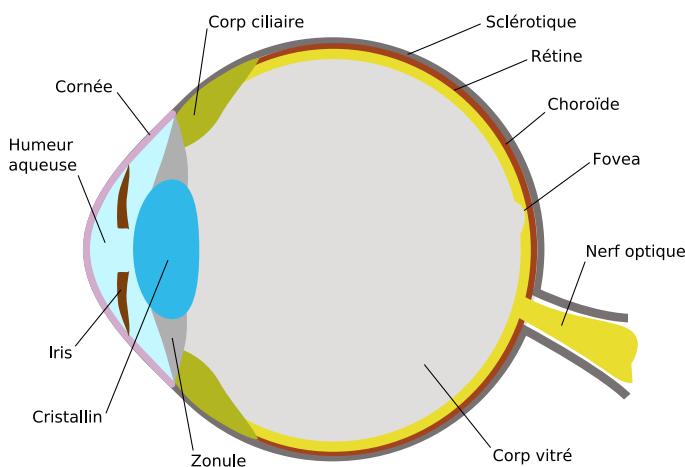


FIG. 2.5 – Coupe latérale de l'œil humain

La cornée

La cornée est une sorte de fenêtre, par laquelle la lumière provenant de l'extérieur pénètre dans notre œil. Elle joue un rôle prépondérant dans la focalisation de la lumière sur la rétine. De ce fait, elle doit toujours être parfaitement propre et transparente. La fermeture régulière des paupières (clignement) et la sécrétion lacrymale maintiennent la surface de la cornée libre de toute impureté.

Le cristallin

Le cristallin est une lentille transparente biconvexe qui constitue l'objectif de l'œil. Il est enveloppé par une capsule appelée hyaloïde sur laquelle sont fixées les fibres de la zonule de Zinn reliées aux procès ciliaires par les muscles ciliaires. Ces derniers, lorsqu'ils se contractent, étirent les fibres de la zonule de Zinn, ce qui a pour effet d'aplatir le cristallin et de modifier sa courbure. La variation de courbure du cristallin est un phénomène appelé accommodation.

La pupille

La pupille, au diamètre variable, est l'orifice central de l'iris qui contrôle le flux de lumière pénétrant dans l'oeil. L'iris comprend deux groupes de muscles : l'un composé de fibres radiales (disposées comme les rayons d'une roue) qui élargit la pupille, l'autre, comportant des fibres circulaires, qui la rétrécit. Leur action modifie le diamètre de la pupille et régule ainsi la quantité de lumière entrant dans l'oeil, tout comme le diaphragme d'un appareil photo détermine le diamètre d'ouverture de l'objectif.

La rétine

Structure lamellaire d'un quart de millimètre d'épaisseur, la rétine est constituée de trois couches de neurones. La couche la plus externe comporte des photorécepteurs contenant un pigment photosensible, qui réagit à la lumière par une modification chimique transformant l'énergie lumineuse en énergie électrique (transduction). Cette énergie est ensuite transmise au cerveau, via les cellules bipolaires puis ganglionnaires contenues dans la couche la plus interne.

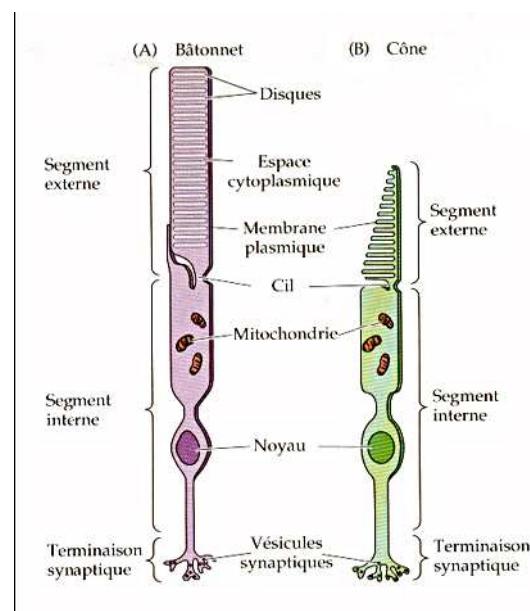


FIG. 2.6 – Les photorécepteurs : cônes et bâtonnets

Il existe deux types de photorécepteurs :

- **Les cônes :** Entre 5 et 7 millions, chacun d'eux est sensible à une longueur d'onde particulière en fonction du pigment qu'ils contiennent : le rouge (560 nm pour l'erythropsine), le vert (530 nm pour la chloropsine) ou le bleu (420 nm pour la cyanopsine). Leur densité est très élevée au centre de la rétine

(zone appelée *fovéa*) et va en diminuant vers la périphérie de la rétine. Ils sont peu sensibles à la lumière mais leur perception des détails est très grande. Chaque cône transmet son information à plusieurs fibres du nerf optique. Ils servent à la vision photopique (de jour).

- **Les bâtonnets :** Ils se situent en périphérie de la rétine et sont les plus nombreux (130 millions environ). Ils sont très sensibles à la lumière. Ils ne fournissent qu'une faible perception des détails et une très faible perception des couleurs car des dizaines de bâtonnets ne sont liés qu'à une seule fibre du nerf optique (intégration de l'information). Les bâtonnets contiennent une substance chimique, la rhodopsine, qui génère un faible courant électrique lorsqu'elle est soumise à la lumière. Ils servent à la vision scotopique (de nuit) et sont sensibles aux longueurs d'ondes proches de 500 nm.

2.3.2 Traitement de l'information visuelle

Après une succincte description de l'anatomie de l'œil décrivant comment la lumière est captée et transformée en signal électrique, nous nous intéressons maintenant au traitement de l'information visuelle.

Le nerf optique propage les informations visuelles depuis les globes oculaires jusqu'au chiasma optique (figure 2.7). Les fibres nerveuses y sont séparées avant d'être regroupées selon la partie du champ visuel à laquelle elles correspondent [SdM87] : les fibres de la partie nasale de la rétine de chaque œil traversent le chiasma optique vers l'autre hémisphère cérébral tandis que les fibres de la partie temporelle de la rétine restent dans le même hémisphère. Ce partage, appelé hémidécussation, permet de traiter dans chaque hémisphère cérébral l'intégralité des informations visuelles concernant une même moitié du champ visuel.

Une minorité des fibres nerveuses rejoint les colliculi supérieurs vers lesquels convergent des informations multisensorielles. Ils sont employés dans les tâches de contrôle de la direction de l'attention et des mouvements rapides des globes oculaires, appelés saccades. La majeure partie des axones des cellules ganglionnaires émerge du chiasma optique par les tractus optiques pour rejoindre les corps genouillés latéraux. Au sein de ces noyaux, ils forment des synapses avec des cellules relais qui, à leurs tours, propagent l'information vers le cortex visuel.

Le cortex visuel est composé de plusieurs aires (de V1 à V5), l'aire V1 étant connue sous le nom de cortex strié. L'organisation spatiale des données visuelles au sein des premières aires visuelles est dite rétinotopique, c'est-à-dire qu'elle reflète leur organisation au sein de la rétine. Du lobe occipital où se situe le cortex strié, les informations visuelles sont ensuite propagées vers d'autres régions du cerveau, en suivant principalement deux voies. Une voie dite ventrale, occupant la partie inférieure du lobe temporal, et une voie dorsale, passant par le lobe pariétal. Ces deux voies procurent des fonctions différentes : la voie ventrale est associée à l'identification des objets tandis que la voie dorsale permet leur localisation.

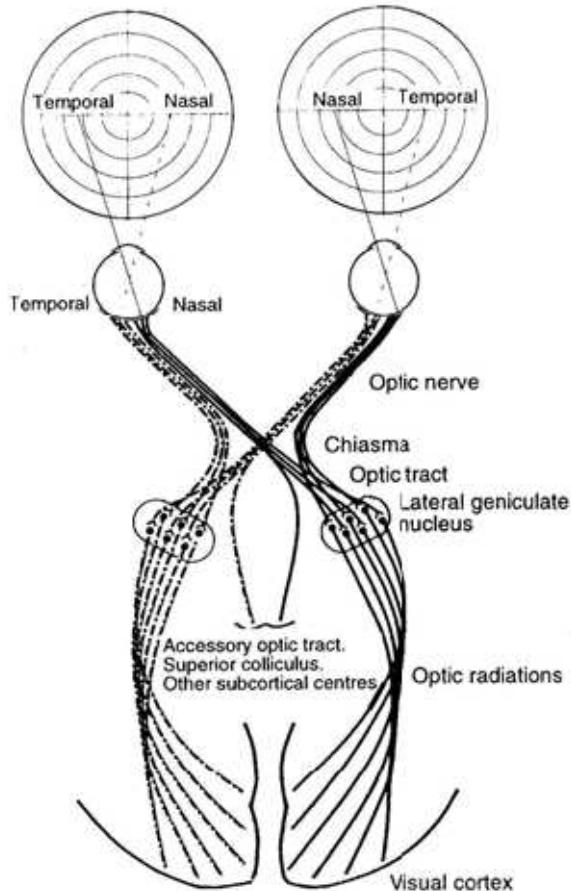


FIG. 2.7 – Projection des champs visuels sur les aires corticales. (Howard, 2002)

2.3.3 Domaines de vision et fonction d'efficacité spectrale

L'œil est sensible à une très large plage de luminance, de 10^{-6} à 10^{+8} Cd/m^2 . Cette plage se décompose en trois domaines distincts :

- Le domaine scotopique (de 10^{-6} à 10^{-3} Cd/m^2) correspond à la vision de nuit où seuls les bâtonnets sont réactifs.
- Le domaine photopique (de 10^{+1} à 10^{+8} Cd/m^2) correspond à la vision de jour où seuls les cônes sont réactifs.
- Le domaine mésopique (de 10^{-3} à 10^{+1} Cd/m^2) correspond au domaine de vision, pour lequel les bâtonnets et les cônes sont réactifs.

Pour chacun de ces domaines, la réponse de l'œil en fonction de la longueur d'onde, connue sous le nom de *fonction d'efficacité spectrale*, est différente. Elles ont

été établies expérimentalement par la CIE (Commission internationale de l'éclairage) en 1924 [dl24] pour le domaine photopique, nommée $V(\lambda)$, et en 1951 [dl51] pour le domaine scotopique, nommée $V'(\lambda)$. La fonction d'efficacité spectrale de l'oeil pour le domaine mésopique reste délicate à déterminer car les différents types de photorécepteurs réagissent simultanément. Il existe toutefois des modèles empiriques combinant les fonctions $V(\lambda)$ et $V'(\lambda)$.

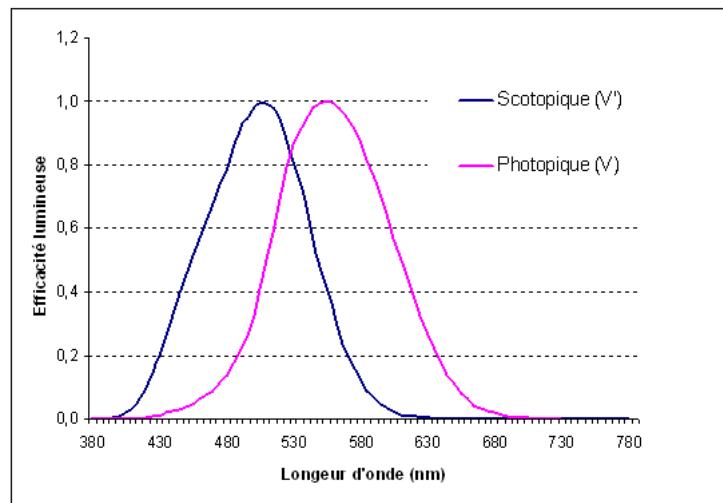


FIG. 2.8 – Efficacité lumineuse spectrale en vision scotopique (CIE 1951) et photopique (CIE 1924)

La fonction d'efficacité spectrale décrit la capacité de l'oeil à convertir l'énergie de l'onde électromagnétique en sensation lumineuse. Le bleu sera perçu moins lumineux que rouge qui lui même sera perçu moins lumineux que le vert, à même niveaux d'énergie (voir les courbes de la figure 2.8).

2.4 Photométrie

La photométrie est la science qui décrit les radiations électromagnétiques visibles par l'oeil humain. Elle consiste à traduire les grandeurs de la radiométrie en les pondérant par les fonctions d'efficacité spectrale de l'oeil $V'(\lambda)$ ou $V(\lambda)$ selon le niveau de luminance de la scène visuelle.

2.4.1 Conversion des grandeurs radiométriques en grandeurs photométriques

La luminance visuelle (celle perçue par l'oeil) se calcule, en fonction de la luminance énergétique, en intégrant le rayonnement électromagnétique, pondéré par

la fonction d'efficacité spectrale, sur tout les spectre visible. Elle s'exprime de la façon suivante :

$$L_v(\lambda) = K_m \int_{380nm}^{780nm} V(\lambda) L(\lambda) d\lambda$$

Avec :

- $L_v(\lambda)$: luminance visuelle spectrale (exprimée en cd/m^2).
- K_m : coefficient d'efficacité lumineuse (683 lm.W^{-1} pour le domaine photopique et 1700 lm.W^{-1} pour le domaine scotopique).
- $V(\lambda)$: courbe d'efficacité spectrale du domaine correspondant.
- $L(\lambda)$: luminance énergétique spectrale.

2.4.2 Grandeur photométriques

La photométrie définit trois grandeurs : le lumen, le candela et le lux. Le tableau suivant permet de visualiser les correspondances entre les grandeurs radiométriques et photométriques :

Grandeur	Unités énergétiques	Unités lumineuses	Relation
Flux (Φ)	Watt (W) $1W = 1J.s^{-1}$	Lumen (lm) $1 \text{ lm} = 1/683 \text{ W}$	
Intensité (I)	$\text{W}.sr^{-1}$	Candela (cd) $1\text{cd} = 1\text{lm}.sr^{-1}$	$I_s = \frac{d\Phi}{d\Omega}$
Éclairement (E)	W.m^{-2}	Lux (lx) $1\text{lx} = 1\text{lm.m}^{-2}$	$E = \frac{d\Phi}{dS}$
Luminance (L)	$\text{W.m}^{-2}.sr^{-1}$	Candela/ m^2 $1\text{cd.m}^{-2} = 1\text{lm.m}^{-2}.sr^{-1}$	$L = \frac{d^2\Phi}{dS.d\Omega.\cos\theta}$

TAB. 2.1 – Correspondances entre grandeurs radiométriques et photométriques

2.5 Colorimétrie

2.5.1 Définitions

La colorimétrie est la science qui consiste à spécifier les couleurs et les relations qui existent entre elles. La couleur est une modalité de la sensation visuelle issue de la variation du spectre de la lumière visible.

L'artiste peintre Albert Henry Munsell (1858-1918) a proposé une classification intuitive des couleurs selon les trois critères suivants :

- La teinte (ou tonalité) qui correspond à la distinction entre violet, bleu, vert, jaune, orange, rouge.
- La clarté (ou luminosité pour les sources lumineuses) qui situe la couleur parmi les clairs ou les foncés.

- La saturation qui exprime l'éloignement par rapport au gris de même clarté.

Cette classification a aboutie à un atlas de couleurs encore très utilisé de nos jours dans l'industrie textile.

Des expériences ont montré que les couleurs peuvent être représentées par la combinaison linéaires de trois couleurs primaires. Une couleur est dite primaire si elle ne peut être obtenue par le mélange de deux autres couleurs.

2.5.2 Espace de couleur RGB (CIE 1931)

Dans les années 1920, la CIE eu pour mission de concevoir un système colorimétrique de référence correspondant aux caractéristiques de l'œil humain. Pour cela, la CIE a du définir, à partir d'expérimentations sur un groupe de personnes représentatif, un observateur de référence colorimétrique nommé CEI 1931.

L'expérience consista à comparer des couleurs formées à partir de sources monochromatiques (de longueur d'onde unique) et des couleurs générées par le mélange de trois couleurs primaires.

Connaissant les sensibilités des trois types de cônes de la rétine, trois primaires formant un espace de couleurs (nommé CIERGB) ont été définies de la manière suivante :

- 700,0 nm pour le rouge (R)
- 546,1 nm pour le vert (G)
- 435,8 nm pour le bleu (B)

Cette expérience a permis de définir un diagramme de la sensibilité tri-chromatique représentatif des réponses de l'œil pour l'observateur de référence. Ce diagramme (figure 2.9) est constitué des courbes $\bar{r}(\lambda)$, $\bar{g}(\lambda)$ et $\bar{b}(\lambda)$ (fig. 2.9) qui permettent de convertir un spectre en une combinaison linéaire des trois composantes R, G et B.

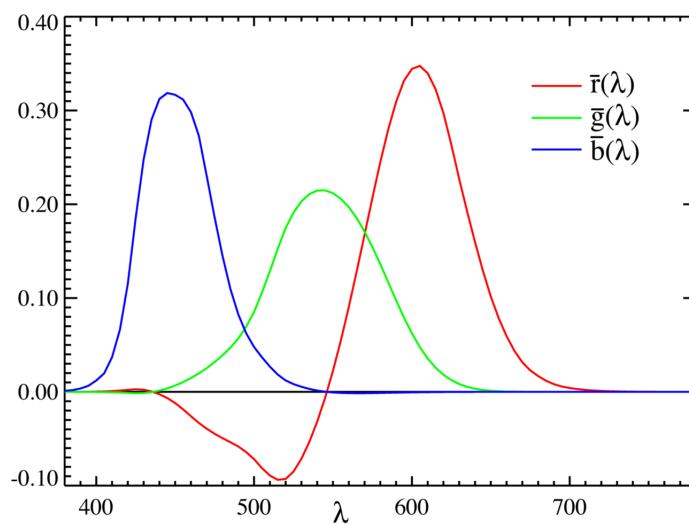


FIG. 2.9 – Fonctions colorimétriques du système RGB

Certaines valeurs de ces courbes sont négatives et sont donc en dehors de l'espace défini par les primaires, ce qui pose problème pour un modèle qui se veut universel et manipulable par les professionnels utilisant les notions de couleurs. Cette limitation a été prise en compte dans le modèle XYZ.

2.5.3 Espace de couleur CIE XYZ

Pour palier le problème du modèle CIERGB, la CIE a définit en 1931 trois nouvelles couleurs primaires XYZ (respectivement 444 nm, 526 nm et 645 nm) afin d'élargir l'espace colorimétrique pour ne manipuler que des courbes expérimentales positives $\bar{x}(\lambda)$, $\bar{y}(\lambda)$ et $\bar{z}(\lambda)$ (figure 2.10).

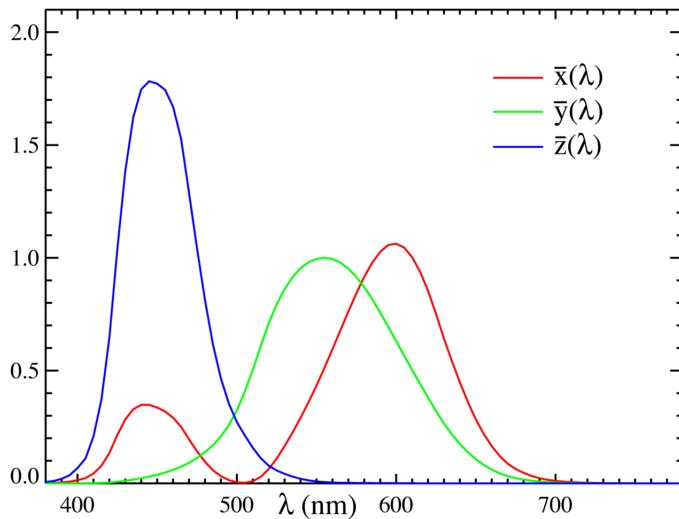


FIG. 2.10 – Fonctions colorimétriques du système XYZ

Les courbes ont été définies de telle sorte que $\bar{y}(\lambda)$ corresponde à la fonction d'efficacité spectrale de l'oeil. Les composantes primaires XYZ d'un stimulus de répartition spectrale $P(\lambda)$ se calculent de la manière suivante :

$$\begin{aligned} X &= k \int P(\lambda) \bar{x}(\lambda) d\lambda \\ Y &= k \int P(\lambda) \bar{y}(\lambda) d\lambda \\ Z &= k \int P(\lambda) \bar{z}(\lambda) d\lambda \end{aligned}$$

Un rayon issu de l'origine du repère XYZ correspond à une couleur particulière avec des niveaux de luminosité différents. Pour dissocier la teinte de la luminosité, on projette le rayon sur la plan d'équation $X + Y + Z = 1$. On définit un ensemble de teintes de même niveau de luminosité par :

$$x = \frac{X}{X + Y + Z}, y = \frac{Y}{X + Y + Z}, z = \frac{Z}{X + Y + Z}$$

avec $x + y + z = 1$.

Comme $x + y + z = 1$, il est possible de ne garder que deux composantes pour représenter la couleur (la dernière est implicite). C'est ce que fait le diagramme de chromacité CIE xy (fig. 2.11).

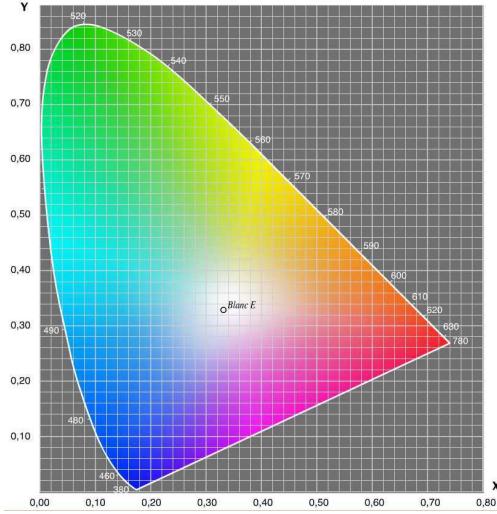


FIG. 2.11 – Diagramme de chromacité CIE xy

2.5.4 Espace de couleur YCrCb

L'espace YCrCb se calcule depuis l'espace RGB où Y représente la luminance, Cr et Cb, respectivement les composantes de chromacité rouge et bleu. La transformation s'écrit de la manière suivante :

$$Y = 0.299R + 0.587G + 0.114B$$

$$Cb = 0.5 - 0.168R - 0.331G + 0.5B$$

$$Cr = 0.5 + 0.5R - 0.419G - 0.081B$$

L'oeil étant moins sensible aux variations de chrominance qu'aux variations de luminance, ce modèle a pour principal intérêt de pouvoir représenter les composantes de chromacité Cb et Cr avec moins de précision que pour la luminance Y. L'espace YCrCb est très utilisé dans les systèmes vidéos, par exemple, le format MPEG utilise un encodage 4 :2 :2 (4 bits pour Y, 2 pour Cr et Cb).

Chapitre 3

Les systèmes d'affichage

Les informations visuelles ont un impact fondamental sur l'immersion du sujet en simulation de conduite. Le choix du matériel visant à restituer les images numériques calculées en temps réel est donc capital. Habituellement, le système visuel d'un simulateur de conduite est basé sur un système de vidéo projection, toutefois, il existe d'autres systèmes comme les casques issus de la réalité virtuelle. Après avoir présenté succinctement l'architecture d'un simulateur de conduite, nous détaillons dans ce chapitre les différentes technologies constituant les systèmes d'affichage, leurs applications et insistons sur leurs différences et limitations. Nous verrons en particulier que la mise en place (choix, installation, calibration) d'un système d'affichage pour un simulateur de conduite produisant une bonne immersion est une tâche complexe.

3.1 Introduction aux simulateurs de conduite

Avant de détailler les différents systèmes d'affichage et leurs technologies, voici une description générale de l'architecture matérielle et logicielle d'un simulateur de conduite.

Un simulateur est composé d'un poste de conduite, réalisé à partir d'un volant, d'un pédalier et d'un levier de vitesse ou plus généralement d'un cockpit de véhicule réel. Tous ces éléments sont autant d'interfaces haptiques visant à restituer les efforts ressentis lors de la conduite d'un vrai véhicule (vibrations, retours d'efforts volant, etc.). Un système de visualisation affiche les images calculées en temps réel en fonction de la position du conducteur à chaque instant, sur un ou plusieurs écrans ou encore dans un casque de réalité virtuelle (placé sur la tête du conducteur). Enfin, pour les simulateurs dynamiques, une plate-forme est utilisée pour restituer les sensations de mouvement propre ressenties lors des accélérations longitudinales et latérales du véhicule.

Renault développe depuis 1990, le logiciel SCANeR©(Simulation de Conduite Automobile Normalisée En Réseau), aujourd'hui commercialisé par la société Oktal dans sa deuxième version. Il est constitué d'un ensemble de modules répartis selon une architecture distribuée (figure 3.1). Chacun de ces modules réalise une tâche particulière de la simulation, comme par exemple :

- l'acquisition des commandes du conducteur ;
- le calcul des images en temps réel ;
- le calcul de la dynamique du véhicule interactif (réagissant aux actions du conducteur) ;
- le calcul de l'environnement sonore ;
- la génération d'un trafic autonome ;
- etc.

Les actions effectuées par le conducteur sont lues par le module d'acquisition. A partir de ces informations, le modèle avancé de dynamique de véhicule (MADA) calcule la position, la vitesse et l'accélération du véhicule dans le monde virtuel. La stratégie de commande transforme alors l'accélération donnée par le modèle en instructions de déplacement pour la plate-forme. Le module visuel met à jour les images générées en temps réel à partir de la position du véhicule dans le monde réel et le module sonore génère les sons en fonction du régime moteur et éventuellement des bruits ambients de l'environnement (vent, trafic, etc.). L'ensemble de ces modules est contrôlé depuis une machine de supervision.

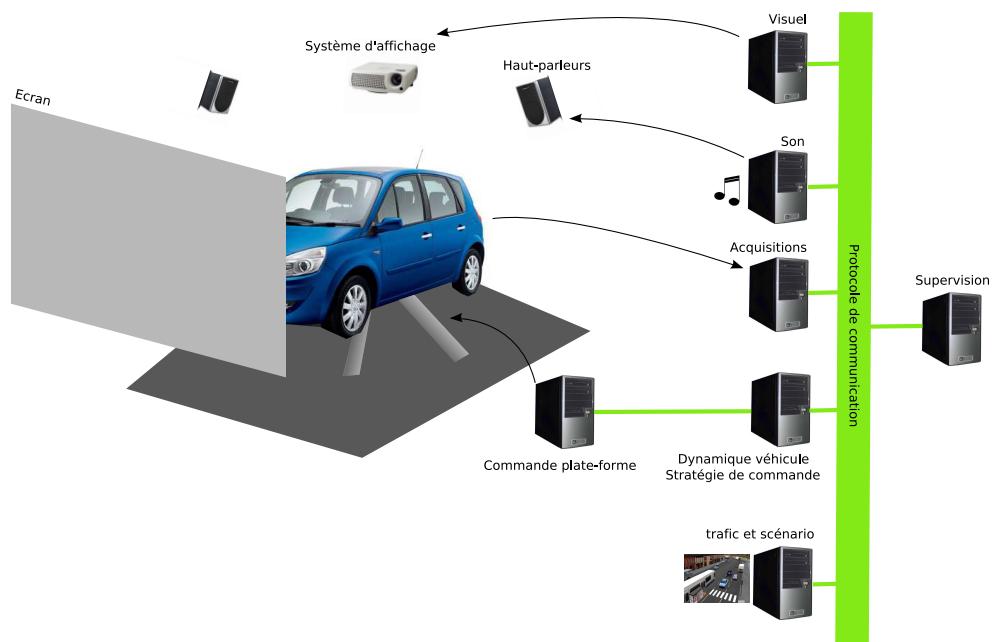


FIG. 3.1 – Architecture logicielle de SCANeR II

3.2 Les systèmes de vidéo-projection

Les systèmes d'affichage sont basés sur différentes technologies que nous allons présenter et comparer afin de préciser leurs applications potentielles dans le contexte de la simulation de conduite.

3.2.1 Technologie CRT (Cathode Ray Tube)

Inventée en 1897 par Karl Ferdinand Braun, le tube cathodique (fig. 3.2) est la technologie la plus ancienne. On la retrouve à grande échelle dans les écrans d'ordinateurs, téléviseurs (remplacés peu à peu depuis le début des années 2000 par des écrans plasma et LCD) et oscilloscopes.

Cette technologie consiste à envoyer des flux d'électrons (3) (générés par la cathode (2) par effet thermoïonique) à haute vitesse vers l'anode (5) recouverte d'une matière phosphorescente. Les électrons sont focalisés par des bobines magnétiques (4) et déviés électrostatiquement par des électrodes (1). Les phosphores ainsi excités émettent de la lumière. Afin de restituer des images en couleur, les vidéo-projecteurs CRT possèdent trois tubes cathodiques pour chacune des couleurs primaires (rouge, vert et bleu), ainsi que des lentilles de convergence pour projeter une image nette sur un écran distant.

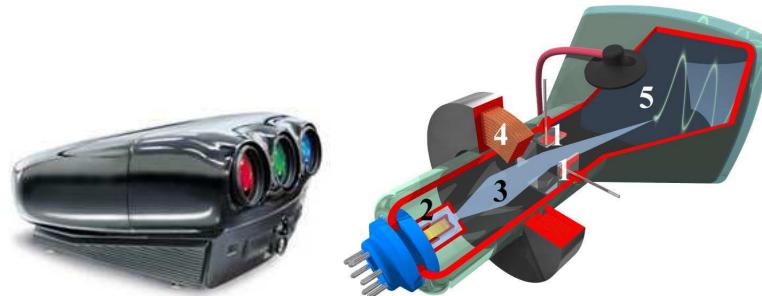


FIG. 3.2 – Principe de fonctionnement d'un vidéo projecteur CRT

3.2.2 Technologie LCD (Liquid Cristal Display)

La technologie LCD est un dispositif filtrant passif utilisant deux principes fondamentaux : la polarisation de la lumière et la biréfringence des cristaux liquides. Un écran LCD est composé de deux filtres polarisants (formant un angle de 90 degrés) et de deux plaques de verres renfermant des cristaux liquides.



FIG. 3.3 – Principe de fonctionnement d'un vidéo projecteur LCD

Les plaques de verres comportent des matrices d'électrodes transparentes. Soumises à une différence de potentiel, elles modifient l'orientation des molécules, et ainsi la transparence du dispositif. Ce système transmissif doit donc être rétro-éclairé par une lampe halogène (lumière de forte puissance et dont la température de couleur est proche de celle du soleil, soit environ 6000 Kelvin). Pour obtenir une image colorée, la matrice comporte un filtre en bandes verticales rouges, vertes et bleues. Il existe des systèmes avec trois matrices LCD pour chacune des composantes de couleur RVB, l'image étant

alors restituée en sommant les trois composantes à travers un prisme (fig. 3.3).

3.2.3 Technologie DLP (Digital Light Processing)

La technologie DLP a été développée par la société Texas Instruments. Cette technologie est dite réflective : chaque pixel correspond à un micro miroir (DMD pour Digital Micromirror Device) (figure 3.4). Un DMD oscille à haute fréquence (jusqu'à 50 kHz) sous l'action d'un champ électrique, réfléchissant ou non la lumière vers l'écran. Ainsi, l'intensité du pixel est modulée entre 0 et 100% et une image en niveaux de gris est obtenue. L'obtention de la couleur se fait en filtrant la source de lumière par un filtre rotatif tricolore RVB.

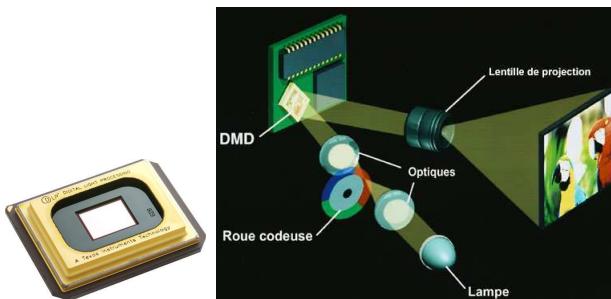


FIG. 3.4 – Puce DMD (gauche), principe de fonctionnement d'un vidéo projecteur DLP (droite)

3.3 Systèmes de visualisation

La réalisation d'un simulateur de conduite immersif nécessite un système de visualisation adapté. Il peut être un casque issus de la réalité virtuelle ou encore basé sur l'utilisation de vidéo-projecteurs couplés à un (ou plusieurs) écran.

3.3.1 Écrans de projection

Dans la majorité des cas, le système de visualisation frontal d'un simulateur de conduite (figure 3.5) se compose de trois écrans plats ou d'un écran cylindrique. Un écran plat peut également être situé à l'arrière du cockpit pour la rétrovision. Pour un simulateur dynamique, ces écrans sont parfois solidaires de la plate-forme dynamique et nous verrons que dans le cas contraire, la mise en place du système d'affichage est plus complexe.

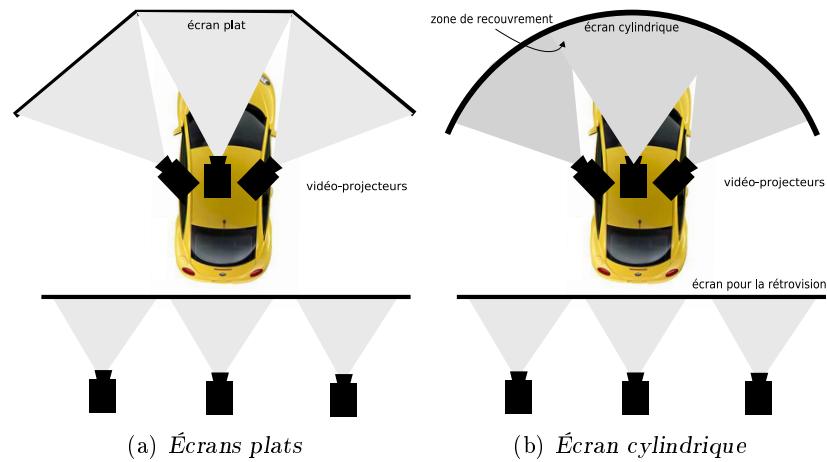


FIG. 3.5 – Systèmes de visualisation par vidéo-projection

3.3.2 Casque de réalité virtuelle

Les casques de réalité virtuelle offrent une vision stéréoscopique (un écran affichant une image différente pour chaque œil). Couplés à un capteur de tête, ils peuvent être utilisés dans le cadre de la simulation de conduite pour l'affichage de cockpits virtuels de véhicule. Ainsi, il est possible d'évaluer la visibilité (masquage de la scène par des éléments du cockpit) ou encore l'éclairage intérieur du véhicule en situation de conduite interactive.



FIG. 3.6 – Casque de réalité virtuelle à large champs de vision (SEOS)

La réalisation d'un casque de qualité, en terme de résolution, champ de vision, netteté des images, répartition du poids est de nos jours encore délicate. Aussi, ce système est dit "intrusif" car il nécessite de le fixer sur la tête et de le calibrer pour chaque utilisateur (réglage des fixations, écart inter-pupillaire, netteté de l'image en fonction de la vision). Le casque SEOS,

visible sur la figure 3.6, offre un large champ de vision (120° dont 40° de recouvrement).

3.4 Étude des systèmes d'affichage

3.4.1 Comparatifs des technologies de vidéo-projection

Les différentes technologies CRT, LCD et DLP présentent des caractéristiques différentes dont nous allons faire un bref comparatif. Pour chacun des critères, nous insistons sur les limitations et le périmètre de validité des systèmes d'affichage en particulier pour le domaine de la simulation de conduite.

- **Puissance lumineuse** : La mesure de la puissance lumineuse d'un vidéo-projecteur se mesure en lumen ANSI (American National Standards Institute). Ce critère consiste à mesurer neuf points répartis uniformément sur la surface éclairée. Elle fournit une bonne représentativité des capacités d'un vidéo-projecteur en terme de puissance lumineuse mais aussi d'uniformité de l'éclairage sur toute la surface éclairée. Le ratio entre les mesures ne doit pas excéder 30% pour un vidéo-projecteur de qualité. Les technologies LCD et DLP fournissent respectivement un flux de 2500 et 5000 lumens tandis qu'un vidéo-projecteur CRT ne fournit que 200 lumens.

Nous avons effectué des mesures de luminance sur les écrans utilisés sur le simulateur d'éclairage Renault, et nous avons obtenu des valeurs de l'ordre de 70 cd/m^2 . Comme nous l'avons vu précédemment, ceci est très inférieur à la luminance maximale que l'oeil peut percevoir (10^{+8} cd/m^2). Ainsi, l'éblouissement naturel du conducteur par les phares des véhicules du trafic en conduite de nuit n'est pas possible sur simulateur de conduite (contrairement à la réalité).

- **Contraste** : Il existe deux méthodes pour mesurer le contraste qui représente le rapport de luminance entre le niveau de blanc le plus pur et le niveau de noir le plus extrême. Plus le contraste est élevé, plus le vidéo-projecteur sera capable d'afficher des nuances de couleurs subtiles. La première méthode, appelée "*Full On/Off*" consiste simplement à mesurer le rapport de luminance lorsque l'on affiche une image blanche (*Full On*) et un image noire (*Full Off*). Cette méthode n'est pas très représentative car elle ne tient pas compte de la lumière parasite. En effet, afficher une image contenant à la fois du blanc et du noir, affecte le contraste de l'image. La deuxième méthode (ANSI), consiste donc à afficher un damier de 16 rectangles noirs et blancs et à calculer le rapport des moyennes des luminances des rectangles blancs et noirs pour obtenir une valeur de contraste représentative. La technologie

CRT offre le meilleur taux de contraste On/Off (environ 1000 :1) car le noir est très profond. La technologie DLP, réflective, est meilleure (400 :1) que la technologie LCD, transmissive (200 :1) car les cristaux liquides ne sont pas capables de "bloquer" totalement la lumière émise par la lampe. Un mauvais taux de contraste (i.e. un noir lumineux) peut être très gênant, en particulier pour la simulation d'éclairage où les niveaux d'éclairage que l'on visualise sont faibles et peuvent ainsi être "noyés" dans la lumière parasite.

- **Résolution :** De nos jours, les vidéo-projecteurs courants, toutes technologies confondues, atteignent une résolution de 1280x1024 pixels (SXGA). Il est plus coûteux d'avoir des résolutions supérieures avec la technologie DLP étant donné la difficulté à produire des DMD contenant beaucoup de micro-miroirs. La résolution d'un vidéo-projecteur peut être une limitation critique pour un simulateur de conduite : en affichant 1280 pixels sur un écran de plusieurs mètres de largeur, un pixel recouvre une surface de plusieurs millimètres, ainsi l'affichage est loin de correspondre à la résolution de l'oeil (environ une minute d'arc). Cette limitation engendre également des problèmes de crénelage (aliasing) des images même si les cartes graphiques possèdent aujourd'hui des filtres efficaces pour adoucir les images.
- **Pixélisation :** La technologie LCD présente des problèmes de pixélisation de l'image qui provient du fait qu'il existe un espace entre les cristaux liquides. Ainsi, seul environ 60% de l'image est réellement couverte par le vidéo-projecteur. Toutefois, si l'observateur est suffisamment éloigné de l'écran, ces espaces ne sont pas visibles car intégrés par le système visuel humain. La technologie D-ILA des projecteurs DLP offre de biens meilleurs résultats avec un taux de couverture de 90%.
- **Taux de rafraîchissement :** La fréquence de rafraîchissement des technologies CRT et LCD/DLP n'est pas comparable car la technologie CRT utilise un système de balayage alors que les LCD/DLP mettent à jour l'ensemble des pixels au même instant. Ainsi avec la technologie LCD/DLP, un taux de rafraîchissement de 60 Hz est tout à fait satisfaisant, alors que pour la technologie CRT 75 Hz est la fréquence minimale pour éviter la fatigue visuelle due à des effets de scintillement. Pour la vision stéréoscopique, il existe deux principes. La stéréovision active (affichage successif d'une image pour l'oeil droit puis à l'instant d'après pour l'oeil gauche) qui nécessite des vidéo-projecteurs onéreux avec une fréquence élevée (120 Hz, soit 60Hz pour chaque oeil). La stéréovision passive, consiste à afficher les deux images avec deux vidéoprojecteurs ayant un filtre polarisé, des lunettes polarisées servant à

filtrer l'image pour l'oeil correspondant. Dans ce cas, une fréquence standard de 60 Hz est suffisante mais nécessite deux fois plus de projecteurs.

- **Rémanence** : Certains vidéo-projecteurs LCD souffrent d'une rémanence élevée (jusqu'à 16 ms). La rémanence provient du temps de réponse des cristaux liquides pour changer d'état (de couleur, pour s'allumer ou s'éteindre). Ce phénomène est problématique pour la simulation de conduite car des effets de flous apparaissent lorsque l'on affiche des images à une fréquence élevée (60Hz ou plus). Il existe aujourd'hui des matrices LCD ayant une rémanence faible (de l'ordre de 2 ms), qui est obtenue par une fréquence d'affichage plus élevée (100 Hz) et en insérant une image noire (non perceptible) entre deux images.
- **Espace des couleurs (Gamut)** : Les vidéos projecteurs ne sont pas capables de reproduire toutes les couleurs visibles par le système visuel humain, mais seulement un sous-espace, appelé *gamut*. Sur la figure 3.7, sont représentées les couleurs affichables par un vidéo-projecteur CRT qui, on le voit, est un ensemble très restrictif. La technologie LCD permet d'obtenir des couleurs plus vives et ainsi de couvrir une espace de couleur plus grand. Néanmoins, l'uniformité des couleurs à l'intérieur de l'espace n'est pas respectée rendant certaines nuances plus difficiles à percevoir.

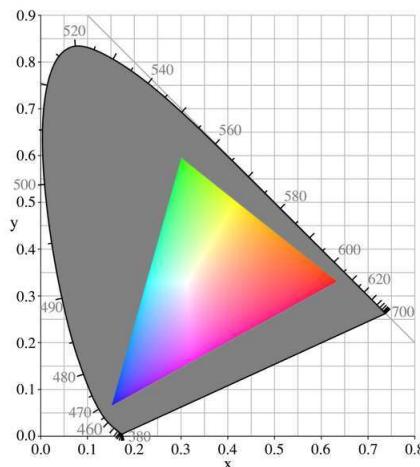


FIG. 3.7 – Gamut d'un écran CRT

Le choix de telle ou telle technologie pour la réalisation d'un système d'affichage pour un simulateur de conduite dépend des contraintes de chaque application. Par exemple, pour la simulation d'éclairage (voir chapitre 4.4),

la qualité du "noir" sera privilégiée en utilisant des vidéo-projecteurs DLP. Pour la visualisation de cockpits virtuels, un casque de réalité virtuelle (avec un écran LCD) couplé à un capteur de position est préférable. La technologie CRT est toutefois de moins en moins répandue (comparativement aux technologies LCD et DLP) de par sa faible luminosité et son encombrement.

3.4.2 Limitations des systèmes d'affichage

La mise en place d'un système d'affichage pour la simulation de conduite est un exercice délicat. En effet, choisir un bon système d'affichage dédié à une utilisation particulière nécessite de connaître dans les détails les caractéristiques des différents éléments constituant le système d'affichage. Nous allons exposer une liste non exhaustive des problématiques fréquemment rencontrées :

- **Calibration** : La réponse en terme de luminance des systèmes d'affichage n'est pas linéaire : si l'on affiche un dégradé entre 0 et 1 à pas constants, la luminance obtenue en sortie correspond à une courbe gamma. Ceci a pour effet d'écraser les niveaux faibles de lumière. Afin d'obtenir une réponse en luminance linéaire, il faut donc appliquer une courbe gamma inverse sur les pixels en entrée (P_e).

$$P_s = P_e^{\frac{1}{\gamma}}$$

La valeur du gamma (γ) dépend des caractéristiques du système d'affichage qu'il convient d'évaluer à l'aide d'un spectro-photomètre (ou visuellement de manière moins précise à l'aide de mires). Il est généralement situé entre 1,7 et 2,5.

- **Écran courbe** : Les écrans courbes (figure 3.8) permettent d'obtenir un champ de vision horizontal très large (210° pour le simulateur d'éclairage Renault). Lorsque l'on projette une image sur un écran courbe, il est nécessaire de calculer la déformation de l'image en fonction des caractéristiques de l'écran. Cette déformation se calcule pour une position d'observateur fixe donnée. Ainsi, les corrections appliquées sont fausses si l'observateur se déplace par rapport à l'écran ou tout simplement pour un autre utilisateur.

Néanmoins, si l'observateur est en mouvement, il est possible de mettre à jour en temps réel l'image projetée en fonction de la position de l'observateur en utilisant un tracker de tête.

- **Compensation visuelle** : Pour un simulateur dynamique, il n'est pas



FIG. 3.8 – Simulateur de conduite immersif avec écran courbe (210° de champs de vision horizontal)

toujours possible que les écrans et la plate-forme dynamique soient solidaires pour des raisons de poids et d'encombrement. C'est le cas du simulateur CARDS (figure 3.9), pour lequel il est nécessaire d'effectuer une compensation visuelle car le point de vue de l'observateur se déplace en fonction des mouvements de la plate-forme alors que les écrans sont fixes par rapport au sol. Cette compensation se calcule relativement simplement mais un déphasage peut être présent entre les mouvements de l'observateur et la mise à jour des informations visuelles. Ce phénomène, connu sous le nom de "*transport delay*" peut atteindre plusieurs millisecondes et devient alors visible et gênant pour l'observateur. Il est généralement le résultat de l'accumulation de plusieurs étapes prenant un temps non négligeable comme l'inertie de la plate-forme, la lecture de la position de la plate-forme sur le réseau, le calcul pour la mise à jour de l'image, etc. Une étude a été menée [MDK02] montrant que les observateurs ont une tolérance vis à vis de ce délai inévitable, et que cette tolérance change si l'observateur est actif ou passif (respectivement 500ms et 100ms).

- **Edge blending** : Pour avoir une bonne sensation d'immersion, il est nécessaire d'avoir un large champ de vision, ce qui ne peut se faire qu'en utilisant un écran courbe ou trois écrans plats. Ceci exige l'utilisation de plusieurs vidéo-projecteurs (trois en général). La jonction entre les écrans est alors délicate. Il existe une zone de recouvrement appelée "edge blending" pour laquelle il faut calibrer le mélange des images au pixel près afin qu'il n'y ait pas de flou du au décalage des images ainsi qu'un accroissement de la luminance du à la superposition des images générées par deux projecteurs différents dans cette zone.
- **Conflit accommodation-convergence** : Dans la réalité, l'accommodation et la vergence du système visuel humain sont des phéno-



FIG. 3.9 – Simulateur de conduite dynamique CARDS

mènes étroitement liés (car contrôlés par une même triade musculaire). L'accommodation, phénomène monoculaire, consiste à faire la mise au point sur l'objet observé afin d'en avoir une perception nette. La vergence, phénomène binoculaire, consiste à faire converger les deux yeux sur l'objet observé. Ces deux phénomènes, par les contractions musculaires, donnent des informations sur la distance [MWT99] entre les objets et l'observateur, ceci pour des distances proches (inférieures à 3m).

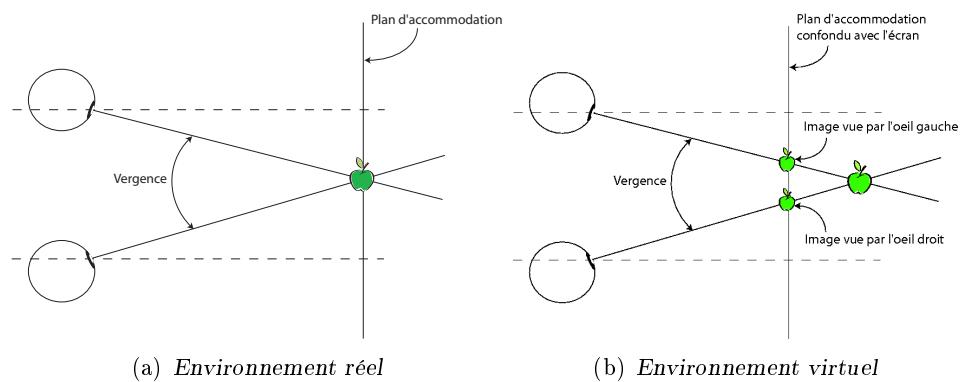


FIG. 3.10 – Fonctionnement de la convergence et l'accommodation en environnement réel (a) et virtuel (b). (a) La vergence et l'accommodation se font sur l'objet. (b) la vergence se fait sur l'objet mais l'accommodation sur l'écran.

En réalité virtuelle, pour un système d'affichage stéréoscopique, l'ac-

comodation est fixe et se fait sur le plan de l'écran (afin de voir les images nettes), mais la convergence varie, et se fait sur l'objet affiché en trois dimensions (figure 3.10). Il y a donc un conflit accommodation-convergence, qui peut entraîner une gêne, voir des maux de têtes en cas d'utilisation prolongée [MMWR93]. La perception des distances est elle aussi perturbée, bien que dans le cadre de la simulation de conduite, les plus proches objets affichés sont éloignés de plusieurs mètres, sauf en cas d'affichage d'un cockpit virtuel.

3.5 Conclusion

Nous avons présenté les systèmes d'affichage employés dans la simulation de conduite, ainsi que les technologies utilisées. Nous avons vu, de part les diverses limitations, qu'il est très difficile de créer un environnement visuel proche de la réalité. Il faut parfois prendre en compte ces limitations (calibration gamma, éblouissement, ...) dans les modèles de synthèse d'image afin de générer des images réalistes. Néanmoins, un système d'affichage adapté et correctement calibré, permet une immersion satisfaisante des conducteurs sur simulateur de conduite.

Chapitre 4

La synthèse d'image

Contrairement à la photographie qui est une capture du monde réel, la synthèse d'image a pour but la création d'images numériques par ordinateur. Son développement est étroitement lié à l'évolution de la puissance de calcul des ordinateurs, et il faudra attendre les années 1950 pour qu'elle commence à faire son apparition avec un système de contrôle aérien (appelé SAGE) développé par le MIT (*Massachusetts Institute of Technology*). En 1964, Ivan Sutherland met au point un système "SketchPad" de dessin vectoriel à deux dimensions [Sut64] par ordinateur à l'aide d'un crayon optique et d'un tube cathodique. Dans les années 1970, les premiers logiciels de DAO (Dessin Assisté par Ordinateur) font leurs apparitions dans plusieurs sociétés (General Motors, Bell, NASA) et les terminaux graphiques à bas prix (de marque tektronix) se démocratisent. C'est à partir des années 80-90 que la synthèse d'image connaît un essor important, les outils informatiques disponibles permettant de générer des images calculées par des modèles complexes (lancer de rayon, etc.). Aujourd'hui, les ordinateurs sont équipés de matériels spécifiques dédiés à la génération d'images en temps réel (interactif) et sont utilisés dans de nombreux domaines (imagerie médicale, jeux vidéos, films d'animations, etc.) ainsi que dans l'industrie automobile comme outil de conception numérique.

Dans ce chapitre, les modèles de synthèse d'images simulant des scènes visuelles où les objets sont décrits dans un espace à trois dimensions (puis projetés en 2D sur un écran) sont présentés. Nous insistons plus particulièrement sur les techniques de modélisation et de rendu concernant la réflexion des surfaces (illumination locale) ainsi que les échanges lumineux entre les objets (illumination globale). Nous présentons l'architecture des cartes graphiques actuelles, détaillons leurs fonctionnalités et expliquons, à travers la présentation du simulateur d'éclairage Renault, comment elles peuvent être de véritables outils de conception pour la mise au point des projecteurs automobiles.

4.1 Modèle d'illumination local

Un modèle d'illumination local décrit l'interaction entre la lumière et une surface en un point donné indépendamment de son environnement. Nous pouvons distinguer dans la littérature deux types distincts : les modèles implicites et explicites. La micro-géométrie d'une surface, qui joue un rôle important dans les propriétés de réflexion de la lumière, n'est pas prise en compte ou est approximée par des modèles statistiques dans les modèles implicites phénoménologiques. Ces modèles, souvent empiriques, ont l'avantage d'avoir un nombre réduit de paramètres mais ne s'appliquent qu'à une catégorie limitée de matériaux. Les modèles explicites se basent sur une représentation détaillée de la micro-géométrie et calculent les échanges radiatifs qui ont lieu à petite échelle sur la surface afin de définir un modèle d'illumination local qui comporte en général un grand nombre de paramètres qu'il est difficile de régler si ce n'est par la connaissance très approfondie des caractéristiques du matériau. Les principaux modèles et les notions associées vont être présentés dans ce chapitre.

4.1.1 Fonction de distribution de réflectance bidirectionnelle (BRDF)

La lumière interagit avec les surfaces selon plusieurs principes qui peuvent intervenir simultanément :

- Réflexion de la lumière par la surface,
- Absorption de la lumière par la surface,
- Transmission de la lumière par la surface.

Les deux premiers phénomènes sont caractérisés par une BRDF (Fonction de distribution de réflectance bidirectionnelle) qui exprime le rapport (nommé albédo) de la luminance réfléchie par le matériau et l'éclairement reçu par la surface correspondante. Une BRDF dépend des paramètres suivants : direction incidente de la lumière, direction d'observation de la surface, longueur d'onde et polarisation de la lumière. Les deux derniers paramètres n'étant nécessaires qu'à la description de la BRDF comportant des phénomènes lumineux complexes (diffraction, etc.), la BRDF (figure 4.1) est généralement exprimée par une fonction qui ne dépend que de quatre paramètres, à savoir :

- la direction incidente de la lumière $\vec{\omega}_i(\theta_i, \varphi_i)$,
- la direction d'observation $\vec{\omega}_o(\theta_o, \varphi_o)$.

Elle s'exprime en Sr^{-1} et s'écrit de la manière suivante :

$$f(\theta_i, \phi_i, \theta_o, \phi_o) = \frac{L_o(\theta_o, \phi_o)}{E_i(\theta_i, \phi_i)} \text{ avec : } E_i(\theta_i, \phi_i) = L_i(\theta_i, \phi_i) \cdot \cos(\theta_i) \cdot d\omega_i$$

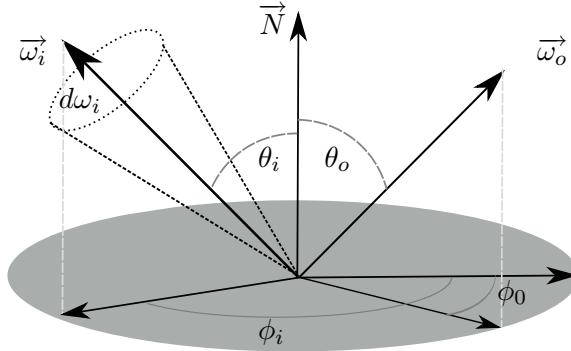


FIG. 4.1 – Représentation de la BRDF qui exprime le rapport entre la luminance réfléchie et l'éclairement reçu par la surface.

L'absorption de la lumière par la surface est implicite car comprise dans le coefficient de la BRDF (compris entre 0 et 1). Afin de caractériser les matériaux translucides, on définit une BTDF (Bidirectionnal Transmission Distribution Function) de la même manière que l'on définit une BRDF. Le coefficient exprime alors, non pas la réflexion de la lumière, mais la quantité de lumière traversant le matériau.

D'après cette expression, la luminance réfléchie par la surface dans une direction d'observation donnée consiste à intégrer l'éclairement arrivant sur la surface sur tout l'hémisphère et s'écrit :

$$L_o(\theta_o, \phi_o) = \int_{\phi_i=0}^{2\pi} \int_{\theta_i=0}^{\frac{\pi}{2}} f(\theta_i, \phi_i, \theta_o, \phi_o) \cdot L_i(\theta_i, \phi_i) \cdot \cos(\theta_i) \cdot \sin(\theta_i) \cdot d\theta_i \cdot d\phi_i$$

Elle doit respecter deux principes fondamentaux que sont la réciprocité d'Helmotz et la conservation d'énergie :

- **Réciprocité d'Helmotz** : si l'on inverse la direction incidente et réfléchie, le rapport entre la luminance incidente et réfléchie reste inchangé : $f(\theta_i, \phi_i, \theta_o, \phi_o) = f(\theta_o, \phi_o, \theta_i, \phi_i)$.
- **Conservation d'énergie** : Une surface ne peut réfléchir plus de lumière qu'elle n'en reçoit, soit :

$$\rho(\theta_i, \phi_i) = \int_{\Omega} f(\theta_i, \phi_i, \theta_o, \phi_o) \cdot \cos(\theta_o) \cdot d\omega_o \leq 1$$

Où $\rho(\theta_i, \phi_i)$ est le coefficient de réflectance pour une direction incidente ω_i donnée.

Il est possible de diminuer la dimension de la BRDF ou encore le nombre de données nécessaires à sa description. En effet, certains matériaux ont une réflexion isotropique (BRDF invariante par rotation horizontale de la surface), ce qui permet de réduire le nombre de dimension de la BRDF en négligeant l'angle ϕ_o . D'autre part, certains matériaux réfléchissent la lumière de manière équivalente vers la partie droite et vers la gauche de l'hémisphère (réflexion symétrique), ainsi, il est possible de diviser par deux le nombre de données nécessaires à la description de la BRDF.

La BRDF d'un matériau est généralement mesurée par un gonio spectrophotomètre. Les systèmes permettant de mesurer une BRDF présentent quelques limites. Ils introduisent du bruit dans les mesures obtenues. Celui-ci devient même largement prépondérant sur le signal pour les angles rasant étant donné la valeur de l'angle solide dans ces zones (la surface éclairée devient presque invisible par la cellule photométrique). Une autre limite provient du fait qu'il est impossible, pour des raisons mécaniques évidentes, de mesurer la BRDF pour une position équivalente d'éclairement et d'observation ($\vec{\omega}_i = \vec{\omega}_o$). Les données sont donc interpolées et non mesurées pour ces angles. Pour palier ces inconvénients, la BRDF d'un matériau peut être approximée par différents modèles, ce que nous allons voir ci-après.

4.1.2 Modèles de réflectance

Les modèles de réflectance visent à décrire mathématiquement la BRDF d'un matériau. Ces modèles peuvent être empiriques ou basés sur des approches statistiques.

Pour un matériau qui réfléchit la même quantité de lumière dans toutes les directions (purement diffus), l'expression de la BRDF est la suivante :

$$f_{diffus} = \frac{\rho_d}{\pi}$$

Où ρ_d est une constante correspondant à la réflectance de la surface (la valeur π sert à normaliser la fonction).

A ce terme diffus, Phong [Pho75] a proposé en 1975 d'ajouter une composante spéculaire. Afin que le modèle respecte le principe de conservation d'énergie, Lewis [Lew93] l'a quelque peu modifié en 1993. La BRDF associée à ce modèle est donnée par la relation suivante :

$$f_{phong} = \frac{\rho_d}{\pi} + \rho_s \frac{n+2}{2\pi} \cos^n \alpha$$

Avec :

- ρ_d : Coefficient de réflectance diffus.
- ρ_s : Coefficient de réflectance spéculaire.
- α : Angle entre la réflexion spéculaire idéale et le vecteur de vue.
- n : Coefficient de spécularité de la surface. Plus n est grand, plus la réflexion spéculaire de la surface se resserre autour du vecteur réfléchi.

Il existe des modèles plus complexes, comme par exemple, le modèle empirique de Ward [War92] qui permet de représenter des surfaces anisotropes à partir de distributions gaussiennes paramétrables pour contrôler la déformation de la réflexion. Parmi les modèles statistiques, nous pouvons citer le modèle de Cook-torrance [CT92] (1992) qui considère qu'une surface est composée d'une distribution de micro-facettes sur lesquelles sont appliquées les lois de l'optique géométrique en tenant compte de l'auto-masquage et de l'auto-ombrage pouvant intervenir entre les micro-facettes.

4.2 Modèle d'illumination global

Le modèle d'illumination global formalise les échanges lumineux existants entre les sources de lumière et les objets d'une scène, y compris les interréflexions entre objets. Certains modélisent le trajet direct de la lumière, d'autres considèrent le parcours inverse ou encore utilisent une description en facettes des objets pour en calculer les échanges radiatifs. Les principales méthodes de synthèse d'image basées sur de tels modèles sont détaillées dans ce chapitre.

4.2.1 Lancer de rayon

Le principe du lancer de rayon consiste à calculer la luminance pour chaque pixel de l'écran en parcourant le trajet inverse de la lumière (figure 4.2). Le rayon primaire est issu du point de vue et passe au milieu d'un pixel.

Lorsque le rayon intersecte une surface de la scène 3D, on calcule un modèle d'illumination local au point d'intersection, on regarde si ce point est dans l'ombre d'une autre surface vis à vis des sources de lumières présentes, puis un rayon réfléchi (dit secondaire) dans la direction spéculaire idéale est relancé. Si la surface est transparente, un rayon transmis est également lancé dans la direction de transmission spéculaire idéale. On répète ces

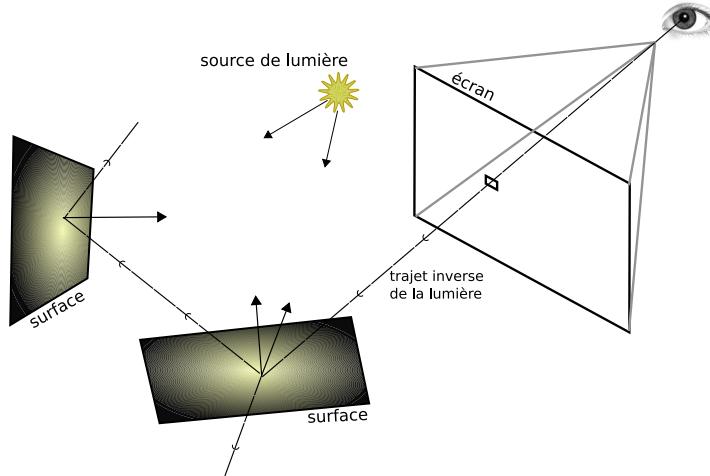


FIG. 4.2 – Principe du lancer de rayons

étapes de manière récursive jusqu'à un nombre limite défini par l'utilisateur. A chaque étape, on ajoute les contributions lumineuses à la couleur finale du pixel. Le résultat ne tient donc compte que des éclairages directs par des sources de lumière ponctuelles et des reflets spéculaires entre surfaces. Ces contributions sont obtenues en appliquant le modèle de Whitted [Whi80] qui, pour m sources de lumière, exprime le bilan énergétique en terme d'intensité :

$$I_x = I_a + I_e + \sum_{j=1}^m \left(\rho_d (\vec{N} \cdot \vec{L}_j) + \rho_s (\vec{N} \cdot \vec{H}_j)^n \right) I_j + \rho_s I_s + \rho_t I_t$$

Avec :

- I_a : Intensité de la lumière ambiante.
- I_e : Intensité de Lumière émise par la surface.
- I_s et I_t : Respectivement intensités de la contribution lumineuse dans les directions de réflexion et transmission.
- \vec{N} : Normale à la surface au point x .
- \vec{L}_j : Direction de la lumière j .
- \vec{H}_j : Vecteur *halfway* (somme du vecteur \vec{L}_j et du vecteur de vue).
- n : Coefficient de spécularité de la surface.
- ρ_d , ρ_s et ρ_t : Coefficient de réflectance diffus, spéculaire et de transmittance spéculaire.

Il existe de nombreuses évolutions de cet algorithme de rendu. Par exemple, afin réduire les problèmes de crénelage (aliasing), il est possible de lancer plusieurs rayons pour chaque pixel et de moyenner le résultat obtenu pour

chacun de ces rayons. En effet, un seul rayon n'est en général pas suffisant pour intégrer les phénomènes lumineux présents sur la totalité de la surface représentée en ce pixel. Des techniques stochastiques (lancer de rayons distribué), permettent d'intégrer la notion de BRDF sur les surfaces, ainsi, un rayon ne sera pas réfléchi dans une seule direction (spéculaire idéale) mais dans plusieurs directions générées aléatoirement en fonction de la BRDF de la surface.

La simulation d'éclairage de scène routière de nuit par des projecteurs automobiles, sous différentes conditions météorologiques n'est pas très répandue dans la littérature. On relève deux travaux majeurs dans ce domaine. La technique de rendu utilisée (lancer de rayon) dans ces deux travaux n'est pas temps réel, et les résultats obtenus n'ont pas bénéficiés de validation.



FIG. 4.3 – Simulation d'éclairage automobile. T. Nishita (gauche), Optis (droite)

La première étude menée sur le sujet par T. Nishita, [NKON90] a permis d'obtenir des images d'éclairage de route humide de qualité visuelle satisfaisante. Elle est basée sur la modélisation de l'état de la surface (sèche, humide, flaque d'eau, etc.) et utilise des modèles d'éclairage adaptés pour chaque zone ainsi que du *bump-mapping* (technique de perturbation de normales détaillée au chapitre 4.3.4) pour simuler la granularité de la route. Plus récemment, la société Optis a travaillé sur l'éclairage de route humide. Elle emploie la même technique de rendu que T. Nishita et intègre des mesures de réflectance (BRDF) de bitume réel. Du bruit est utilisé pour simuler la granularité de la route. Les images ainsi obtenues, bien que basées sur des mesures de matériau réel offrent un réalisme visuel moindre que celui obtenu par le rendu de T. Nishita (figure 4.3).

4.2.2 Lancer de photon

Technique inventée par Arvo en 1986 [Arv86], le lancer de photon consiste à parcourir le trajet direct de la lumière, afin de simuler des phénomènes de concentration de lumière appelés caustiques (figure 4.4). Des photons, transportant de l'énergie, sont lancés depuis les sources de lumière à travers la scène de façon aléatoire. Lorsqu'un photon intersecte une surface, une partie de son énergie est stockée sur la surface et le reste de l'énergie est réfléchie ou réfractée. Une carte d'illumination, sous forme de texture, est attribuée à chacune des surfaces. Une seule passe de lancer de rayon vient enfin lire les informations contenues dans ces cartes afin de calculer la couleur finale en chaque pixel.

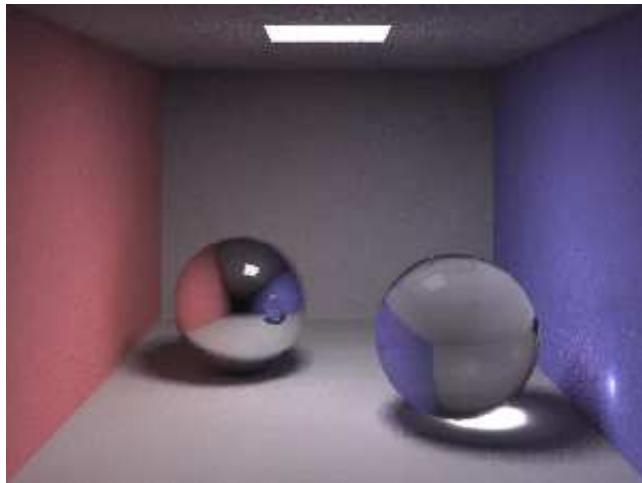


FIG. 4.4 – Simulation des phénomènes caustiques avec la technique de lancer de photon (*Dimitri Fischler*).

En 1996, Jensen [Jen96] s'inspire de cette méthode et y ajoute la notion de *photon-map* (figure 4.5). Cette structure de données, représentée sous la forme d'un arbre binaire de recherche *kd-tree*, contient les photons interagissant avec les surfaces de la scène. Lorsqu'un photon émis par une source de lumière atteint une surface, celui-ci est stocké dans la photon-map en fonction de sa position dans l'espace avec sa direction incidente et le flux énergétique lui correspondant. Une fonction de probabilité indique si le photon est ré-émis et dans quelle direction en fonction de la BRDF de la surface. Une deuxième photon-map est générée selon le même principe pour simuler les phénomènes caustiques qui à eux seuls nécessitent un très grand nombre de photons.

L'étape finale consiste à effectuer une passe de lancer de rayons et pour

chaque rayon intercepté, une estimation de l'éclairage est faite à partir de la photon-map : au point d'intersection, tous les photons contenus dans une sphère de rayon déterminé, sont comptabilisés et couplés à la BRDF de la surface pour calculer la couleur finale du pixel.

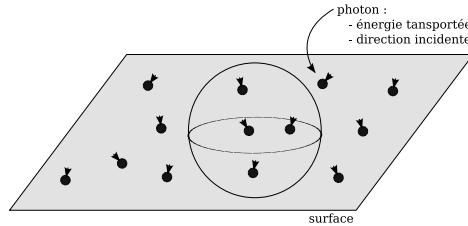


FIG. 4.5 – Photon map

Avec cette méthode, il est nécessaire de lancer un grand nombre de photons pour éviter des problèmes de non-uniformité de l'éclairage sur les surfaces. La taille de la sphère doit également être choisie de manière adaptée, car un rayon trop grand entraînerait un lissage de l'image ainsi que des "ba-vures" entre les différentes surfaces.

Remarque : Cette technique de rendu peut servir à la génération de BRDF : connaissant la géométrie de la surface, des photons sont lancés depuis une direction précise et les photons réfléchis sont collectés sur un hémisphère englobant la surface, on obtient ainsi la répartition de la réflexion de la lumière permettant de définir la BRDF de la surface. Il est toutefois nécessaire de connaître ou modéliser finement les propriétés de réflexion de la matière constituant la surface.

4.2.3 Technique de radiosité

Cette méthode, proposée en 1984 par Torrance et al [GTGB84], considère les échanges radiatifs entre des surfaces purement diffuses. Son objectif consiste à réaliser un bilan énergétique global sur toute la scène à partir de l'expression mathématique des échanges radiatifs pour des longueurs d'ondes données (figure 4.7).

La radiosité en un point x , est donnée par l'équation de base définie comme suit :

$$B(x) = E(x) + \frac{\rho_d(x)}{\pi} \int_{y \in S} B(y) G(x, y) V(x, y) dS_y$$

Avec :

- $E(x)$: l'émittance propre de la surface (exprimée en W/m^2).
- $V(x, y)$ une fonction déterminant la visibilité entre les points x et y (valant 0 ou 1).
- $G(x, y)$ une fonction géométrique exprimant la différence d'orientation des surfaces ainsi que la distance séparant les points x et y .

Tous les objets de la scène sont discréétisés en facettes, chacune d'elles ayant ses propres propriétés d'absorption, de réflexion et de transmission. L'échange radiatif entre deux facettes purement diffuses, qui traduit la proportion d'énergie quittant une facette A_i atteignant une facette A_j est définie par un facteur de forme (figure 4.6), expression purement géométrique, notée $F_{A_i \rightarrow A_j}$:

$$F_{A_i \rightarrow A_j} = \frac{1}{A_i} \int_{A_i} \int_{A_j} V(dA_i, dA_j) \frac{\cos\theta_i \cos\theta_j}{\pi r^2} dA_i dA_j$$

Avec :

- A_i : Surface de la facette i .
- $V(dA_i, dA_j)$: Fonction de visibilité entre surfaces élémentaires dA_i et dA_j .
- r_{ij} : distance entre le centre des surfaces élémentaires dA_i et dA_j .
- θ_i : Angle entre la normale à la surface dA_i et le vecteur r_{ij} .

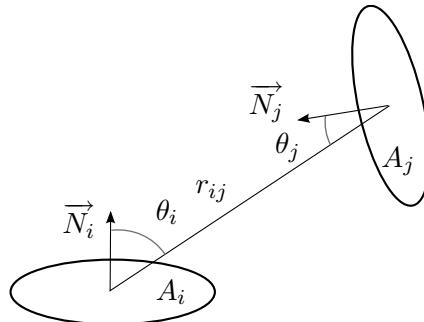


FIG. 4.6 – Facteur de forme utilisé dans la calcul de radiosité

Le calcul du facteur de forme, de part sa double intégrale et des effets d'occlusion entre facettes, n'est pas simple. Toutefois, la méthode projective de l'hémicube de Cohen [CG85] permet d'en faire une approximation.

Pour une scène subdivisée en N facettes de radiosité constante, le bilan énergétique se calcule par un système de N équations à N inconnues :

$$B_i = E_i + \rho_i \sum_{j=1}^N B_j F_{A_i \rightarrow A_j} \text{ pour } i = 1 \dots n$$

Le système peut s'écrire sous forme matricielle. La radiosité étant constante

sur chacune des facettes, la subdivision de la scène doit être fine (en centaines de milliers de facettes), donnant une matrice difficilement calculable de part sa taille. Pour palier le problème, Cohen [CCWG88] a proposé une méthode itérative de raffinement progressif. Le principe consiste à partir de la facette la plus émettrice et à propager son énergie sur les autres facettes, ainsi une seule colonne de la matrice est calculée à chaque itération. Le processus est réitéré jusqu'à convergence du système, ce qui permet également de visualiser de manière interactive la progression du résultat au fur et à mesure des itérations.



FIG. 4.7 – Rendu d'une scène avec la technique de radiosité

4.3 Le temps réel

Contrairement aux modèles d'illumination locaux présentés ci-dessus, les images de synthèse temps réel sont calculées en s'appuyant sur une architecture dédiée, appelée carte graphique, laquelle est pilotée à l'aide d'une couche logicielle telles que OpenGL (multi-plateforme) ou Direct3D (dédié au système d'exploitation Microsoft Windows). Le modèle d'éclairage implémenté nativement sur ces cartes graphiques est le modèle de Phong. Nous abordons dans ce chapitre, en plus du calcul de l'éclairage, quels sont les calculs effectués par une carte graphique (pipeline graphique) et les dernières fonctionnalités permettant d'implémenter des effets spéciaux (techniques de vertex et pixel shaders).

4.3.1 Modèle de Phong

Le modèle d'illumination de Phong décrit de manière empirique la réflexion d'une surface par combinaison de la lumière ambiante (ρ_a) , diffuse (ρ_d) et spéculaire (ρ_s). Très répandu, il est le modèle d'éclairage standard

utilisé par OpenGL qui est implémenté dans les cartes graphiques (fig. 4.8). Toutefois, ce modèle empirique ne permet pas de simuler des matériaux aux propriétés de réflexion complexes telles que l'anisotropie (ex : velours, aluminium). De plus, il ne respecte pas le principe de physique élémentaire de conservation d'énergie (le modèle de Phong crée de l'énergie).

Le modèle de Phong s'écrit :

$$I = \rho_a + \rho_d \cdot (\vec{N} \cdot \vec{L}) + \rho_s \cdot (\vec{R} \cdot \vec{E})^n$$

Le vecteur \vec{R} correspond à la réflexion du vecteur lumineux \vec{L} par rapport à la normale \vec{N} . Le coefficient n permet de définir la spécularité du matériau (plus alpha est grand, plus la réflexion spéculaire se "réduit" autour du vecteur réfléchi \vec{R}).

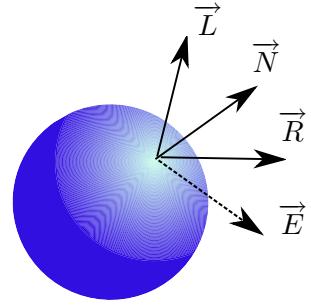


FIG. 4.8 – éclairage de Phong

4.3.2 Pipeline graphique

Le pipeline graphique est fait d'une succession d'étapes de diverses natures (géométrique, colorimétrique, interpolation, ...) qui permet de calculer en temps réel des images à partir d'instructions envoyées par l'application 3D. La figure 4.9 représente une vue simplifiée du pipeline graphique de la librairie OpenGL, supportée par la plupart des cartes graphiques accélératrices 3D.

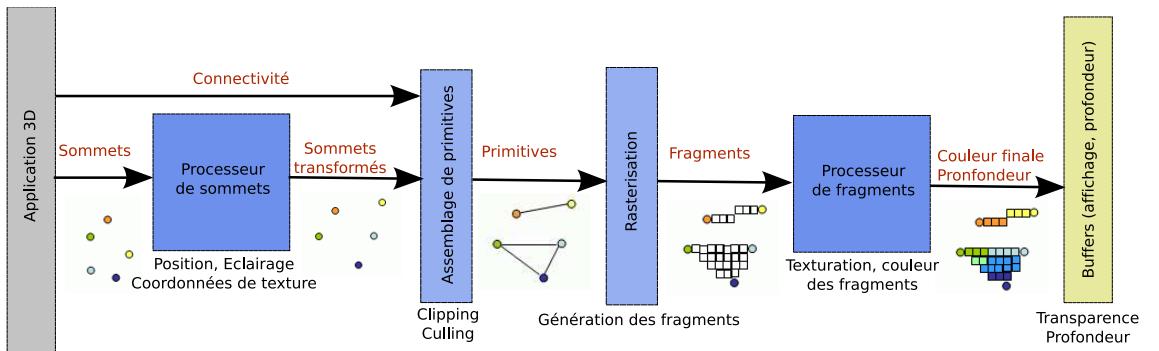


FIG. 4.9 – Vue générale du pipeline graphique OpenGL

Les principales étapes successives du pipeline graphique sont les suivantes :

- **Données d'entrée** : C'est au niveau de l'application que sont définies la géométrie de la scène (sous forme de polygones), ainsi que les sources

lumineuses. A chaque sommet d'un polygone est associé une normale, une couleur, un matériau et des coordonnées de textures pour les objets texturés.

- **Processeur de sommets** : A cette étape, les transformations géométriques sont appliqués aux sommets (application de la matrice de transformation de la caméra ("*model view*") et de la matrice de projection). Le calcul d'éclairage (modèle de Phong) est également réalisé en fonction des sources lumineuses de la scène et des propriétés des matériaux.
- **Assemblage de primitive** : Les sommets sont "réunis" selon la forme géométrique à laquelle ils appartiennent afin de créer des primitives (triangle, ligne, ...). Une fois les primitives formées, le "culling" détermine si l'on dessine ou non les faces en fonction de leurs orientations (déterminé par la normale à la surface) ainsi que le "clipping" qui permet d'éliminer les sommets se trouvant en dehors de la pyramide de vision.
- **Rasterisation** : Cette étape consiste à remplir les surfaces définies par les primitives en créant des fragments pour chaque pixel. Un fragment véhicule la profondeur, la couleur, les coordonnées de texture et la normale, dont les valeurs sont interpolées en chaque pixel.
- **Processeur de fragment** : A partir des fragments créés, cette étape consiste à effectuer le mélange entre la couleur des fragments et la texturation.
- **Tests de *frame-buffer*** : Les fragments arrivent enfin dans le *frame-buffer*, où ils subissent des tests de profondeur ("*Z-buffer*"), et de transparence en fonction de la composante alpha des fragments. Une fonction de mélange ("*blending*") indique sur le pixel du frame-buffer doit être remplacé ou mélangé avec ceux déjà présents.

4.3.3 Pipeline graphique programmable

L'architecture du pipeline graphique OpenGL présenté ci-dessus est fixe, ainsi les opérations effectuées par les processeurs de sommets et de fragments sont prédéfinies et non modifiables par l'API (*Application Programming Interface*) graphique OpenGL. Les besoins en terme d'effets spéciaux étant de plus en plus grand dans l'industrie du jeux vidéo, cette architecture fermée est devenue problématique, c'est pourquoi le pipeline graphique est devenu programmable.

Le pipeline graphique OpenGL s'est ouvert de sorte à ce que les opérations effectuées dans les processeurs de sommets et de fragments puissent être programmées par l'utilisateur [Ros05]. Ceci est possible en écrivant des programmes (appelé couramment *shader*) compilés et chargés dans la carte

graphique (*vertex program* et *fragment program*) qui vont prendre la place des instructions effectuées par les processeurs de sommets et fragments du pipeline OpenGL classique (figure 4.10).

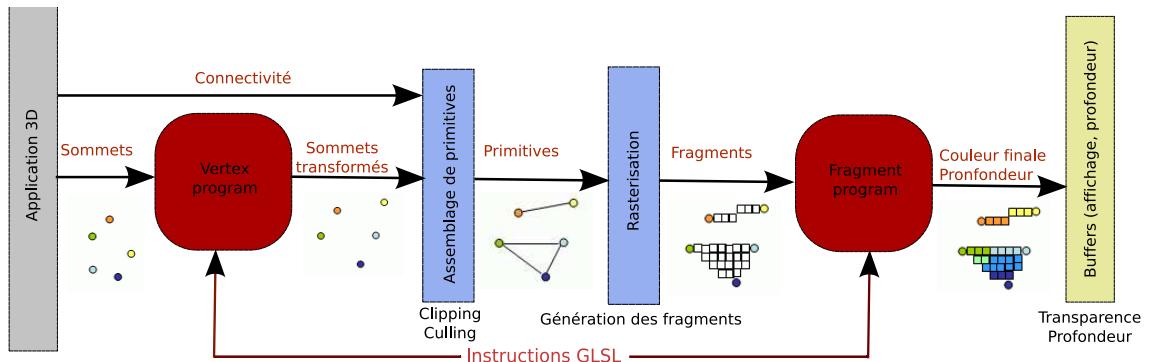


FIG. 4.10 – Pipeline graphique OpenGL avec programmes GLSL

Les programmes sont écrits à l'aide d'une API définie par un langage haut niveau ressemblant au langage C. Depuis la version OpenGL 2.0, l'API de programmation des shaders, nommée GLSL (Graphic Library Shading Language) est incluse en standard dans l'API OpenGL.

Cette API comprend un certain nombre d'instructions prédéfinies très utilisées en synthèse d'image (trigonométrie, vecteurs, matrices, texturation, etc.) et permet d'accéder directement aux états OpenGL (sources lumineuses, matériaux, etc.). Cette technique permet de réaliser un très grand nombre d'effets spéciaux tels que le bump-mapping, le "cartoon rendering", avec seulement quelques lignes de code.

Voici un exemple de code GLSL (source *Nvidia*) très simple permettant de faire du "cartoon rendering" :

Le "vertex program" calcule pour chaque sommet la position dans l'espace de la caméra et un vecteur lumineux :

```
// normale du sommet transmise au fragment program (sortie)
varying vec3 normal;

// vecteur lumineux transmit au fragment program (sortie)
varying vec3 lightDir;

void main (void)
```

```
{
    // position du sommet dans l'espace de la caméra
    vec3 ecPos = vec3 (gl_ModelViewMatrix * gl_Vertex);

    // transformation de la normale
    normal = gl_NormalMatrix * gl_Normal;

    // calcul du vecteur lumineux
    lightDir = normalize (vec3 (gl_LightSource[0].position) - ecPos);

    // calcul de la position du sommet (matrice de vue et de projection)
    gl_Position = ftransform ();
}
```

Le "*fragment program*" récupère la normale et le vecteur lumineux et attribue une couleur au fragment en fonction de l'angle entre la normale et le vecteur lumineux :

```
// normale interpolée (entrée)
varying vec3 normal;

// vecteur lumineux interpolé (entrée)
varying vec3 lightDir;

void main (void)
{
    // couleur du fragment
    vec4 color;

    // normalisation de la normale
    vec3 n = normalize (normal);

    // produit scalaire entre la normale et le vecteur lumineux
    float intensity = dot (lightDir, n);

    // sélection de la couleur en fonction du produit scalaire

    if (intensity > 0.95)
        color = vec4 (1.0, 0.5, 0.5, 1.0);
    else if (intensity > 0.5)
        color = vec4 (0.6, 0.3, 0.3, 1.0);
    else if( intensity > 0.25 )
        color = vec4 (0.4, 0.2, 0.2, 1.0);
    else
        color = vec4 (0.2, 0.1, 0.1, 1.0);

    // attribution de la couleur du fragment (sortie)
    gl_FragColor = color;
}
```

Le résultat de ce code GLSL est représenté sur la figure 4.11 où l'on peut voir un objet 3D dont le rendu ressemble à une bande-dessinée par décimation du nombre de couleur selon les plages angulaires.



FIG. 4.11 – Rendu "cartoon rendering" temps réel par shader GLSL

4.3.4 Bump-mapping

La technique de *bump-mapping*, conçue par Blinn en 1978 [Bli78], est destinée à simuler un relief sur une surface. Elle consiste à perturber la normale d'une surface en utilisant une texture (figure 4.12) contenant les perturbations que l'on applique à la normale dans son espace local TBN (Tangent, Binormal, Normal). La couleur de la texture provient du fait que la composante en z de la normale reste dominante comparativement à la norme des perturbation appliquées en x et y . Les valeurs de la texture étant comprises entre 0 et 1, il est nécessaire de leur appliquer une transformation afin d'obtenir des perturbations comprises entre -1 et $+1$.

Elle permet donc de simuler des variations géométriques en chaque pixel, évitant ainsi une définition géométrique (sommets et triangulation) fine trop coûteuse pour un calcul temps réel. Cette méthode atteint ses limites pour les angles de vue rasants étant donné qu'elle intervient uniquement pour l'éclairage de la surface et ne modifie pas réellement la géométrie de l'objet.

Plusieurs extensions de cette technique ont été proposées récemment : le parallax-mapping permet de corriger les coordonnées de textures en fonction de l'angle de vue et l'auto-ombrage [Tat06] peut maintenant être approximé en temps réel (figure 4.14). Le "*displacement mapping*" (appelé aussi "*relief*

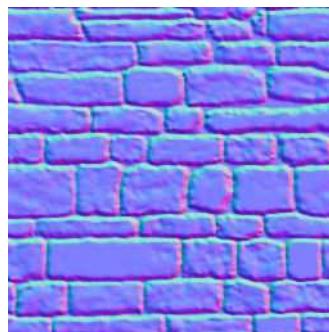


FIG. 4.12 – Texture de perturbation de normales ("normal map")

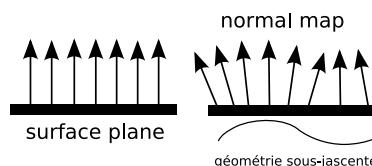


FIG. 4.13 – Technique de perturbation de normales

mapping) [POJ05] permet de déplacer les sommets en fonction de la texture de normales afin de modifier réellement la géométrie des surfaces (figure 4.15) .

Ces méthodes, fréquemment utilisés dans le monde du jeu vidéo, apportent des modifications sur les variables d'une BRDF (angles entre la normale et les directions d'éclairage et d'observation) et permettent ainsi de modifier l'aspect des surfaces planes par des variations d'éclairage.

4.4 Simulateur d'éclairage Renault

Les environnements virtuels utilisés en simulation de conduite s'appuient sur la synthèse d'images temps réel étant donné qu'il est nécessaire de mettre à jour l'affichage en fonction des actions du conducteur (position à chaque instant). Le réalisme de ces environnements dépend principalement de la modélisation des objets 3D constituant la scène. Toutefois, la simulation de certains phénomènes (éclairage, brouillard, pluie, etc.) peut être obtenue par l'implémentation de modèles physiques en temps réel en utilisant la puissance de calcul des cartes graphiques ainsi que les technologies de shaders présentées. En particulier la simulation d'éclairage, que nous allons présenter ci-après, est un outil primordial de conception des projecteurs automobile pour Renault. Il est un exemple typique de l'utilisation de la synthèse d'image



FIG. 4.14 – Rendu temps réel avec la technique de bump-mapping



FIG. 4.15 – Rendu temps réel avec la technique de "relief mapping"

temps réel pour répondre aux enjeux de l'industrie et constitue la base pour nos travaux de recherche concernant le réalisme des environnements virtuel pour la simulation de conduite.

Renault a développé un simulateur d'éclairage [LP99] temps réel (figure 4.16) permettant de simuler l'éclairage des projecteurs automobile de nuit. Il est utilisé comme outil de développement et de validation des projecteurs pour chaque projet véhicule depuis 1998, permettant de réduire les coûts, d'améliorer la qualité des projecteurs et d'effectuer des tests à toute heure de la journée tout en maîtrisant les conditions météorologiques (temps clair et brouillard).

La simulation de l'éclairage repose principalement sur deux points : la prise en compte des sources de lumière que sont les projecteurs automobile caractérisés par leur répartition photométrique ainsi que la modélisation de la réflexion des surfaces constituant la scène virtuelle par des modèles d'éclairage calculés en temps réel.



FIG. 4.16 – Simulateur d'éclairage Renault

Un phare automobile est composé de plusieurs miroirs et d'une glace striée destinés à reproduire une distribution d'énergie lumineuse donnée (figure 4.17).



FIG. 4.17 – Projecteur de Renault Scénic

L'introduction d'un modèle numérique complet d'un tel système en simulation d'éclairage est très coûteux en temps de calcul. Il faut en effet considérer toutes les réflexions lumineuses qui se produisent à l'intérieur du bloc optique afin de déterminer avec précision le flux lumineux en sortie du phare. Il n'existe pas de lois mathématiques simples pour caractériser la répartition du flux lumineux permettant d'aboutir à une simulation en temps réel de ces phénomènes. Pour cela, il est plus simple d'acquérir une cartographie (figure 4.18) décrivant la répartition d'énergie lumineuse du phare en sortie du bloc optique. Les photométries des projecteurs sont mesurées à l'aide d'un gonio-spectrophotomètre ou simulées à l'aide de logiciels non temps réel (SPEOS).



FIG. 4.18 – Distribution d'éclairage d'un projecteur (feu de croisement) en fausses couleurs.

L'algorithme de rendu utilise ensuite la technique des textures projetées afin de simuler l'éclairage provenant du projecteur sur la scène virtuelle. La texture comprenant la répartition photométrique du projecteur joue alors un rôle de filtre (figure 4.19) pour la source de lumière ponctuelle. Le modèle de Phong est alors utilisé [LP99] pour modéliser la réflexion des surfaces, la luminance est ainsi calculée en chaque sommet de la scène par :

$$L = \rho \frac{I_{max} \cos \theta}{\pi r^2} F(\vec{\omega}_i)$$

Où ρ est le coefficient de réflexion diffus de la surface, I_{max} est l'intensité maximale du projecteur, r la distance entre le sommet et la source de lumière, θ l'angle entre la normale à la surface et le rayon lumineux $\vec{\omega}_i$ et $F(\vec{\omega}_i)$ la distribution lumineuse normalisée vérifiant :

$$0 \leq F(\vec{\omega}_i) \leq 1$$

Ce calcul s'effectue en plusieurs passes si il y a plusieurs projecteurs, le résultat de chaque passe étant additionné dans le "*frame-buffer*" de la carte graphique. Le calcul s'effectuant en chaque sommet (puis interpolé linéairement), il est nécessaire de sur-discrétiser la géométrie de la scène, c'est-à-dire que la scène est définie par un nombre de sommets supérieur à celui nécessaire à la représentation de sa géométrie globale.

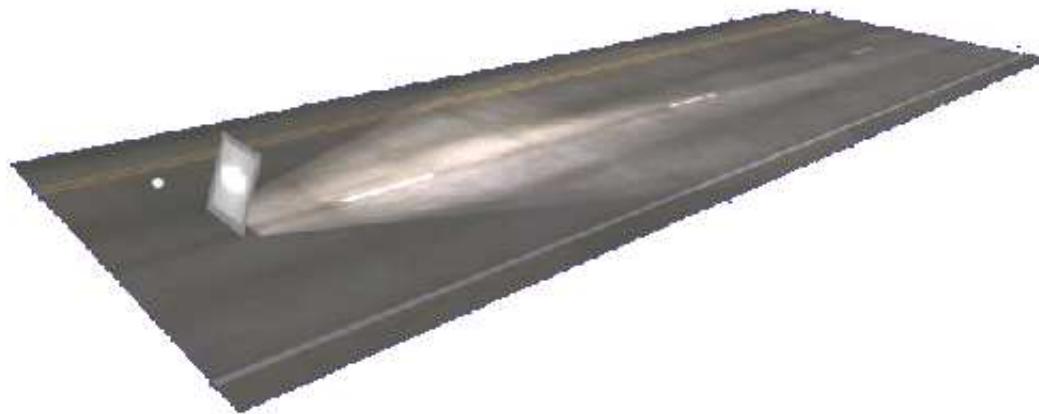


FIG. 4.19 – Principe de la texture projetée (filtrage de la source ponctuelle par la photométrie du projecteur).

Une campagne de validation a été menée. Elle a consisté dans un premier temps à l'évaluation visuelle de la restitution des caractéristiques du faisceau (uniformité, largeur et portée). La deuxième étape a consisté à mesurer (par un spectrophotomètre) et comparer l'éclairage d'une route par un même projecteur en situation réelle et sur simulateur de conduite. Les résultats [dLR99] fournissent un taux d'erreur de l'ordre de 3% à 4% et montrent que la simulation d'éclairage est un outil dont la validité permet une utilisation dans le cycle de conception des projecteurs automobile pour Renault.

Chapitre 5

Les Fonctions de Textures Bidirectionnelles (BTf)

La peinture des véhicules, le revêtement routier, les panneaux de signalisation, la végétation, sont autant de matériaux aux propriétés de réflexion très différentes. L'aspect de ces matériaux joue un rôle important dans le réalisme visuel et leur modélisation est aujourd'hui un challenge majeur dans le monde de l'image de synthèse. Habituellement, la géométrie d'une surface est modélisée explicitement (triangles) jusqu'à une certaine échelle tandis que la micro-structure responsable du comportement en terme de réflectance est simulée en utilisant des modèles d'éclairage analytiques (chapitre 4.1) relativement simples. Cette méthode permet de simuler, de manière plus ou moins réaliste, un certain nombre de matériaux appartenant à des groupes particuliers comme le métal et le plastique. Obtenir l'effet désiré, n'est pas toujours facile car les paramètres de ces modèles ne sont pas toujours intuitifs ou n'ont pas de véritable sens physique.

Comme nous l'avons vu précédemment, une BRDF donne une description du comportement d'un matériau à la lumière, elle intègre les phénomènes lumineux qui interviennent à petite échelle sur un matériau (inter-reflections, auto-ombrages, diffraction, etc.). Cette fonction, qui correspond à un bilan énergétique, ne permet pas de restituer les phénomènes locaux que l'on désire parfois visualiser. Par exemple, la peinture d'un véhicule est formée de plusieurs couches contenant des paillettes métalliques qui ont un impact important sur son aspect visuel. La simulation de peinture basée uniquement sur l'utilisation d'une BRDF ne permet pas de restituer la brillance non homogène provenant des paillettes [DBP01] mais seulement sa réflexion globale. Ceci est également vrai pour des matériaux d'aspect granuleux, tel que l'asphalt, qui ne peut être représenté par sa BRDF que pour des distances d'observation suffisamment éloignées afin que les variations d'éclairage soient intégrées dans un pixel. Il est donc nécessaire de représenter la mésostructure d'un tel matériau pour en avoir une perception réaliste pour des distances d'observation proches.

Simuler la mésostructure d'un matériau hétérogène devient beaucoup plus complexe car elle se situe entre la géométrie à grande échelle (modélisée explicitement par des triangles) et la micro-structure modélisée par une BRDF. Elle joue un rôle très important dans l'aspect visuel des matériaux, et sans une modélisation fine, les images générées paraissent artificielles.

Les méthodes de type "*image-based rendering*" permettent d'intégrer des images du monde réel dans les images de synthèse. Par exemple, l'utilisation de texture 2D ajoute des détails colorimétriques sur les surfaces à partir d'une photo du matériau correspondant dans le monde réel. Toutefois, les textures ne sont capables que de représenter une partie restreinte de l'apparence d'un matériau : ces derniers sont considérés comme plats et diffus et les phénomènes dépendants des conditions d'éclairage et d'observation tels que l'auto-ombrage, l'auto-masquage, et les réflexions complexes non diffuses, pourtant importants dans l'aspect visuel du matériau, ne sont pas restitués.

La notion de BTF (Fonction de Texture Bidirectionnelle), introduite par Dana et al. [DvGNK99], est une extension de la notion de texture 2D classique. Elle contient tous ces phénomènes en définissant un ensemble de textures 2D, chacune d'entre elles, correspondant à une condition d'éclairage et de vue particulière. Une telle fonction contient une quantité de données très importante (plusieurs gigaoctets) et nous allons voir dans ce chapitre, les différentes problématiques liés au domaine des BTF. En particulier, nous présentons les méthodes d'acquisition, de compression et les techniques de rendu temps réel pour la simulation d'éclairage réaliste de matériaux hétérogènes.

5.1 Modèles de réflectance

Pour décrire les propriétés de réflectance d'un matériau, il est nécessaire de définir un certains nombre de paramètres afin de pouvoir décrire le trajet de la lumière sur la surface. La lumière, de longueur d'onde λ_i , arrive sur une surface au temps t_i , à la position x_i avec une direction d'incidence caractérisée par les angles (θ_i, ϕ_i) . Elle parcourt un certain trajet sur la surface (ou par exemple à travers des couches minces le cas échéant) et la quitte au temps t_o en un point x_o dans une direction (θ_o, ϕ_o) , et une longueur d'onde potentiellement modifiée λ_o . On obtient ainsi une fonction à 12 dimensions (figure 5.1) pour décrire la réflectance de la surface, qui est très difficile à mesurer et à exploiter.

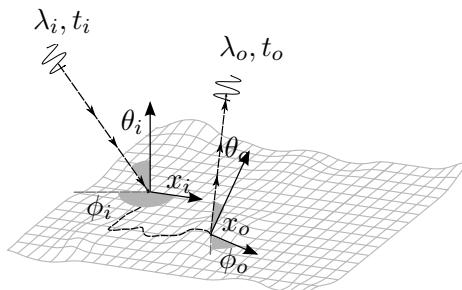


FIG. 5.1 – Paramètres nécessaires à la description de l'interaction lumière-matière

En considérant que la longueur d'onde ne varie pas et en négligeant le temps du trajet de la lumière sur la surface, le modèle de réflectance est défini par une fonction à huit dimensions : BSSRDF (Bidirectional Surface Scattering Reflection Distribution Function) introduite par Nicodemus et al. [NF77]. La figure 5.2 montre comment il est possible d'obtenir des modèles plus simples (de dimensions inférieures) en fixant certaines variables. Nous allons détailler brièvement les différents modèles existants afin de spécifier leur champ d'application et comprendre l'intérêt des fonctions de textures bidirectionnelles.

La dispersion sous-surface d'un matériau homogène, comme du lait, peut être représentée par une BSSDF. Les matériaux opaques et uniformes n'ont pas besoin de dimensions spatiales, et peuvent ainsi être représentés par une BRDF à quatre dimensions. Si l'on fixe la position de l'observateur, les "surface light-field" permettent de représenter une surface sous plusieurs conditions d'éclairage. Si l'on fixe la position de la lumière, les "surface reflectance-field" permettent de représenter une surface sous plusieurs conditions de vue.

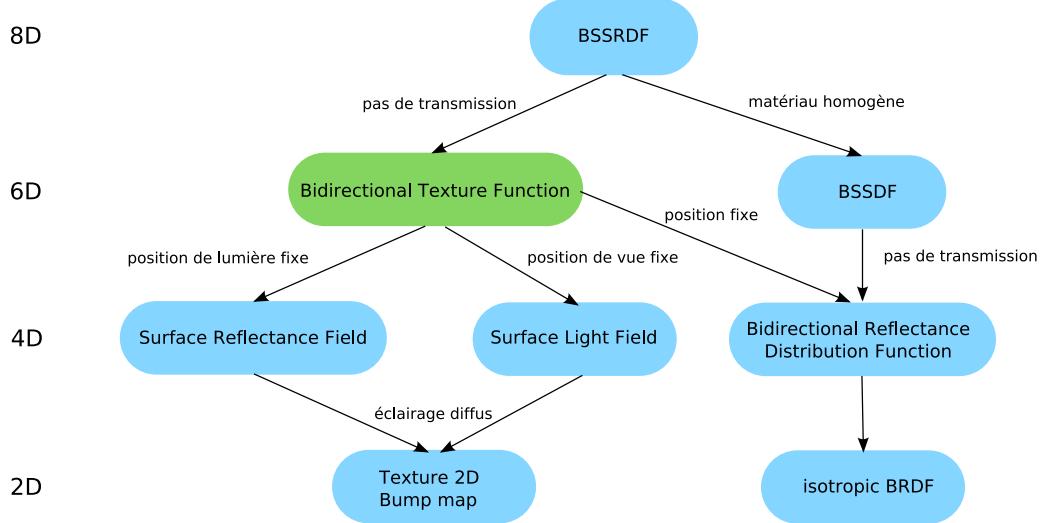


FIG. 5.2 – Hiérarchie des modèles de réflectance

5.2 BTF

5.2.1 Définition

Une BTF est une représentation en six dimensions de la fonction générale de réflectance. En chaque point est réalisée une intégration des interactions lumineuses avec les autres points de la surface :

$$BT{F_{rgb}}(x, \theta_i, \varphi_i, \theta_o, \varphi_o) = \int_S BSSRDF_{rgb}(x_i, x, \theta_i, \varphi_i, \theta_o, \varphi_o) dx_i$$

La notion de BTF (*Bidirectional Texture Function*) est équivalente à celle de la BRDF avec une dimension spatiale. C'est un ensemble discret de textures représentant un élément surfacique d'un matériau :

$$\{T_{(v,l)}\}_{(v,l) \in M}$$

Chaque texture de cet ensemble discret correspond à une condition d'éclairage et de vue donnée (comme pour l'albédo dans le cas de la BRDF). M représente l'ensemble des conditions de vue et d'éclairage (notés (v, l) à la place de (ω_o, ω_i) afin d'alléger les notations).

Une BTF peut également être interprétée comme un ensemble de BRDF en chaque point x d'une texture de résolution N :

$$\{B_x\}_{x \in N^2}$$

Toutefois, chaque BRDF B_x ne respecte pas les principes fondamentaux de conservation d'énergie et de réciprocité, il est alors préférable de parler de ABRDF (*Apparent BRDF*). En effet, un point donné de la texture peut être "affecté" par ses voisins : en fonction des conditions d'éclairage et d'observation, il peut se trouver à l'ombre ou être masqué par un de ses voisins, ce qui peut potentiellement introduire de fortes discontinuités dans la fonction décrivant l'éclairage en ce point.

L'intérêt d'une BTF se trouve dans le fait qu'elle contient des phénomènes qu'il serait très difficile à simuler en temps réel, tels que l'auto-ombrage, l'auto-masquage et les inter-reflections, normalement obtenus avec des algorithmes de synthèse d'images complexes non temps réel comme le lancer de rayon (chapitre 4.2). En contrepartie, la quantité de données d'une BTF est très importante (plusieurs gigaoctets pour un matériau en fonction du pas d'échantillonnage pour les différentes variables). Une telle fonction, de par sa dimension et sa taille, est donc difficile à exploiter directement en temps réel pour la simulation d'éclairage. La recherche autour des BTF, en dehors de son acquisition, a pour enjeu la compression des données afin d'en effectuer le rendu (décompression) en temps réel.

Après avoir présenté un système d'acquisition, nous exposons les différentes techniques de compression et de rendu de BTF que l'on trouve dans la littérature. Les techniques temps réel sont en général fondées sur la recherche d'un modèle analytique représentant la BTF ou encore sur des outils statistiques tels que l'analyse en composantes principales (ACP) ou encore la décomposition en valeurs singulières (DVS).

5.2.2 Acquisition de BTF

Contrairement à une texture 2D classique, l'acquisition de BTF requiert un matériel coûteux et un processus complexe. En effet, la mesure physique de la réflexion de matériaux réels nécessite une précision et une calibration parfaite afin que la BTF mesurée ne contiennent pas d'artefacts. Les systèmes de mesure sont en général fondés sur des capteurs CCD, alors qu'une BRDF est mesurée à l'aide d'un spectro-photomètre. Le premier système de mesure réalisé par Dana et al. [DvGNK99] en 1999, prenait 205 images de matériaux isotropiques. Une base de données de 61 matériaux est disponible sur internet (base de données Curet). L'échantillonnage de cette mesure est trop important pour l'exploiter en rendu de surface, mais son application première était la vision par ordinateur comme la reconnaissance de surface ou encore l'extraction de la géométrie d'une surface à partir de photographies.

Plus récemment, l'Université de Bonn a réalisé un appareil pour la mesure de BTF [MMS⁺] et propose également quelques échantillons disponibles sur son site internet.

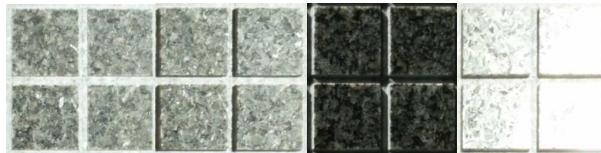


FIG. 5.3 – Échantillons de BTF mesurée.

Ce système est composé d'un ensemble de caméras CCD fixes (figure 5.4) et mesure des échantillons jusqu'à 10 centimètres de largeur pour 81 points de vues et 81 positions d'éclairage (6561 images). Encodée en RVB (sur 8 bits) avec une résolution de 256x256, la BTF finale mesurée occupe une place mémoire de 1,2 Gigaoctets.

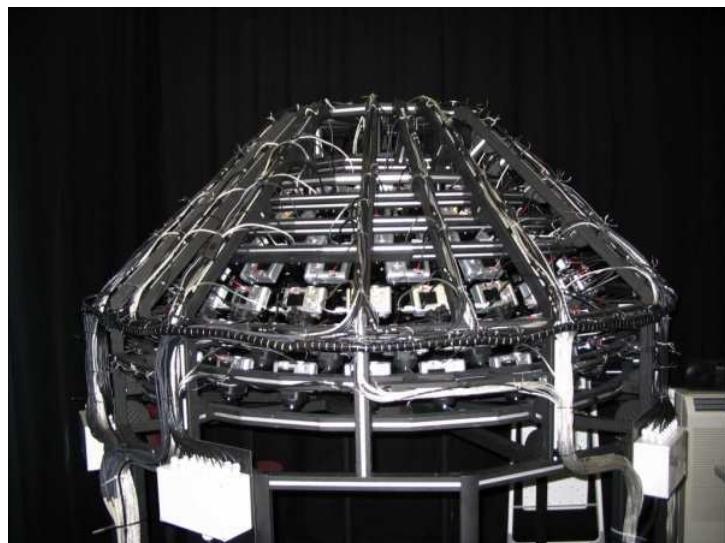


FIG. 5.4 – Système de mesure de BTF (Université de Bonn)

Pour garantir la validité des résultats, un certains nombre de réglages (calibration) et de post-traitements des données sont effectués :

- Afin de corriger les erreurs de position (problèmes de précision) du système de mesure, des marqueurs sont placés sur le bord de la surface, permettant d'effectuer un post-traitement sur l'image qui corrige les

éventuels décalages.

- Une calibration géométrique qui permet de réduire la distortion en bâillet introduite par le capteur CCD est appliquée, ce qui nécessite de connaître les caractéristiques du capteur CDD.
- La sensibilité du capteur est réglée afin d'avoir un temps d'ouverture constant pour qu'il n'y ait pas de saturation ou d'images trop sombre pour l'ensemble des images constituants la BTF.
- Afin de reproduire les couleurs avec la meilleure exactitude possible, une calibration colorimétrique du système de mesure et de la source de lumière est réalisée.
- Après avoir mesuré l'ensemble des images, il est nécessaire d'effectuer une transformation homographique afin de replacer les images dans le plan ($\theta = 0, \phi = 0$) de manière à pouvoir les utiliser comme des textures classiques.

5.3 Méthodes de compression

Étant donné la dimension et l'espace mémoire occupé par une BTF, en faire une exploitation directe pour le rendu de surface est difficile. Ainsi, pour arriver à simuler l'aspect des matériaux en temps réel à partir d'une telle fonction, il est nécessaire d'appliquer une sorte de compression sur les données de la BTF. Les méthodes de compression doivent réduire considérablement la quantité de données représentant la BTF, préserver autant que possible les caractéristiques du matériau, exploiter la redondance des données constituant la BTF, tout en gardant son objectif principal de décompression temps réel. En résumé, les trois enjeux parallèles à optimiser sont :

- Réduction maximale de la taille de la BTF,
- Minimisation du taux d'erreur du à la compression,
- Reconstruction temps réel de la BTF.

La méthode de compression d'image la plus répandue est la compression JPEG (Joint Photographic Experts Group). Il est donc naturel d'étudier son fonctionnement afin de déterminer si celui-ci peut répondre aux contraintes citées ci-dessus pour la compression de BTF. Son calcul nécessite plusieurs d'étapes (figure 5.5), à savoir :

- Changement d'espace de couleurs RVB vers YCrCb.
- Sous échantillonnage de l'image par bloc (calcul de la chrominance moyenne par bloc de 16x16)
- Calcul de la DCT (Discret Cosinus Transformation) par bloc de 8x8 pour la luminance.

- Quantification des données afin d'éliminer les hautes fréquences de l'image.
- Linéarisation des données afin de regrouper les données nulles.
- Codage RLE (Run Length Encoding) des données linéarisées.
- Codage de Huffman afin d'obtenir une nombre réduit de données servant à décrire les répétitions présentes.

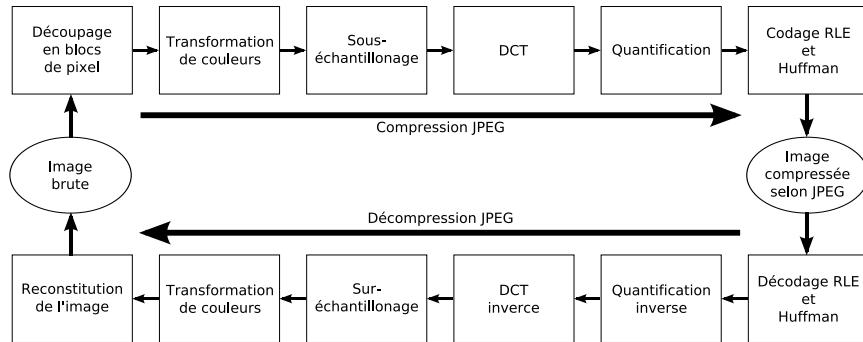


FIG. 5.5 – Méthode de compression JPEG

Cet algorithme, qui comprend donc un grand nombre d'étapes, permet d'obtenir des taux de compression élevés pour lesquels la qualité de l'image reste satisfaisante (figure 5.6). Toutefois, pour les raisons suivantes, il ne répond pas aux contraintes de la compression de BTF :

- La méthode JPEG ne permet pas de réduire la dimension (6D) de la BTF. Il sera alors difficile d'exploiter ces données lors du rendu temps réel étant donné que les cartes graphiques ne supportent au maximum que des fonctions à trois dimensions (texture 3D).
- La décompression nécessite tout autant de calculs que la compression, ce qui pose problème pour une simulation en temps réel, étant donné que le nombre d'étapes de la méthode JPEG est très important .

Nous allons maintenant faire une exploration des différentes méthodes de compression (analytiques, statistiques) de la littérature employées pour la compression de BTF respectant les contraintes citées ci-dessus.

5.3.1 Modèles analytiques

Comme nous l'avons vu, une BTF peut être interprétée comme un ensemble de BRDF. L'approche la plus naturelle pour la compression de BTF consiste donc à la représenter par des modèles de BRDF analytiques facilement exploitables en temps réel, comme le très utilisé modèle de Phong



FIG. 5.6 – Comparaison d'une image (Lena) au format bmp (130 Ko, à gauche) et jpeg (40 Ko, à droite)

[Pho75], de Ward [War92] ou encore les lobes en cosinus de Lafourture [LFTG97] que nous allons détailler.

Lobes de Lafourture

La méthode la plus simple de compression de BTF par un modèle analytique fut proposée par McAllister et al. [MLH02] et se base sur les lobes de Lafourture [LFTG97]. Lafourture fait une approximation d'une BRDF par la somme de lobes en forme de cosinus. Un lobe de Lafourture j est défini comme suit :

$$s_{x,j}(v, l) = \left(\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}^T \cdot \begin{bmatrix} C_{j,x} & & \\ & C_{j,y} & \\ & & C_{j,z} \end{bmatrix} \cdot \begin{bmatrix} l_x \\ l_y \\ l_z \end{bmatrix} \right)^{n_j}$$

Où v et l sont respectivement les directions de vue et d'éclairage, et où la matrice C_j et l'exposant n_j symbolisent les paramètres du lobe spéculaire j . Pour trouver les paramètres des lobes, une méthode d'optimisation non-linéaire, comme la très répandue méthode de Levenberg-Marquart [PFTV88], est utilisée. Toutefois, le coût d'évaluation des lobes est tel, qu'il est difficile d'utiliser plus de 4 lobes pour représenter la BTF, ce qui limite cette méthode à un ensemble très limité de matériaux ayant une géométrie simple.

La BTF est approximée en utilisant k lobes par l'expression suivante :

$$BTF(x, l, v) = \rho_{d,x} + \sum_{j=1}^k \rho_{s,x,j} \cdot s_{x,j}(v, l)$$

Où ρ_d et ρ_s sont respectivement le coefficient de réflexion diffus et spéculaire. Ce modèle permet d'aboutir à une représentation très compacte car

elle n'utilise que peu de paramètres (environ 2 mégaoctets en fonction du nombre de lobes et de la résolution spatiale).

Les ABRDF peuvent contenir des effets discontinus d'ombrage et de masquage qu'il est impossible de modéliser avec des lobes en forme de cosinus. Daubert et al. [DLHS01] ont alors amélioré le modèle en y ajoutant un terme multiplicatif $T_x(v)$ pour tenir compte des effets d'occlusion :

$$BTF(x, l, v) \approx T_x(v) \cdot \left(\rho_{d,x} + \sum_{j=1}^k \rho_{s,x,j} \cdot s_{x,j}(v, l) \right)$$

Ce terme étant stocké pour chaque pixel, le nombre de paramètres devient beaucoup plus important, accroissant ainsi le coût d'évaluation de la BTF.

Reflectance field

Dans le but d'améliorer le résultat des précédents modèles en terme de qualité, Meseth et al. [MMK03a] ont proposé de modéliser des champs de réflectance (*Reflectance field* en anglais) avec des lobes de Lafortune, et d'effectuer une interpolation linéaire entre chaque condition d'éclairage. L'évaluation de la BTF devient alors :

$$BTF(x, l, v) \approx \sum_{v \in N(v)} w_{x,v} \cdot RF_{x,v}(l)$$

Où $N(v)$ est l'ensemble des conditions de vue les plus proches de la direction v , $w_{x,v}$ est le coefficient servant à l'interpolation linéaire et $RF_{x,v}(l)$ est le champ de réflectance pour la direction de vue v approximé par des lobes de Lafortune :

$$RF_{x,v}(l) \approx \rho_{d,x} + \rho_{s,x,v}(l) \cdot \sum_{j=1}^k s_{x,v}(l)$$

Étant donné que les champs de réflectance sont évalués pour chaque pixel pour une condition de vue particulière, le nombre de paramètres est très important. Cette méthode n'est donc utilisée que pour évaluer les variations en terme de luminance de la BTF et non pas pour chaque composante RVB comme pour la méthode précédente.

Plus récemment, Filip et Haindl [FH04] ont proposé une méthode semblable mais en se basant sur des polynômes dépendants à la fois des condi-

tions de vue et d'éclairage :

$$RF_{x,v}(l) \approx \sum_{I \in N(l)} w_l \sum_{i=1}^k a_{i,x,v,l} (\rho_{s,v} \cdot s_{x,v}(l))^{i-1}$$

Où a est le coefficient du polynôme et $N(l)$ est l'ensemble des directions les plus proches parmi celles correspondantes aux données mesurées. Bien que la qualité de la BTF soit mieux préservée, son évaluation reste très coûteuse au regard du nombre de paramètres, limitant ainsi son application pour la simulation d'éclairage en temps réel.

5.3.2 Décomposition en base linéaire par approche matricielle

L'utilisation de modèles analytiques n'est souvent adaptée qu'à une certaine catégorie de matériaux étant donné que les effets discontinus contenus dans les ABRDF sont très difficiles à restituer avec cette méthode. Afin de palier ce problème et de ne plus être contraint par les hypothèses précédentes (approximation d'une ABRDF par des lobes de Lafortune), la BTF peut être interprétée comme un signal multidimensionnel. Les techniques générales de traitement du signal peuvent alors être utilisées, comme l'analyse en composantes principales (ACP) que l'on peut assimiler à une décomposition en valeurs singulières (SVD). Nous allons maintenant présenter le principe de cette méthode ainsi que sa mise en œuvre par différentes techniques pour la compression de BTF.

Décomposition en Valeurs Singulières (SVD)

Bien que découverte en 1873 par Eugenio Beltrami, la décomposition en valeurs singulières (SVD) a été prouvée mathématiquement en 1936 par [EY36]. L'objectif de la SVD est de trouver une combinaison linéaire de deux ensembles de variables tels que cette combinaison aie la plus grande covariance possible. Cette maximisation se fait sous la contrainte d'orthogonalité des coefficients de la combinaison linéaire, que nous allons expliciter ci-après.

Définition mathématique

Soit X et Y deux matrices, respectivement de taille n, p et n, q . Soit C_{XY} la matrice de covariance de taille p, q définie par :

$$C_{XY} = \left(\frac{1}{n}\right) X^T Y$$

L'objectif de la décomposition en valeurs singulières est de trouver une combinaison linéaire de Xa_i et Yb_i ayant la covariance maximale ($i =$

$1..r, r = \min(p, q)$). Les vecteurs a_i et b_i , respectivement de taille $p, 1$ et $q, 1$, doivent respecter la contrainte d'orthogonalité suivante :

$$a_i^T a_j = b_i^T b_j = \begin{cases} 0 & \text{si } i = j \\ 1 & \text{si } i \neq j \end{cases}$$

La décomposition en valeurs singulières de la matrice C est de la forme :

$$C = U.D.V^T$$

Où :

- U est une matrice orthogonale $U^T.U = I$ de taille p, r
- D est une matrice diagonale de taille r, r
- V est une matrice orthogonale de taille q, r

Les colonnes de la matrices U contiennent les vecteurs singuliers gauches. Les lignes de la matrice V^T contiennent les vecteurs singuliers droits. Les éléments de la matrice D sont différents de zéro sur sa diagonale et correspondent aux valeurs singulières qui, par convention, sont classées par ordre décroissant. Elles correspondent à la variance des données projetées dans le nouvel espace formé par les vecteurs singuliers.

Cette décomposition permet de trouver la matrice la plus proche de C de rang inférieur l de la manières suivante :

$$C^{(l)} = \sum_{k=1}^l U_k.D_k.V_k^T$$

Le terme "plus proche" se traduit par la minimisation de la somme des différences au carré des éléments de C et $C^{(l)}$, soit :

$$\sum_{ij} |c_{ij} - c_{ij}^{(l)}|^2$$

Méthode de calcul

Une solution (il en existe plusieurs) pour calculer la décomposition en valeurs singulières est de calculer V^T et D en diagonalisant $C^T C$:

$$C^T C = V.S^2.V^T$$

et ensuite de calculer U par :

$$U = C.V.D^{-1}$$

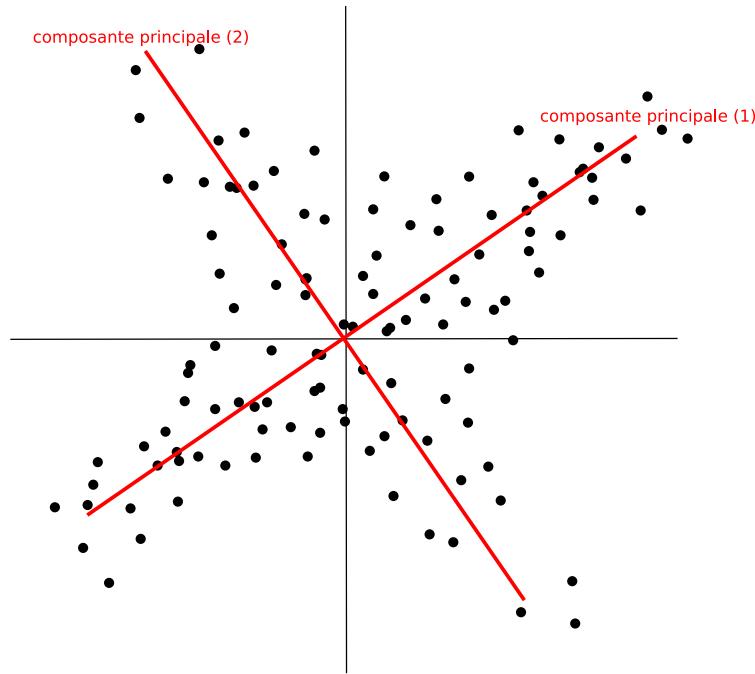


FIG. 5.7 – Principe de l'analyse en composantes principales (exemple en 2D)

Analyse en composantes principales (ACP)

L'analyse en composantes principales est une méthode classique utilisée pour la réduction de dimension d'un ensemble de données : elle permet d'extraire de données d'un nuage de point multidimensionnel les directions importantes (figure 5.7).

Bretherton et al. ont démontré en 1992 [BSW92] que lorsque les matrices X et Y sont égales et que les colonnes de cette matrice sont centrées (moyenne de chaque colonne égale à 0), alors la SVD est équivalente à l'analyse en composantes principales.

Factorisation de matrice par texel

L'approche envisagée par Kautz et McCool [KM99] consiste à appliquer une analyse en composantes principales sur les ABRDF $B_x(v, l)$ pour chaque point (texel) x . Le but est de trouver une factorisation d'une fonction à 4 dimensions en un ensemble de fonctions à 2 dimensions :

$$B_x(v, l) \approx \sum_j^c g_{x,j}(\pi_1(v, l)) h_{x,j}(\pi_2(v, l))$$

Les fonctions π_1 et π_2 sont les projections des paramètres de l'ABRDF

sur un espace à 2 dimensions. La paramétrisation de l'ABRDF doit être choisie avec attention car elle impacte fortement les résultats obtenus lors de la factorisation sur des espaces de dimensions réduites. La reconstruction en temps réel, basée sur l'utilisation des fonctionnalités d'une carte graphique, est relativement simple car les fonctions $g_{x,j}$ et $h_{x,j}$ peuvent être stockées dans des textures 2D classiques et sont combinées lors du rendu. Le compromis entre la qualité et la rapidité du rendu peut s'ajuster en réglant le nombre de termes c .

Plusieurs méthodes basées sur ce type de factorisation ont été proposées. La solution fournissant le taux d'erreur RMS le plus faible consiste à stocker les ABRDF dans une matrice X et à appliquer l'analyse en composantes principales sur cette matrice. Toutefois, les fonctions projetées contiennent des valeurs négatives, ce qui est problématique pour l'implémentation avec la carte graphique qui ne supporte que des valeurs positives et il n'est pas possible de contrôler l'algorithme par le taux d'erreur car celui-ci ne correspond pas à l'erreur visuelle résiduelle. McCool et al. [MAA01] ont proposé une méthode alternative appelée factorisation homomorphique (HF) : plutôt que d'utiliser une somme de produits de fonction projetées, une approximation de la BRDF est réalisée par le produit de fonctions de bases strictement positives :

$$B_x(v, l) \approx \prod_j^c \rho_{x,j}(\pi_j(v, l))$$

La solution est obtenue en minimisant l'erreur RMS, correspondant directement à l'erreur perceptible, et le fait que les fonctions π_j soient arbitraires laisse la possibilité de représenter un ensemble très large de BRDF, quelque soient leurs spécificités. Cet aspect est primordial étant donné que la matrice de départ peut contenir des fortes variations horizontales et verticales, dues à l'auto-masquage et à l'auto-ombrage, ainsi que des piques spéculaires sur sa diagonale. En fonction de la paramétrisation de la matrice, un terme unique peut représenter un de ces effets de manière isolée.

Plus récemment, Suykens et al. [SVLD03] a proposé une méthode appelée "*factorisation de matrices chaumées*" (CMF), qui réunit les deux principes de factorisation précédents :

$$B_x(v, l) \approx \prod_j^d \sum_k^{c_j} p_{x,j,k}(\pi_{j,1}(v, l)).Q_{x,j,k}(\pi_{j,2}(v, l))$$

Cette méthode, appliquée sur des BTF synthétiques, consiste à effectuer une suite de factorisations élémentaires, basées sur la décomposition en valeurs singulières, en changeant à chaque étape la paramétrisation des ABRDF. L'étape finale consiste à calculer des clusters [Mac67] avec un al-

gorithme de type *k-mean clustering* [LT05] afin de regrouper les fonctions projetées ayant des similitudes. Le nombre de clusters obtenus est constant, indépendamment de la complexité du matériau.

Le taux de compression des techniques de factorisation de matrice par texel dépend de la taille de matrice X , c'est à dire de l'échantillonnage angulaire de la BTF. Rappelons que ces techniques ont été dédiées originellement au rendu temps réel de BRDF, hors une BTF de résolution 256² contient 64000 ABRDF, c'est pourquoi il est nécessaire d'y ajouter une étape de clusterisation qui affecte fortement le résultat en terme de qualité.

Factorisation de matrice globale

Les méthodes de factorisation par texel décrites ci-dessus présentent l'inconvénient de ne pas exploiter la cohérence inter-texel, hors, il est fortement probable que certaines ABRDF soient très similaires sur l'ensemble des texels surtout lorsque la BTF représente un matériau donné. Lui et al. [TZL⁺02] ont alors proposé d'appliquer une analyse en composantes principales sur une matrice globale X contenant l'ensemble des ABRDF :

$$X = (B_{x_0}, B_{x_1}, \dots, B_{x_{|I|}})$$

L'approximation de la BTF en ne gardant que c valeurs propres est alors exprimée par la relation suivante :

$$BTF(x, v, l) \approx \sum_j^c g_j(x) h_j(v, l)$$

Le problème principal est de choisir le nombre de composantes c pour représenter un matériau. Ramamoorthi [Ram02] a démontré de manière analytique qu'un matériau de réflexion lambertienne pouvait être approximé de manière satisfaisante avec seulement 5 composantes principales. Toutefois, pour des BTF plus complexes, il est nécessaire de d'utiliser plus de composantes pour reconstruire la BTF avec une bonne précision. Ainsi, Koudelka [MLK03] utilisa près de 150 composantes principales pour représenter les effets d'auto-masquage, d'auto-ombrage et des réflexions non-diffuses de manière satisfaisante. Afin de réduire encore la taille mémoire nécessaire, les fonctions projetées sont compressées dans des images JPEG, rendant impossible une reconstruction en temps réel avec tant de données (voir chapitre 5.3).

La contrainte principale de ces méthodes réside dans la place mémoire que peut occuper la matrice X qui peut atteindre plusieurs gigaoctets en

représentation flottante et le temps de calcul très important de la matrice de covariance $X \cdot X^T$.

Factorisation de matrice par condition de vue

Pour palier les problèmes liés à la factorisation de la matrice globale que nous avons évoqués, Sattler et al. [MMS⁺] ont proposé une méthode dont le principe est d'appliquer une analyse en composantes principales sur des sous-ensembles de la BTF pour lesquels la position de l'observateur est fixe :

$$X_{v_i} := \left(T_{(v_i, l_0)}, T_{(v_i, l_1)}, \dots, T_{(v_i, l_{M_{v_i}})} \right)$$

Où M_{v_i} est l'ensemble des textures pour les conditions d'éclairage capturées pour une direction d'observation fixe v_i . L'objectif initial est la visualisation de tissus avec des variations de profondeurs très marquantes produisant des effets fortement non-linéaires selon le changement de position de l'observateur. L'analyse en composantes principales est appliquée de manière indépendante sur toutes les matrices X_v , l'approximation de la BTF est alors exprimée par la relation :

$$BTF(x, v, l) \approx \sum_{j=1}^c g_{v,j}(l) h_{v,j}(x)$$

Cette méthode permet de reconstruire la BTF avec une bonne exactitude avec peu de composantes (usuellement entre 4 et 16), ce qui permet une implémentation temps réel possible bien que les besoins en terme de mémoire restent importants.

Factorisation par cluster

Comme nous l'avons vu, les données d'une BTF prises dans leur ensemble présentent des effets fortement non-linéaires, ce qui limite l'efficacité de l'analyse en composantes principales. Toutefois, des sous-ensembles locaux de la BTF ont un comportement linéaire, c'est pourquoi les méthodes de factorisation présentées se concentrent parfois sur des sous-ensembles particuliers (factorisation par texel, par condition de direction d'observation). Une généralisation de cette approche a été introduite par Kambhatla et Leen [KL97], qui propose de choisir les sous-ensembles en fonction des données et non pas uniquement en fonction des variables d'entrée. Plus récemment, Mueller et al. [MMK03b] ont repris cette idée pour la compression de BTF en combinant l'analyse en composantes principales et la technique de clusters (regroupement de données similaires), ces derniers étant déterminés en fonction de l'erreur obtenue à la reconstruction. L'approximation de la BTF est donnée par l'expression suivante :

$$BTF(x, v, l) \approx \sum_j^c g_{k(x),j}(x) h_{k(x),j}(v, l)$$

Où $k(x)$ est l'index du cluster correspondant à la position x . Cette méthode est appliquée dans l'espace des ABRDF qui est plus approprié au rendu temps réel que dans l'espace image et le nombre de clusters est déterminé pour trouver un compromis entre le nombre de calcul et la qualité du résultat.

5.4 Discussion

Nous avons exposé les différentes méthodes de la littérature visant à représenter de manière compacte (en terme de taille mémoire et de dimension) les BTF, dans le but d'en effectuer un rendu en temps réel. Comme nous l'avons vu les méthodes de décomposition en bases linéaires offrent de meilleurs résultats que les méthodes qui tentent d'associer un modèle paramétrique à la BTF même lorsque ceux-ci ajoutent des paramètres d'échelle ou lorsqu'ils sont appliqués sur des larges sous-ensembles de la BTF.

La méthode donnant les meilleurs résultats en terme de qualité utilise une factorisation de sous-ensemble défini par condition d'observation mais cette méthode est aussi celle qui nécessite le plus de mémoire étant donné qu'il faut stocker un ensemble de textures et de poids pour chaque condition de vue constituant la BTF. La méthode utilisant une factorisation par texel n'exploite pas la cohérence qu'il existe entre les différentes ABRDF d'un même matériau, et ne permet donc pas d'aboutir à un résultat satisfaisant en terme de qualité même en utilisant un très grand nombre de composantes principales lors de la reconstruction. Suykens et al. [SVLD03] ont proposé d'ajouter une étape de clusterisation à cette méthode mais les artefacts engendrés par cette méthode la rendent difficilement exploitable, en particulier pour des BTF complexes de matériaux naturels.

Deuxième partie

Rendu de matériaux
hétérogènes en temps réel

Chapitre 6

Génération de BTF synthétique

Comme nous l'avons vu au chapitre 5.2.2, l'acquisition d'une BTF nécessite non seulement un matériel complexe et coûteux, mais aussi des échantillons de matériaux réels. Afin de pouvoir bénéficier d'une base de données de matériaux que l'on retrouve dans les scènes visuelles routières classiques (différents asphalts, pavés, végétation, etc.), et de pouvoir tester nos algorithmes de rendu pour plusieurs matériaux, nous avons mis en place une méthode de génération de BTF synthétique. Après avoir donné des précisions sur le format d'échantillonnage des BTF, nous allons détailler dans ce chapitre les étapes nécessaires à la génération de BTF synthétiques, depuis le calcul des images par des algorithmes de synthèse d'images non temps réels jusqu'à la création des textures constituant la BTF.

6.1 Méthode de génération

6.1.1 Format de l'échantillonnage de la BTF

La BTF est un ensemble discret de textures dont il convient de définir l'échantillonnage. Les premiers travaux de mesure effectués par Dana et al. [DvGNK99], ainsi que par l'Université de Bonn ont défini un échantillonnage considérant 81 points uniformément répartis sur un hémisphère pour le vecteur d'observation et d'éclairage. Ainsi, on obtient $81 * 81 = 6561$ mesures pour toute la BTF. Nous avons utilisé la même définition pour nos BTF synthétiques afin de pouvoir tester aisément nos algorithmes de rendu à la fois sur les BTF mesurées et sur nos BTF synthétiques. La résolution des textures, codées en RVB, est de 256x256 pixels (taille de texture très largement utilisée par le rendu 3D temps réel). La taille de la BTF ainsi générée est de :

$$81 * 81 * 256 * 256 * 3 = 1289945088 \text{ octets}$$

soit environ 1,29 Giga-octets.

6.1.2 Synthèse d'images photoréalistes non temps réel

Les échantillons synthétiques formant une BTF doivent atteindre le meilleur photoréalisme possible en s'approchant au mieux des échantillons réels mesurés. Pour les générer, il est possible d'utiliser des techniques de synthèse d'images non temps réel permettant d'aboutir à des échantillons de haute qualité. En particulier, nous utilisons des techniques de lancer de rayon couplé à des modèles d'éclairage adaptés (BRDF). Le temps nécessaire pour générer ces échantillons peut être conséquent, mais contrairement au rendu temps réel, il n'y a pas de contrainte forte sur le temps d'exécution nécessaire à la synthèse des textures.

Le logiciel *Povray* (*Persistence Of Vision Raytracer*) [Pov96] supporte les algorithmes classiques de synthèse d'images (lancer de rayons, lancer de photons, radiosité, etc.) mais aussi et surtout le contrôle de la scène virtuelle par des scripts. Ainsi, il est relativement aisé de réaliser un goniomètre virtuel en déplaçant la caméra et la source de lumière de façon automatique et pour chaque condition, de synthétiser une texture.

6.1.3 Données sources

Pour calculer un échantillon de BTF, un certain nombre de données sources est nécessaire :

- La géométrie de la surface (positions des sommets, surfaces paramétriques, champs de hauteur, bump-map, etc.).

- Les fonctions de réflectance (BRDF) des matériaux présents (modèles analytiques, mesures, etc.).
- Les positions angulaires de la caméra et de la source de lumière.
- Des marqueurs pour délimiter la zone de travail.

Nos premières démarches ont consisté à effectuer des mesures de BRDF d'asphalt, de manière à posséder des données sur les propriétés de réflexion globale des matériaux recouvrant les surfaces routières. Les données des BRDF sont mesurées avec un pas d'échantillonnage de 1° , sur un hémisphère complet pour la direction d'observation. Sur la figure 6.1 sont représentées en fausses couleurs (qui expriment les angles de réflexion) les BRDF que nous avons mesurées pour un même asphalt sec et humide.

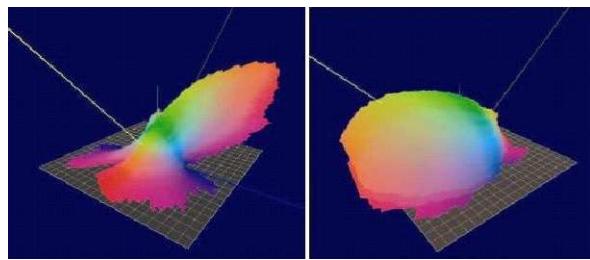


FIG. 6.1 – Mesures de BRDF : Asphalt humide (gauche). Asphalt sec (droite).

On peut y remarquer la forte spécularité de la BRDF de l'asphalt humide. En effet, lors d'une conduite de nuit sur route humide, l'oeil du conducteur reçoit peu de lumière des projecteurs de son propre véhicule mais peut être ébloui par la réflexion de l'éclairage des projecteurs des véhicules confrontant du trafic. Pour l'asphalt sec, la BRDF présente une grande rétrodiffusion, ce qui traduit que dans ces conditions le conducteur a une bonne visibilité de la route éclairée par les projecteurs de son véhicule. On peut voir que ces fonctions sont toutes deux très différentes d'un modèle de Phong classiquement utilisé en image de synthèse temps réel et que ce dernier ne peut ainsi suffir à simuler de manière correcte les propriétés de réflexion pour de tels matériaux.

Comme nous l'avons évoqué au chapitre 4.1.1, les mesures de BRDF posent quelques problèmes, en particulier pour des angles d'observation rasant pour lesquels les données mesurées sont très bruitées. Ce phénomène est particulièrement remarquable sur la figure 6.1 pour la BRDF humide. De telles erreurs nécessitent donc l'usage de filtres sur les données afin de les exploiter.

Afin de ne pas se confronter directement à ces problèmes et de débuter avec un matériau plus simple, nous allons dans un premier temps synthétiser

une BTF de pavés qui devrait présenter, de part sa géométrie, les effets que nous cherchons à simuler en temps réel (auto-masquage, auto-ombrage). Pour son calcul, un champ de hauteurs qui détermine la géométrie de la surface en chaque point de la texture a été utilisé (figure 6.2). Ne possédant pas de données physiques sur la réflexion des matériaux simulés, un modèle de Phong (avec des composantes diffuses et spéculaire) a été appliqué pour le calcul d'éclairage. Bien que celui-ci ne soit pas physiquement correct, il devrait permettre de synthétiser des échantillons suffisamment "visuellement réalistes" pour le rendu de BTF.

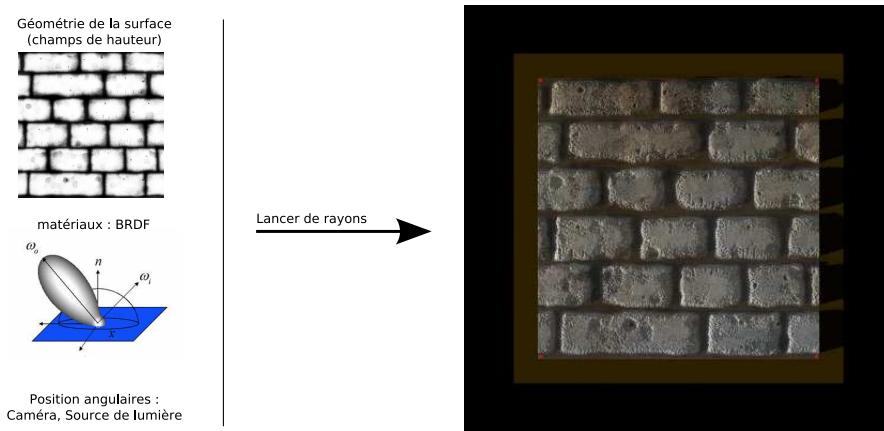


FIG. 6.2 – Principe de génération de BTF synthétique

6.1.4 Calcul des textures de la BTF

A partir des données sources présentées ci-dessus, un algorithme de lancer de rayon est appliquée afin d'estimer l'éclairage en chaque point de la surface. L'image est générée en utilisant une projection orthographique et non perspective afin de ne pas perdre d'informations sur la surface. En effet, avec une projection perspective, les coordonnées x_p et y_p projetées sur un plan d'un point 3D (x, y, z) sont déterminées à partir de la distance entre le plan de projection et ce point, ainsi plus les points sont éloignés du plan de projection, plus leurs coordonnées projetées se resserrent, écrasant le nombre d'informations projetées. La projection orthographique utilise des rayons parallèles évitant ce problème et ainsi de retrouver un maximum d'informations sur le plan projeté.

La source de lumière est directionnelle et les rayons émis sont tous parallèles (équivalent à un positionnement à l'infini de la source de lumière) afin qu'il n'y ait pas de différence d'éclairement sur l'ensemble de la surface pour

une même condition d'éclairage (vecteur ω_i constant).

Pour chaque échantillon, un script positionne la caméra et la source de lumière, lance le calcul et sauvegarde l'image ainsi générée. Voici un exemple du script permettant de positionner la source de lumière :

```
#declare lightClock = floor(clock/81);

#declare lightPosition =
<
ld*sind(angles(lightClock).x)*cosd(angles(lightClock).y),
ld*cosd(angles(lightClock).x),
ld*sind(angles(lightClock).x)*sind(angles(lightClock).y)
>

light_source
{
    lightPosition
    color White
    parallel
    fade_distance 0
}
```

6.1.5 Transformation homographique

Les images calculées ne sont pas exploitables directement et doivent subir une transformation afin de créer des textures constituant une BTF. En effet, chaque échantillon de texture correspond à une condition d'observation particulière, l'image contient donc un plan oblique qui ne couvre par la totalité des pixels et n'est donc pas utilisable comme une texture classique représentée dans un plan parallèle au plan de la caméra. Afin de constituer une BTF avec un ensemble de textures dans le plan de la caméra quelque soit la condition d'observation initiale, une transformation homographique doit être appliquée sur chaque texture (figure 6.3).

Une transformation homographique est une transformation linéaire bijective entre deux plans projectifs. C'est à dire qu'un ensemble de points 2D projectifs q_i sur un plan π_1 (dans son système de coordonnées) peut être projeté sur un deuxième plan π_2 en des points p_i donnés par :

$$p_i = H \cdot q_i$$

Où H est la matrice d'homographie :

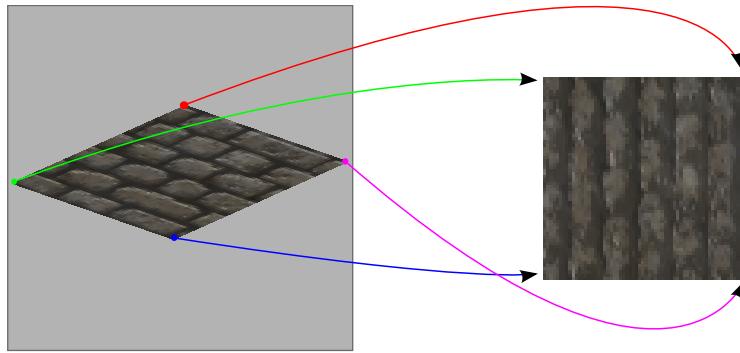


FIG. 6.3 – Transformation homographique

$$H = \begin{bmatrix} h_{11} & h_{21} & h_{31} \\ h_{12} & h_{22} & h_{32} \\ h_{13} & h_{23} & h_{33} \end{bmatrix}$$

Pour calculer les coefficients de la matrice d'homographie h_{ij} , des marqueurs repérables (par leurs couleurs) sont placés aux quatre coins de la surface afin de définir la correspondance entre les coins de l'image source et ceux de l'image de destination :

Pour chaque condition d'observation k , les quatre coins q_{ij} trouvés dans l'image source sont représentés dans la matrice Q_k suivante :

$$Q_k = \begin{bmatrix} q_{11} & q_{21} & q_{31} & q_{41} \\ q_{12} & q_{22} & q_{32} & q_{42} \\ q_{13} & q_{23} & q_{33} & q_{43} \end{bmatrix}$$

Ces points ont une correspondance dans l'image de destination (coordonnées des quatre coins d'un carré normalisé) :

$$P_k = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Pour chaque condition d'observation k , les coefficients de la matrice d'homographie H_k sont calculés, par :

$$P_k = H_k \cdot Q_k$$

Où $h_{k(33)} = 1$ car la matrice H_k s'applique à des vecteurs décris en coordonnées homogène. Ainsi, il reste huit inconnues qu'il est possible de calculer à partir des deux équations pour chacun des quatre coins. Une fois la matrice d'homographie calculée à partir des coins de l'image, il est possible de transformer l'ensemble des points de l'image source vers la texture

de destination. L'image source doit être de résolution supérieure à la texture de destination afin d'obtenir suffisamment de données pour les conditions d'observation rasantes pour lesquelles on obtient des plans particulièrement obliques qui ne couvrent qu'une faible partie de l'image source. Étant donné que les images source et destination sont dans un espace discret et que la correspondance des pixels entre les deux images n'est pas toujours exacte (valeurs flottantes), un filtrage bi-linéaire est appliqué lors de la transformation homographique afin que l'image de destination ne comporte pas d'effet de crénelage (*aliasing*) et soit représentative de l'image source.

6.2 Résultats

Les scripts nécessaires permettant d'exploiter les fonctionnalités du logiciel *Povray* ont été réalisés afin de créer un goniomètre virtuel. La transformation homographique a été implémentée en langage *C++* et un exécutable indépendant permet de l'appliquer sur l'ensemble des images synthétisées. La synthèse complète d'une BTF requiert plusieurs heures de calculs, en fonction de la complexité des modèles d'éclairage utilisés. Sur la figure 6.4, sont représentés quelques échantillons d'une BTF de pavés. L'intérêt de la représentation par BTF de ce type de matériau réside en la restitution des ombres entre les pavés ainsi que la possibilité d'ajouter une couche d'eau entre les pavés dont la réflexion est très spéculaire. Ainsi, nous avons un matériau type contenant des ABRDF très différentes (diffuses et spéculaires) ainsi que des phénomènes d'auto-ombrage et d'auto-masquage dus à la géométrie de la surface.

6.3 Conclusion

Nous avons vu qu'il est possible de générer des BTF synthétiques, en utilisant un goniomètre virtuel, à partir d'algorithme de synthèse d'images non temps réel et d'une description simple de la géométrie et radiométrie de la scène, ainsi que d'une transformation homographique. Cette démarche permet de fournir des données d'entrée, aux propriétés variées, pour les algorithmes de compression de BTF que nous allons présenter dans le chapitre suivant.

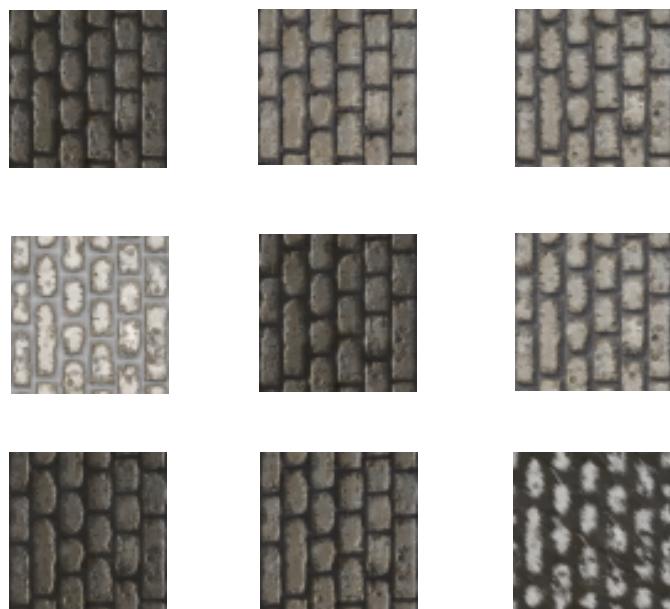


FIG. 6.4 – Échantillons de BTF synthétique (pavés)

Chapitre 7

Rendu temps réel de BTF pour la simulation de conduite

Dans ce chapitre, nous présentons les travaux effectués concernant le rendu réaliste temps réel de surface de route pour les simulateurs de conduite. En premier lieu, les premières tentatives basées sur des techniques classiques de bump-mapping et de fonctions de réflectance (BRDF) mesurées seront présentées. Comme nous l'avons déjà évoqué, ces techniques n'étant pas physiquement correct, il est nécessaire d'utiliser des méthodes plus complexes fondées sur l'utilisation de BTF. Ainsi, nous présentons un algorithme de rendu de BTF temps réel répondant aux contraintes particulières de la simulation de conduite que nous explicitons.

7.1 Étude préliminaire

7.1.1 Besoins et contraintes

Notre objectif est d'intégrer des matériaux réalistes dans le simulateur d'éclairage Renault. En particulier, la simulation d'éclairage d'asphalt sous différentes conditions météorologiques (route sèche et humide) permettrait d'étendre le périmètre de validité du simulateur d'éclairage pour l'évaluation des projecteurs automobile, mais aussi l'étude des stratégies d'éclairage dites intelligentes.

La première contrainte est donc d'effectuer une simulation réaliste des matériaux pour tous les angles d'éclairage et d'observation afin d'évaluer l'éclairage dans différentes conditions. Par exemple, on voudrait pouvoir évaluer l'éblouissement éventuel provoqué par la réflexion sur une route humide des projecteurs des véhicules du trafic confrontant. La deuxième contrainte importante réside dans le filtrage des textures. Contrairement aux applications courantes de réalité virtuelle utilisant des BTF, l'observation des surfaces dans un simulateur de conduite se fait pour un angle relativement rasant et jusqu'à de longues distances, en particulier en ligne droite. Ainsi, le rendu de surfaces de route devra impérativement supporté la technique de "*mip-mapping*" qui consiste à appliquer une texture de résolution adaptée à la distance et à l'angle d'observation de la surface. Enfin, l'aspect temps réel de simulation est primordial étant donné qu'une fréquence image trop faible peut affecter significativement la tâche du conducteur sur simulateur de conduite.

7.1.2 Domaine de visualisation de surfaces hétérogènes

L'utilisation d'une BRDF seule ne permet de simuler que partiellement l'aspect des matériaux hétérogènes comme l'asphalt de manière réaliste : pour des distances d'observation "proches", la granularité du matériau est perceptible et cet aspect rugueux n'est pas simulable par une BRDF étant donné que cette dernière est un bilan énergétique de la surface qui dépend d'ores et déjà de la géométrie de la surface.

Les variations locales de l'éclairage sont donc intégrées dans la BRDF et non pas restituées en chaque points du matériau comme nous le souhaitions. En revanche, pour des distances d'observation plus grande, la rugosité de la surface n'est pas perceptible (figures 7.1 et 7.2) et l'utilisation d'une BRDF pour représenter le matériau est tout à fait appropriée.

Afin de valider qu'il est possible de visualiser (et jusqu'à quelle distance)

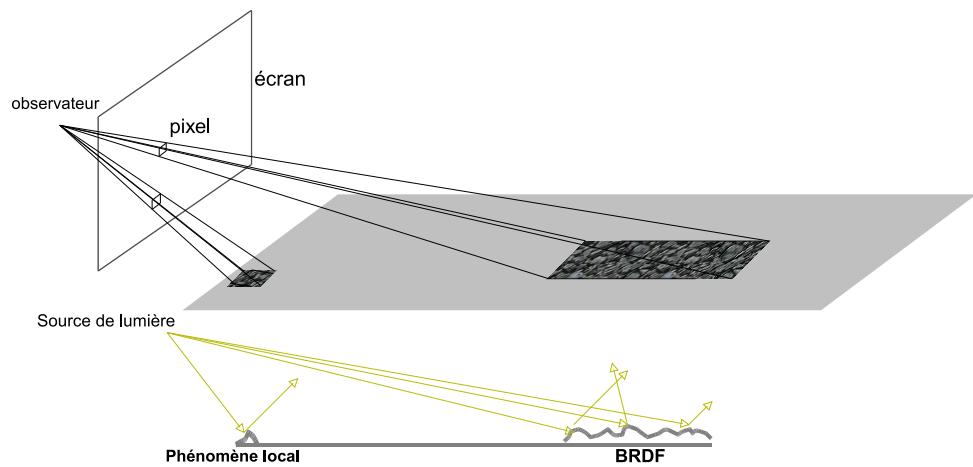


FIG. 7.1 – surfaces intégrées dans un pixel



FIG. 7.2 – Domaine d'usage approprié pour une BRDF

la rugosité de l'asphalt sur un système de visualisation d'un simulateur de conduite, il est nécessaire de calculer la surface de route que représente un pixel à l'écran. De manière simplifiée, on calcule un rayon passant par l'oeil de l'observateur et le centre d'un pixel à l'écran, puis on calcule l'intersection de ce rayon avec le plan de la route virtuelle. Ainsi, on connaît la distance entre chaque point d'intersection qui permet de connaître la résolution des détails que l'on pourra afficher sur la route.

La configuration du simulateur d'éclairage Renault est la suivante :

- Position de l'oeil de l'observateur (hauteur : 1,2 m, et distance à l'écran : 3 m)
- Résolution verticale : 1024 pixels
- Hauteur de l'écran : 2,5 mètres (posé au sol)

- Hauteur de la route virtuelle : 0 mètre.

L'équation du rayon dans ces conditions, en fonction de la hauteur du pixel au sol r se calcule par :

$$y = \left(\frac{r - 1.2}{3} \right) x + r \text{ et } x = \frac{-3r}{r - 1.2}$$

Les points x sur la surface de route (pour lesquels $y = 0$) permettent de calculer la distance pour chaque pixel de hauteur r . Les résultats sont représentés sur la figure 7.3.

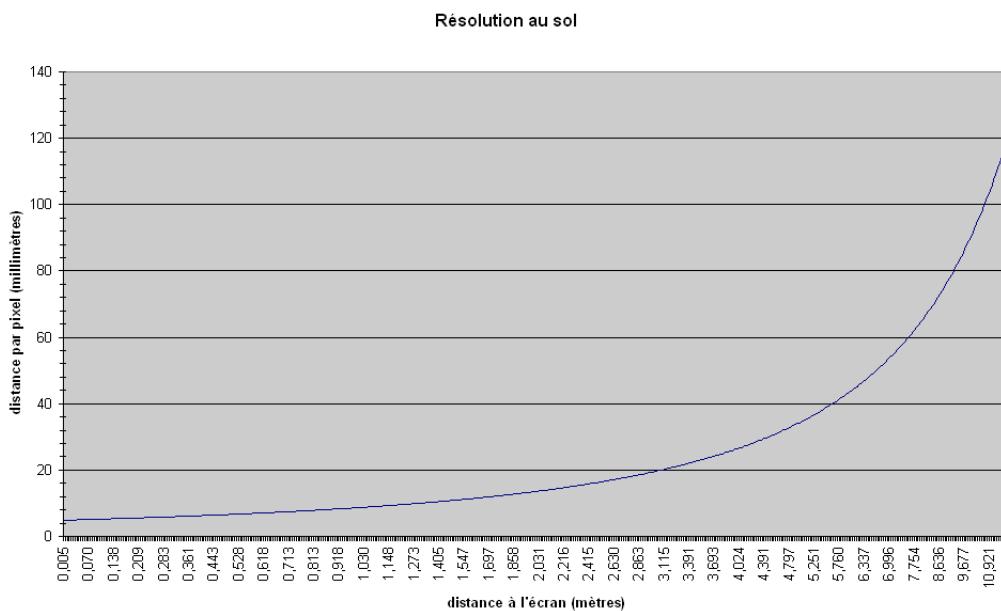


FIG. 7.3 – Résolution au sol (distance de la route représentée pas un pixel)

Les résultats montrent que dans ces conditions la plus petite distance (qui donne une estimation de la surface) contenue dans un pixel est de 4,9 mm et que celle-ci croît relativement vite (près de 2 cm à 3 mètres). Les distances pour lesquelles il est possible de visualiser un asphalt dont la géométrie à une résolution de l'ordre de 5 mm sont donc très courtes (jusqu'à 5 mètres de l'observateur). Cependant, même si un pixel représente une surface plus grande que l'échelle du matériau, les variations locales de l'éclairage peuvent être intégrées et fournissent alors des indices visuels sur l'état de la surface du matériau contenu dans ce pixel. Si l'on prend le cas de la photographie, même avec une résolution faible, la scène paraît réaliste. La résolution des vidéo-projecteurs étant toujours croissante, cette limite à tendance à reculer au cours des années. Il est donc important de simuler de manière fine et

réaliste des matériaux d'échelle plus fine que la résolution de l'écran en vue d'obtenir une observation réaliste de telles surfaces.

La méthode la plus classique pour simuler des variations géométriques à petite échelle consiste à appliquer une technique de perturbations de normales (*bump-mapping*) que nous avons déjà présentée au chapitre 4.3.4. Nous allons voir un exemple de mise en œuvre de cette technique pour le rendu de surface de route humide.

7.2 Rendu de surface de route par BRDF

7.2.1 Rendu avec Bump-mapping

Les BRDF d'asphalt mesurées (fig. 6.1) ont été combinées avec la technique de *bump-mapping* [Bli78] (technique de perturbation de normales) afin de simuler l'aspect rugueux de la route. Bien que nous ayons déjà évoqué les problèmes de validité de cette méthode, nous souhaitons, en première approche, estimer le niveau de réalisme visuel qu'il est possible d'obtenir avec cette méthode.

Afin de pouvoir utiliser en temps réel ces mesures, nous avons considéré des propriétés d'isotropie des BRDF mesurées. De cette manière, en négligeant l'angle de rotation horizontal de la direction d'observation, les BRDF sont des fonctions à trois dimensions, et nous pouvons les stocker dans des textures 3D disponibles dans les cartes graphiques standards du marché.

En chaque pixel, la normale perturbée modifie les angles avec lesquels on accède à la valeur de la BRDF. Ainsi, on simule la variation d'éclairage (figure 7.4) pour modéliser la granularité du matériau.

Le problème de cette méthode provient du fait qu'il n'est pas tout à fait correct de simuler le grain de la route étant donné que la BRDF intègre déjà les phénomènes lumineux dus à la rugosité de la surface. Avec cette technique seuls les accessoires (angles) de la BRDF sont modifiés et non pas la BRDF en elle-même, alors qu'en chaque point de la surface, une BRDF différente devrait décrire les propriétés locales de réflexion du matériau. Pour palier à ce problème, nous allons voir maintenant comment utiliser une BTF qui caractérise les propriétés de réflexion d'une surface en chaque point. En particulier, nous allons proposer une méthode pour la compression et rendu temps réel de BTF concernant la simulation des surfaces hétérogènes comme l'asphalt ou encore les routes pavées humides.



FIG. 7.4 – Simulation d'éclairage de route bosselée (BRDF mesurée et *bump-mapping*)

7.3 Algorithme de rendu temps réel par BTF

7.3.1 Représentation des données et notations

Nous allons décrire les étapes successives de notre algorithme de rendu. Nous nous basons sur un échantillon de BTF mesurée par l'Université de Bonn [MMS⁺] pour illustrer nos explications. Après chaque étape, nous représentons graphiquement l'ensemble des données de la BTF afin d'avoir une vision générale des transformations appliquées. La BTF, dans son état d'origine est constituée d'un ensemble de textures représenté schématiquement sur la figure 7.5.

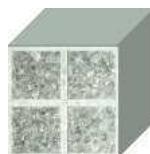


FIG. 7.5 – Représentation schématique de l'ensemble des textures de la BTF d'origine

Par la suite, notons $F(p, \omega_i, \omega_o)$ la BTF où p est un point en deux dimensions (x_p, y_p) , et ω_i et ω_o respectivement les directions d'éclairage et d'observation. Une texture de la BTF pour une condition d'éclairage et d'observation données sera notée $F^{\omega_i, \omega_o}(p)$ et l'ABRDF d'un point $F^p(\omega_i, \omega_o)$.

7.3.2 Changement d'espace de couleur pour la compression

La première étape de la compression consiste à transformer l'espace des couleurs des textures constituant la BTF. L'idée est de se baser sur des techniques utilisés pour la vidéo qui n'utilise pas l'espace de couleur RVB mais l'espace YCrCb (Luminance, Chrominance). Ce modèle a initialement été mis au point afin pouvoir transmettre des informations colorées aux téléviseurs couleurs, tout en s'assurant que les téléviseurs noir et blanc existant continuent d'afficher une image en tons de gris. C'est un standard défini par l'ITU (Union internationale des télécommunications) utilisé encore aujourd'hui pour la télévision sous le nom de ITU-R BT.601. Ce changement d'espace de couleur est également exploité dans la très répandue méthode de compression d'image JPEG.

Cette conversion a un double intérêt : l'oeil humain est moins sensible aux variations de chrominance qu'aux variations de luminance (Y) et les valeurs de chrominance (Cb, Cr) ont généralement une variance relativement faible dans les images naturelles (fig. 7.6) surtout pour une image représentant un même matériau. Ainsi, les canaux Cb et Cr de l'image sont relativement diffus et une compression plus importante (comparativement à la représentation RVB) pourra y être appliquée tout en minimisant la perte de données, la plus grande variance se trouvant dans le canal Y (luminance).

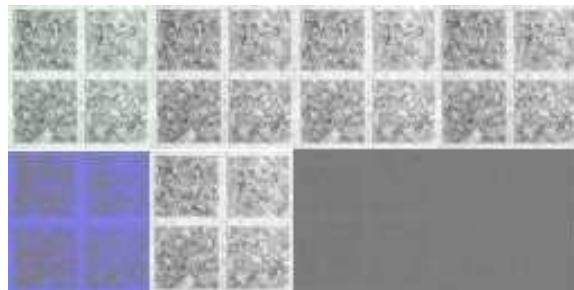


FIG. 7.6 – L'image RVB (en haut à gauche) est décomposée (haut) selon ses composantes R, V et B. La même Image YCrCb (en bas à gauche) est décomposée (bas) selon ses composantes de Luminance (Y), et de chromacité (Cr et Cb).

Cette transformation classique et sans pertes, peu coûteuse en terme de calculs, s'effectue de la manière suivante :

$$\begin{aligned} Y &= 0.299 * R + 0.587 * V + 0.114 * B; \\ Cb &= 0.5 - 0.168736 * R - 0.331264 * V + 0.5 * B; \\ Cr &= 0.5 + 0.5 * R - 0.418688 * V - 0.081312 * B; \end{aligned}$$

Au terme de l'algorithme de compression, la transformation inverse (YCrCb en RVB), exprimée par la relation suivante, devra être appliquée :

$$\begin{aligned} R &= Y + 1.40200 * (Cr - 0.5); \\ V &= Y - 0.34414 * (Cb - 0.5) - 0.71414 * (Cr - 0.5); \\ B &= Y + 1.77200 * (Cb - 0.5); \end{aligned}$$

Après avoir appliqué cette transformation, la BTF est constituée d'un ensemble de textures au format YCrCb :

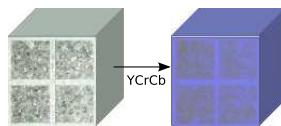


FIG. 7.7 – Représentation de la BTF dans l'espace de couleur YCrCb

7.3.3 Pyramide Laplacienne

Une image $F^{\omega_i, \omega_o}(p)$ de la BTF est décomposée en sous-bandes (sous-images) en calculant une pyramide Laplacienne [BA87], indépendamment pour chaque canal Y, Cr, Cb . Une pyramide Laplacienne consiste à représenter une image (F) par une série de k images.

Tout d'abord, on définit $F_0^{\omega_i, \omega_o}(p) = F^{\omega_i, \omega_o}(p)$ qui est l'image d'origine de la BTF. Puis, on calcule F_{k+1} en appliquant sur le niveau k de la pyramide, un filtre passe-bas ($g(p)$) en chaque point p et une réduction de la taille de l'image notée \downarrow_p :

$$F_{k+1}^{\omega_i, \omega_o}(p) = \downarrow_p g(p) * F_k^{\omega_i, \omega_o}(p)$$

La dimension de $F_k^{\omega_i, \omega_o}(p)$ est égal à $1/(2^k)$ de la dimension de la BTF d'origine, le facteur d'échelle \downarrow_p d'un niveau à l'autre de la pyramide étant égal à un demi. Le filtre passe-bas $g(p)$ consiste à calculer la moyenne locale au point p des pixels dans une matrice de convolution (de taille 2x2).

Pour reconstruire l'image au niveau supérieur, une opération d'agrandissement \uparrow_p est appliquée. Elle consiste à remplir chaque groupe de pixels (de taille 2x2) par la valeur moyenne locale calculée précédemment.

Ensuite, à partir des $F_k^{\omega_i, \omega_o}(p)$, on calcule l'image L_k de niveau k de la pyramide, en calculant la différence entre deux niveaux, soit :

$$L_k^{\omega_i, \omega_o}(p) = F_k^{\omega_i, \omega_o}(p) - \uparrow_p F_{k+1}^{\omega_i, \omega_o}(p)$$

Étant donné que l'image L_k d'un niveau k , ne contient que la différence avec le niveau $k+1$ (calculé avec le filtre moyen), les pixels de l'image L_k sont proches et centrés autour de 0 et celle-ci peut être ainsi fortement compressée (figure 7.8). Seul le dernier niveau de la pyramide contient des données positives non proches de zéro, mais la résolution de l'image de ce niveau étant très faible (par exemple : 4x4 pixels), peu de données doivent être stockées. De plus, les opérations nécessaires au calcul de la pyramide sont relativement simples, ce qui rend l'implémentation de la reconstruction de $F_{k+1}^{\omega_i, \omega_o}(p)$ à partir des sous niveaux de la pyramide $L_k^{\omega_i, \omega_o}(p)$ envisageable en temps réel. Le nombre d'étages maximal de la pyramide dépend de la dimension de l'image constituant la BTF d'origine : pour une image de dimension 2^r , on aura $r+1$ étages pour une pyramide dont le dernier étage est constitué d'une image de dimension 1 (exemple : pour une image de départ de dimension 64, la pyramide aura 7 étages au maximum). Il est toutefois possible d'arrêter la construction de la pyramide à un certain niveau k_{max} avant d'atteindre le dernier niveau, et nous verrons comment exploiter cette propriété plus loin dans l'algorithme de compression ($1 \leq k_{max} \leq r$).

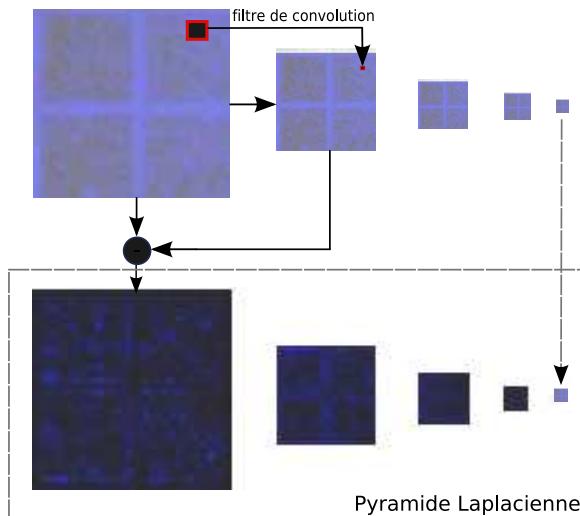


FIG. 7.8 – Pyramide Laplacienne à 5 niveaux (64, 32, 16, 8 et 4)

Chaque niveau de la pyramide peut être utilisé comme un niveau de *mipmapping* de la BTF, ce qui répond à la contrainte que nous avions exposée dans le cadre de l'utilisation de l'algorithme pour la simulation de conduite.

Comme nous pouvons le voir sur la figure 7.9, suite à la création de pyramide, le nombre de données a considérablement augmenté, ce qui semble

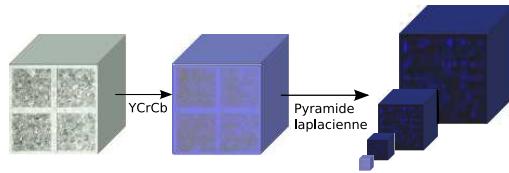


FIG. 7.9 – Représentation de la BTF avec une pyramide Laplacienne

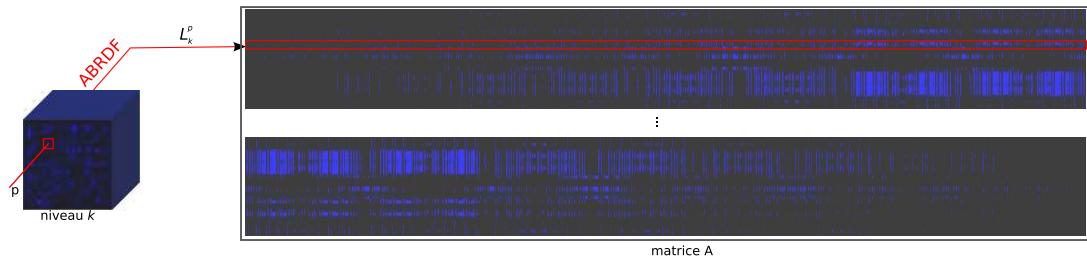
contraire à notre objectif de représentation compacte de la BTF. Nous allons cependant voir maintenant que la principale étape de compression de la BTF tire profit de cette pyramide Laplacienne.

7.3.4 Compression par analyse en composantes principales

L’analyse en composantes principales (chapitre 5.3.2) constitue l’étape majeure de notre algorithme visant à compacter la BTF. Nous allons détailler comment celle-ci est appliquée indépendamment pour chaque niveau k de la pyramide laplacienne.

Tout d’abord, on définit la BTF Laplacienne $L_k(p, \omega_i, \omega_o)$ d’un niveau k par l’ensemble des images $L_k^{\omega_i, \omega_o}(p)$ générées à l’étape précédente.

Chaque BTF Laplacienne L_k peut être également interprétée comme un ensemble de BRDF Laplacienne par texel. Celles-ci sont représentées, pour chaque point p , sous forme de vecteurs L_k^p (de longueur $81 * 81 = 6561$) que l’on juxtapose (en colonnes) de manière à former une grande matrice A (figure 7.10).

FIG. 7.10 – Schéma représentant les ABRDF d’un niveau k de la pyramide transformées en vecteurs L_k^p , puis assemblées dans une matrice A

On applique ensuite une analyse en composantes principales indépendamment pour chaque canal Y, Cb et Cr de cette matrice. Cette méthode permet d’exploiter la redondance d’information sur toute la BTF. En effet, il est fort probable que bon nombre de texels de la texture aient des BRDF très

semblables, ce qui permet de trouver les n premiers vecteurs "dominants" pour le niveau k : e_{kj} avec $j = 1..n$. On peut alors approximer Les BRDF Laplaciennes L_k^p par une combinaison linéaire des vecteurs e_{kj} (composantes principales) :

$$L_k^p(\omega_i, \omega_o) \approx \left(\sum_{j=1}^n d_{kj}(p) e_{kj}(\omega_i, \omega_o) \right) + \mu_k(\omega_i, \omega_o)$$

Où $d_{kj}(p)$ représente le poids de chacune des composantes principales pour un texel p et μ_k la moyenne de la BRDF Laplacienne (figure 7.11). Les propriétés de la pyramide Laplacienne font que cette moyenne est nulle sauf pour le dernier niveau ($\mu_{\lambda_{\max}}$), ce qui permet d'optimiser la compression et d'exprimer la BTF Laplacienne d'un niveau k par :

$$L_k(p, \omega_i, \omega_o) \approx \sum_{j=1}^n d_{kj}(p) e_{kj}(\omega_i, \omega_o)$$

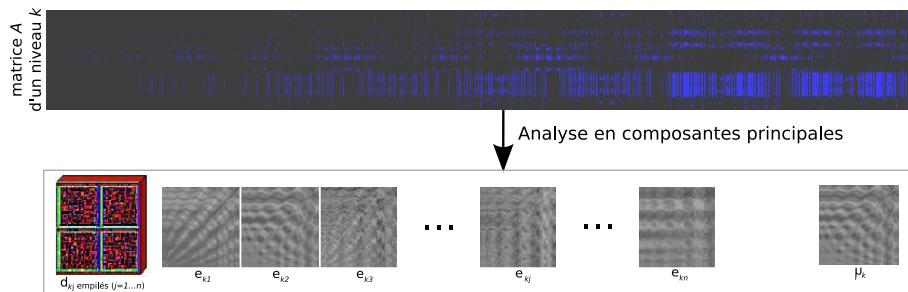


FIG. 7.11 – Schéma représentant les données normalisées issues de l'analyse en composantes principales

Afin de représenter la BTF avec beaucoup moins de données qu'à l'origine, seules les n premières composantes principales sont retenues. Cette étape permet "compresser" la BTF : un niveau donné de la pyramide ayant une résolution r contient r^2 ABRDF et il est possible de n'utiliser que n composantes principales les plus représentatives de la BTF. Par exemple, pour un niveau de résolution de 64 pixels, on peut choisir de représenter ses 4096 ABRDF par 8 composantes principales, réduisant considérablement le nombre de données nécessaires à la reconstruction de la BTF. Toutefois, cette étape engendre des pertes de données et il convient alors de définir le rapport entre le nombre de composantes principales qu'il est judicieux de prendre en compte et la perte due à la compression des données.

Pour évaluer le poids des composantes principales de chaque niveau k , il est intéressant d'étudier les valeurs propres fournies par le calcul de l'analyse

en composantes principales. En effet, ces valeurs traduisent la variance des données sur chaque axe du nouvel espace déterminé par les composantes principales. Plus la variance des données projetées sur une composante principale est grande, plus cette dernière contient d'informations de la BTF d'origine et il convient alors de la prendre en compte pour la reconstruction de la BTF.

Le calcul de l'analyse en composantes principales permet d'obtenir les composantes principales par ordre décroissant de valeurs propres, ainsi les n premières composantes calculées sont les plus représentatives du matériau. Sur les figures 7.12 et 7.13, sont respectivement représentées les valeurs propres du canal de luminance (Y) et de chromacité (Cr et Cb) pour un niveau k de la pyramide Laplacienne (niveau de résolution 64). Pour les canaux de luminance et de chromacité, on peut voir que les valeurs propres décroissent très rapidement : à partir de la huitième composante principale, la variance est environ 10 fois moindre que pour la première composante principale (3,5 contre 42 pour Y).

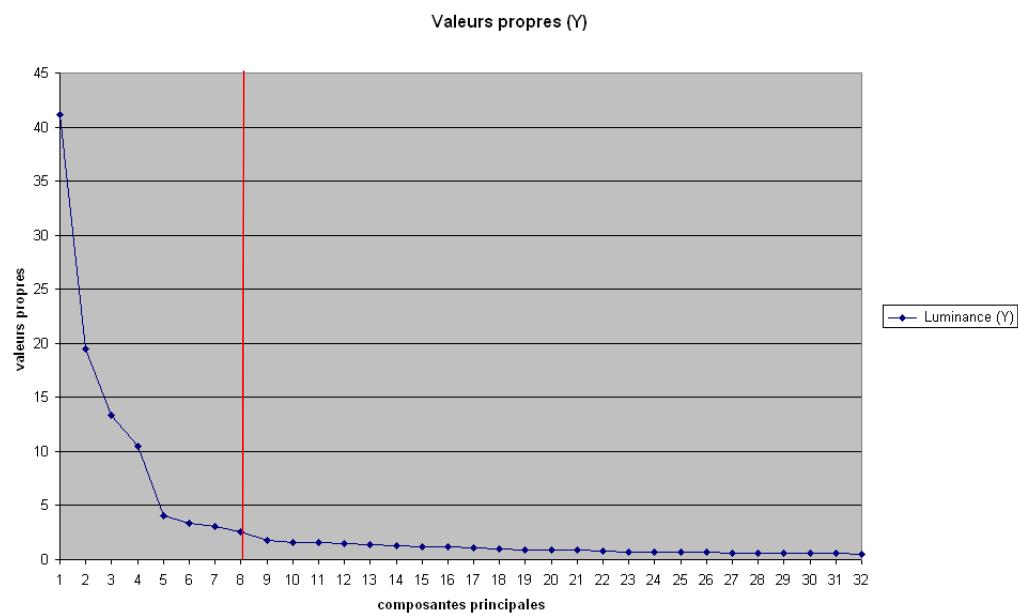


FIG. 7.12 – Valeurs propres pour le canal Y (luminance)

Pour le canal Y , qui contient la luminance de la BTF, il est nécessaire de retenir plus de composantes principales que pour les canaux Cb et Cr car la composante de luminance à une variance beaucoup plus grande que les composantes de chromacité. Ceci est particulièrement visible sur les figures

7.12 et 7.13. La variance de la luminance est environ 40 fois supérieure à la variance de la chromacité.

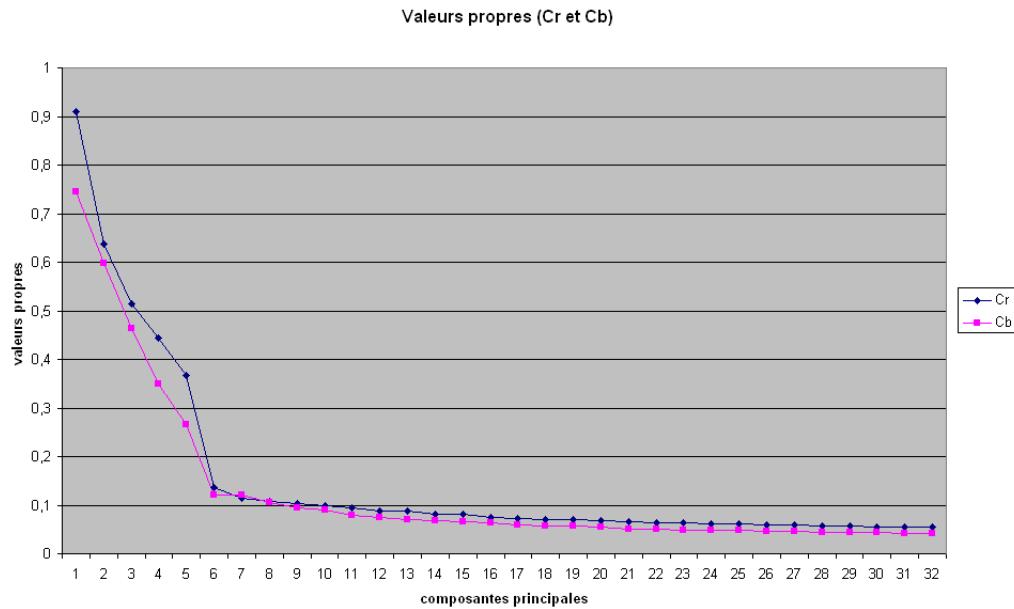


FIG. 7.13 – Valeurs propres pour les canaux Cr et Cb (chromacité)

Grâce au changement d'espace de couleurs, on obtient ainsi une représentation plus compacte de la BTF : à titre indicatif, nous pouvons garder huit composantes principales pour la luminance et seulement deux pour chaque composante de chromacité, ce qui donne douze composantes principales pour représenter un seul niveau de la pyramide. Le taux d'erreur introduit par la réduction du nombre de composantes principales sera exposé dans le chapitre validation. Rappelons que pour reconstruire exactement la BTF, pour tous les niveaux k , il est nécessaire de considérer l'ensemble des n vecteurs e_k fourni par l'analyse en composantes principales.

La construction de la pyramide est arrêtée lorsque la dimension du dernier étage (k_{max}) est légèrement supérieur au nombre de composantes principales nécessaire à la représentation de la BTF pour ce niveau. Par exemple, si l'on retient 8 composantes principales, on s'arrêtera à une dimension de 4 (qui contient 16 ABRDF), car le niveau suivant (de dimension 2) ne contient que 4 ABRDF ce qui est inférieur au nombre de composantes principales.

Pour reconstruire la BTF, il est également nécessaire de stocker les poids

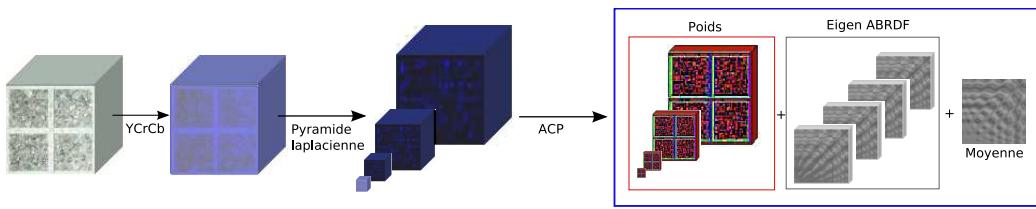


FIG. 7.14 – Analyse en composantes principales de la pyramide Laplacienne

qui correspondent à la BTF d'origine projetée dans le nouvel espace défini par les composantes principales ainsi que la moyenne du dernier niveau $\mu_{k_{\max}}$ de la pyramide qui contient les variations globales du matériau. L'ensemble des données issues de cette étape est représenté sur la figure 7.14.

7.3.5 Représentation des variations globales du matériau par une fonction analytique

Les variations globales du matériau ($\mu_{k_{\max}}$) sont issues des étapes successives de filtrage passe-bas, et ne contiennent donc pas de fortes discontinuités. On peut ainsi envisager de trouver une fonction paramétrique pour décrire cette composante. En particulier, on peut essayer de représenter les variations globales du matériau par un modèle de Phong, très facile à implémenter en temps réel.

Un autre intérêt réside dans le fait qu'une fonction continue, telle que Phong, permet de générer des images sans artefacts visuels entre les angles pour lesquels la BTF est mesurée tandis qu'une simulation directe à partir de l'échantillonnage décrit plus haut engendre des discontinuités dans le rendu et nécessite une interpolation linéaire et donc une approximation pour des angles intermédiaires (non présentes dans les données mesurées). Pour rappel les paramètres du modèle de Phong recherché sont les suivants :

$$\mu'_{k_{\max}}(\vec{\omega}_i, \vec{\omega}_o) \approx \rho_a + \rho_d(\vec{N} \cdot \vec{\omega}_i) + \rho_s(\vec{N} \cdot (\vec{\omega}_i + \vec{\omega}_o))^n$$

Où $\vec{N} = (0, 0, 1)$ et $\rho_a, \rho_d, \rho_s, n$ respectivement les coefficients de réflexion ambiant, diffus et spéculaire.

Cette méthode ne s'applique que pour le canal de luminance Y car les composantes chromatiques Cr et Cb sont quasiment constantes (figure 7.15) pour ce matériau, et sont alors approximées par une valeur moyenne :

$$\mu'_{k_{\max}}(\vec{\omega}_i, \vec{\omega}_o) \approx \bar{c}$$

Cette approximation devrait être valable pour une très grande majorité de matériaux : la variation de colorimétrie selon les angles d'éclairage et

d'observation ne devrait avoir lieu que pour des matériaux très spécifiques tels que des peintures à effets. Dans ce cas, il serait nécessaire d'appliquer la méthode présentée pour les trois canaux Y, Cb et Cr.

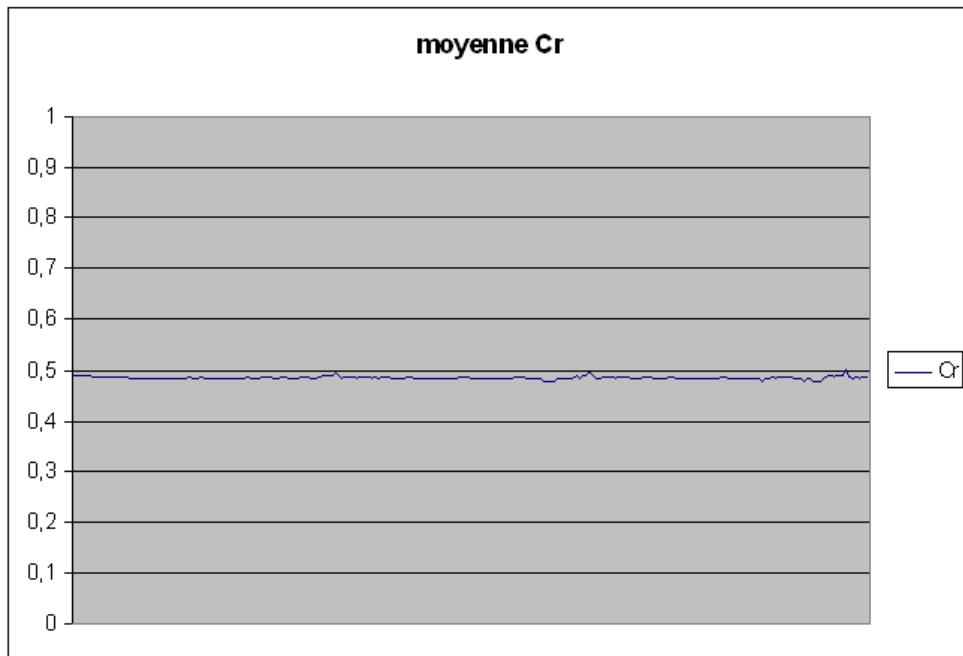


FIG. 7.15 – Valeurs de $\mu_{k_{\max}}$ pour la composante de chromacité Cr

Pour trouver les différents paramètres du modèle de Phong k_a, k_d, k_s, α à partir des données issues de notre algorithme, il faut utiliser une méthode d'optimisation non-linéaire étant donné que les paramètres ne sont pas linéaires entre eux (exposant spéculaire). Nous avons utilisé la méthode de Levenberg Marquardt [Mar63] dont la description et le code source sont fournis en annexe (annexe B).

Implémentation et résultats

Nous avons utilisé le logiciel *Scilab* [Sci89] qui propose une implémentation de cet algorithme. Les résultats montrent qu'il est tout à fait possible de trouver les paramètres du modèle de Phong à partir des données issues de notre algorithme, même avec des paramètres initiaux éloignés de leurs valeurs finales. Nous l'avons appliqué sur plusieurs matériaux (réels ou synthétiques), et l'algorithme a systématiquement convergé, donnant les paramètres du modèle de Phong correspondant. Sur la figure 7.16, on peut voir une représentation des données de départ pour notre matériau.

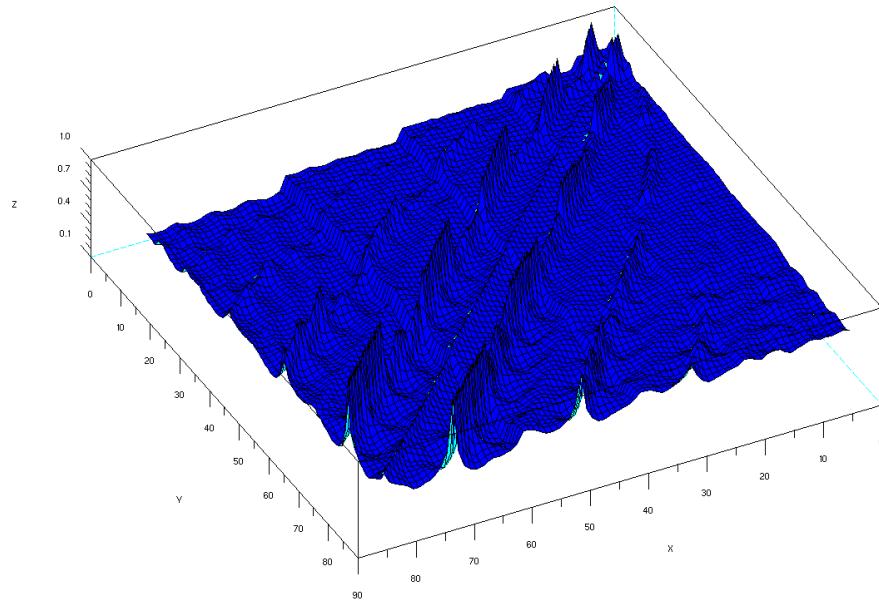


FIG. 7.16 – Données de variations globales du matériaux $\mu_{\lambda_{\max}}$

Les axes x et y correspondent respectivement aux vecteurs d'éclairage et d'observation, l'axe z exprime la valeur normalisée de la réflexion du matériau. Sur la figure 7.17 est représenté le résultat de l'algorithme de minimisation qui correspond aux valeurs retournées par le modèle de Phong avec les paramètres trouvés par la méthode de Levenberg-Marquardt.

On peut voir que l'allure générale de la fonction semble globalement respectée. Toutefois, certains détails de plus hautes fréquences ne sont pas restitués, ce qui provient du fait que de telles variations ne peuvent être représentées par le modèle de Phong qui est trop simpliste. L'erreur ne vient donc pas de la méthode d'optimisation non-linéaire mais réside dans le choix initial de la fonction pour laquelle les paramètres sont calculés. Optimiser le résultat consisterait à trouver de manière intuitive un modèle plus représentatif des données initiales (figure 7.16), mais cela est loin d'être aisé car il n'est pas facile d'imaginer la fonction la plus adaptée à un ensemble de telles données. De plus, un modèle qui serait plus adapté serait probablement aussi plus complexe (en terme de nombre de calculs), ce qui nous éloignerait de notre cible qui reste l'implémentation en temps réel de l'algorithme.

Après cette dernière étape dans notre algorithme de compression, les don-

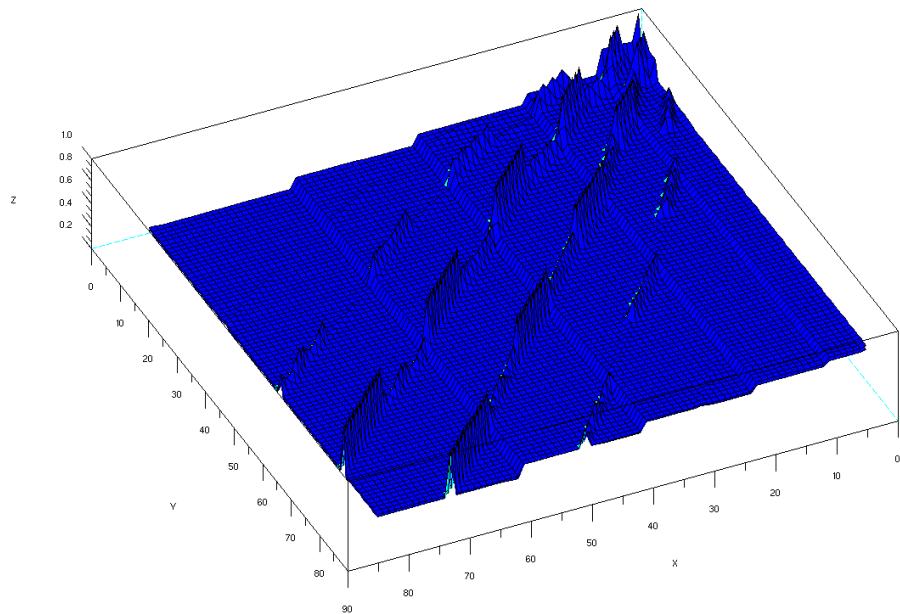


FIG. 7.17 – Modèle de Phong représentant les variations globales du matériau $\mu'_{\lambda_{\max}}$

nées nécessaires au rendu sont représentées sur la figure 7.18.

Nous allons maintenant présenter comment exploiter ces données dans le cadre d'un rendu en temps réel.

7.4 Implémentation temps réel

L'ensemble des données issues de l'algorithme de compression doit être chargé dans la mémoire de la carte graphique afin de réaliser le rendu en temps réel. Nous allons maintenant décrire pour chacune de ces données, comment il est possible de les représenter dans les textures disponibles (2D et 3D). En particulier nous verrons les contraintes liées à l'interpolation et les implications en terme de paramétrisation des données.

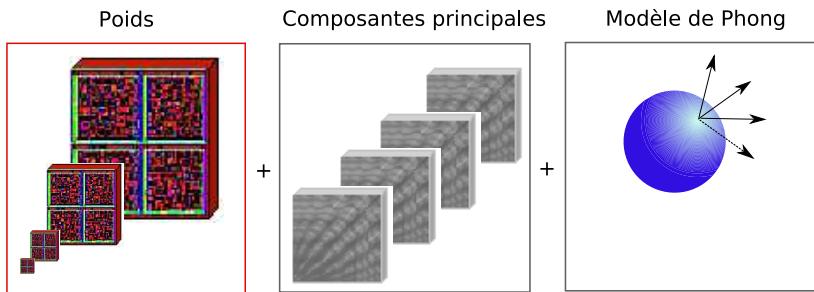


FIG. 7.18 – Données issues de l'algorithme de compression pour le rendu

7.4.1 Paramétrisation des données pour le rendu

Les poids $d_{kj}(p)$ des composantes principales sont des images 2D, il est donc facile de les représenter par des textures 2D classiques. Toutefois, 12 images (8 pour la composante de luminance Y et 2 pour chacune des composantes de chromacité Cr et Cb) sont nécessaires pour reconstruire la BTF Laplacienne à un niveau k . Il est alors plus judicieux de créer trois textures à quatre composantes (au format RGBA) pour assembler les données. Elles sont alors empilées ces trois textures pour chaque niveau de la pyramide dans une texture 3D. Ainsi, une seule unité de texture est requise pour représenter l'ensemble de ces données.

Nous avons vu que la composante $\mu_{\lambda_{\max}}$ est représentée par le modèle d'éclairage de Phong et ne pose donc pas de problème particulier, si ce n'est la sauvegarde des paramètres.

En revanche, les composantes principales (figure 7.19) ne sont pas exploitable aussi facilement pour le rendu : ce sont des fonctions à 4 dimensions qui n'existent pas nativement dans l'API graphique OpenGL. De plus, l'échantillonnage des pas de mesure n'est pas constant, ce qui pose deux problèmes :

- Il est nécessaire d'utiliser une table d'indirection ("Look Up Table") pour accéder au bon texel en fonction des conditions d'éclairage et d'observation, ce qui augmente le nombre de calculs pour l'accès aux données.
- Il n'est pas possible de bénéficier de l'interpolation linéaire entre deux texels voisins (gérée nativement par les cartes graphiques). Sans interpolation, le rendu comportera des artefacts (discontinuités) dû au caractère discret de l'échantillonnage des mesures.

L'objectif est donc d'effectuer une "reparamétrisation" des composantes

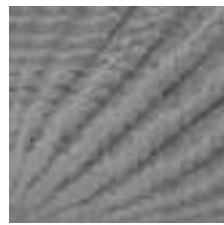


FIG. 7.19 – Composante principale normalisée représentée dans une image 2D de taille 81*81 dont l'accès aux données est difficile.

principales pour optimiser la vitesse et la qualité du rendu. En particulier, nous cherchons une paramétrisation pour laquelle les vecteurs d'éclairage et d'observation permettront un accès facile aux données et surtout d'en effectuer une interpolation correcte. Nous allons proposer maintenant une solution à ce problème, qui tend à respecter les règles suivantes :

- **Préservation de l'aire relative occupée par un sous-ensemble de données** : Le rapport entre l'aire $a(R)$ d'un sous-ensemble de données $R \in M$ et l'aire de l'image initiale $a(M)$ ne doit pas être modifié lors de la transformation en un autre sous-ensemble $m(R)$ sur un hémisphère d'aire $a(H)$ (figure 7.20), soit :

$$\frac{a(R)}{a(M)} = \frac{a(m(R))}{a(H)}$$

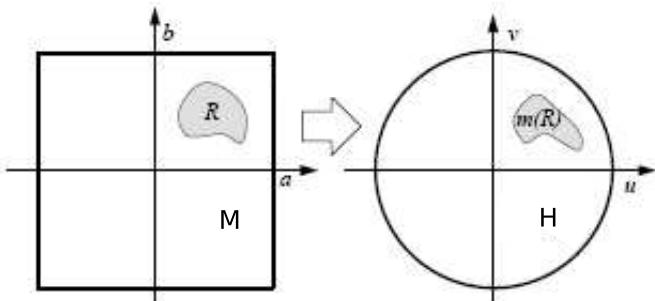


FIG. 7.20 – Préservation du rapport de l'aire du sous-ensemble M lors de la reparamétrisation.

- **bicontinuité** : La paramétrisation et son inverse doit être continue de manière à préserver l'adjacence des données. Cette propriété est nécessaire car les distances sur le carré devront correspondre à des angles sur l'hémisphère.

- **distortion** : la distortion doit être faible afin de préserver le maximum de données lors de la reparamétrisation.

Une carte concentrique élevée [SC97] satisfait les propriétés citées ci-dessus et permet de transformer une distribution uniforme de points sur un carré en une distribution uniforme de points sur un hémisphère (figure 7.21). Celle-ci peut être définie comme une concaténation d'une carte d'élévation Ψ et d'une carte concentrique Φ

$$\Omega(s, t) = (\Psi \circ \Phi)(s, t)$$

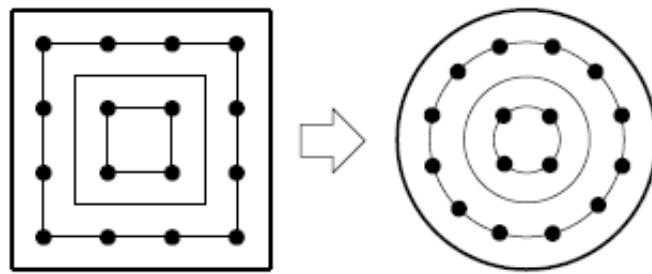


FIG. 7.21 – Ensemble de points uniformément répartis sur un carré reparamétrisés sur une carte concentrique élevée.

La carte concentrique $\Phi(s, t)$ est une fonction transformant des carrés concentriques en des disques concentriques. La méthode consiste à prendre un carré défini par $(s, t) \in [-1, 1]^2$ et à diviser ce carré en quatre régions par deux lignes d'équations $s = t$ et $s = -t$. Pour la première région (figure 7.22), la carte concentrique $\Phi(s, t) = (\rho(s, t), \theta(s, t))$ est définie par :

$$\rho(s, t) = s \text{ et } \theta(s, t) = \frac{\pi}{4} \frac{t}{s}$$

L'angle θ varie entre $[-\frac{\pi}{4}, \frac{\pi}{4}]$ pour cette région, et des transformations similaires sont appliquées pour les autres régions afin de couvrir 2π radians.

Cette carte concentrique peut être étendue à un hémisphère en projetant les points du disque sur un hémisphère de distribution uniforme. En effet, si le disque a une distribution uniforme, alors la distribution de ρ^2 l'est également. Reste à déterminer la composante z en fonction de ρ^2 et d'assigner pour chaque z le point de coordonnées x, y . La carte d'élévation est alors définie par :

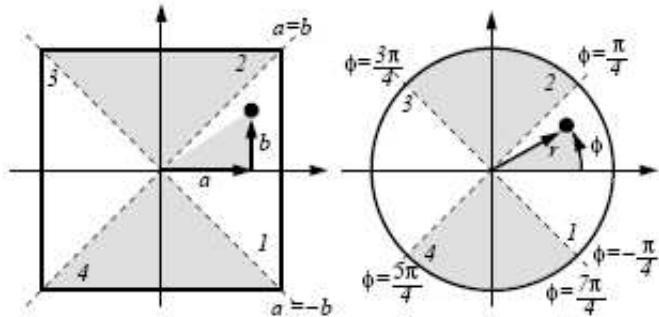


FIG. 7.22 – Les quatre régions formant la carte concentrique.

$$\begin{aligned}x &= \rho \sqrt{2 - \rho^2} \cos \theta \\y &= \rho \sqrt{2 - \rho^2} \sin \theta \\z &= 1 - \rho^2\end{aligned}$$

La carte concentrique est inversible :

$$(s, t) = \Omega^{-1}(x, y, z) = \Omega^{-1}\left(x, y, \sqrt{1 - x^2 - y^2}\right)$$

Il est donc possible de reparamétriser les composantes principales e_{kj} selon les variables (s,t) comme suit :

$$e_{kj}(\omega_i, \omega_o) = e_{kj}(\theta_i, \varphi_i, \theta_o, \varphi_o) = \tilde{e}_{kj}(s_i, t_i, s_o, t_o)$$

Les composantes principales sont toujours des fonctions à quatre dimensions. Nous allons maintenant détailler, comment après cette reparamétrisation, elles vont être organisées dans des textures 3D de manière à exploiter les capacités de filtrage de texture implémentées au sein des cartes graphiques.

Pour une composante principale $\tilde{e}_{kj}(s_i, t_i, s_o, t_o)$, on discrétise un paramètre, par exemple t_i , en un ensemble $\{t_{l_0}, t_{l_1}, \dots, t_{l_n}\}$. Nous obtenons alors un ensemble de fonctions volumétriques (à 3 dimensions) $\{S_0, S_1, \dots, S_n\}$ pour lesquelles t_i est fixé :

$$S_l(s_i, s_o, t_o) = \tilde{e}_{kj}(s_i, t_l, s_o, t_o)$$

Sur la figure 7.23, on peut voir un exemple d'un étage d'une texture volumétrique S_i . Il est juxtaposé dans ce dernier les composantes principales selon 2 variables (s_o et t_o) (à chaque étage de la texture volumétrique se trouve une texture à deux dimensions) pour tous les niveaux k de la pyramide Laplacienne.



FIG. 7.23 – Étage d'une texture volumétrique S_i où sont juxtaposées les composantes principales (selon 2 variables) pour tous les niveaux de la pyramide Laplacienne.

De cette manière, l'interpolation linéaire selon les trois dimensions s_i, s_o, t_o est réalisée par la carte graphique. Il ne reste plus qu'à gérer manuellement l'interpolation selon t_i . Ceci peut se faire en accédant deux fois à la texture (pour chaque condition t_l et t_{l+1}) et en réalisant une combinaison linéaire du résultat :

$$\tilde{e}_{kj}(s_i, t_i, s_o, t_o) = \begin{cases} S_0(s_i, s_o, t_o) & \text{si } t_l \leq t_{l_o} \\ S_n(s_i, s_o, t_o) & \text{si } t_l \geq t_{l_n} \\ (1 - \omega)S_{l_i}(s_i, s_o, t_o) + \omega S_{l_{i+1}}(s_i, s_o, t_o) & \text{si } t_{l_i} \leq t_l \leq t_{l_{i+1}} \end{cases}$$

Où $\omega = (t_l - t_{l_i}) / (t_{l_{i+1}} - t_{l_i})$.

En pratique, les textures volumétriques $S_i(s_i, s_o, t_o)$ sont empilées pour former une seule texture volumétrique afin de n'utiliser qu'une seule unité de texture, l'accès aux données se fait alors selon les coordonnées de textures suivantes :

$$\begin{aligned} (s_o, t_o, z_1 = t_{l_i} + \frac{s_i}{n}) \\ (s_o, t_o, z_2 = t_{l_{i+1}} + \frac{s_i}{n}) \end{aligned}$$

Dans ces conditions, le coût de l'interpolation d'une composante principale est donc de deux accès de texture et d'une interpolation linéaire. Afin de faciliter encore cette étape, on utilise une texture cubique (*cube-map*). Dans cette dernière (figure 7.24), on stocke les index pour accéder à la texture volumétrique ainsi que les poids utilisés pour l'interpolation en fonction de la distance entre les angles désirés et ceux présents dans les données de la BTF. L'avantage des textures cubiques est que l'on peut y accéder directement à partir des vecteurs d'éclairage et d'observation calculés lors du rendu.

Taux de compression

Pour effectuer le rendu temps réel, il faut en premier lieu créer des textures à partir des données issues de la compression. Pour cela on applique

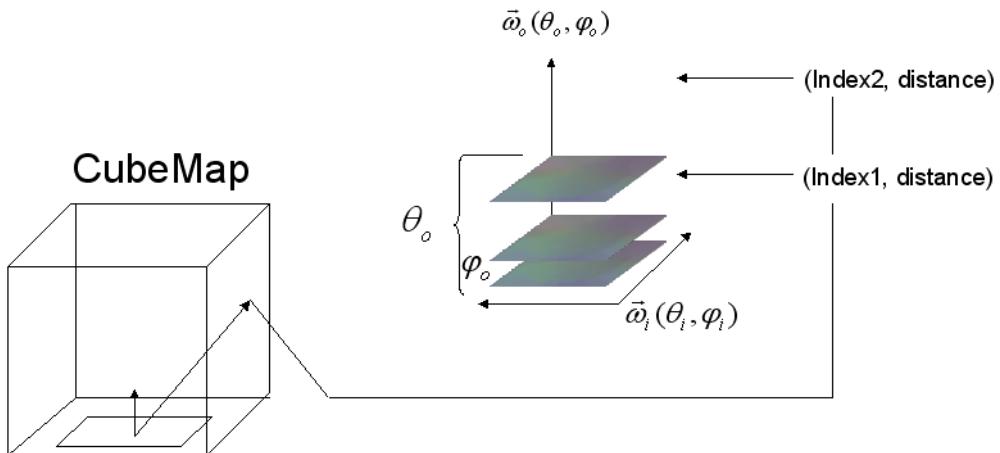


FIG. 7.24 – Accès à la texture volumétrique par *cube-map*.

un facteur d'échelle et de décalage sur les données afin de les normaliser. En effet, les composantes principales peuvent avoir des valeurs négatives et les textures OpenGL ne supportent que des valeurs positives comprises entre 0 et 1. On utilise des textures flottantes sur 32 bits afin de limiter les problèmes de précision lors de nos calculs.

En considérant la configuration suivante :

- Texture de résolution de 128 pixels en valeurs flottantes (4 octets par valeur).
- 6 niveaux de *mip-mapping* (128, 64, 32, 16, 8 et 4).
- 8 composantes principales pour la luminance (Y) et 2 pour chaque composante de chromacité (Cr et Cb), soit 12 composantes principales par niveau de *mip-mapping*.
- Résolution d'un étage de la texture volumétrique des composantes principales égal à 16 pixels étant donné que $16^2 = 256$ est supérieur à la longueur initiale (81 pixels) des ABRDF, ce qui permet de ne pas perdre de données.

Les données stockées et la place mémoire associée sont :

- la texture volumétrique pour les composantes principales utilise $6 * 512 * 16 * 16 * 4 \approx 3,15Mo$ de mémoire.
- la texture volumétrique pour les poids des composantes principales utilise $6 * 12 * 128 * 128 \approx 1,18Mo$ de mémoire.

Sans compter la mémoire utilisée par la texture cubique (ayant une résolution de 256 pixels et commune à toutes les BTF), la place mémoire totale occupée par les données nécessaires à la reconstruction de la BTF (pour un seul matériau) est donc d'environ 4,3 Mega-octets (Mo), ce qui est acceptable pour le temps réel (les cartes graphiques ont aujourd'hui au moins 512

Mo de mémoire). La BTF d'origine pour cette résolution occupe 323 Mo, le taux de compression ainsi obtenu est de 76 :1. Ce dernier ne dépend pas de la complexité de la BTF. Il peut toutefois diminuer si l'on désire utiliser un nombre supérieur de composantes principales pour le rendu.

7.4.2 Algorithme de rendu

La reconstruction de la BTF est entièrement prise en charge par la carte graphique. Les textures sont exploitées par un fragment shader, implémenté en GLSL. Pour chaque pixel de l'écran correspondant à l'affichage du matériau concerné, la BTF est reconstituée en effectuant les étapes suivantes :

1. En premier lieu, on récupère les coordonnées de textures afin d'accéder aux valeurs des poids des composantes principales pour ce pixel. Ensuite, on calcule les angles des vecteurs lumineux et d'observation, à partir desquels on calcule un index permettant d'accéder aux valeurs des composantes principales (on utilise pour cela les espaces TBN locaux définis en chaque sommet de la surface).
2. On calcule la distance d entre le point de la surface au pixel et le point d'observation pour déterminer le niveau de mip-mapping λ (niveau le plus haut de la pyramide à reconstituer). Le niveau de mip-mapping est déterminé selon la méthode de Möller [AMH02] afin d'obtenir une transition continue entre les niveaux compris entre 0 et λ_{\max} : $\lambda = \min(\lambda_{\max}, \max(\log_2(d), 0))$.
3. On récupère les valeurs des composantes principales et leurs poids depuis les textures selon les angles, les coordonnées de texture pour chaque niveau de mip-mapping k . On compose alors pour chaque niveau k des vecteurs dont la taille est définie par le nombre de composantes principales n que l'on souhaite prendre en compte pour le rendu. Les vecteurs D_k contiennent les poids d'un niveau k , où $D_k(j) = d_{kj}(p)$ avec $1 \leq j \leq n$ et les vecteurs E_k contiennent les composantes principales d'un niveau k .
4. On reconstitue la valeur du pixel en réalisant un mélange successif des niveaux de la BTF à partir des vecteurs créés à l'étape précédente et en ajoutant l'éclairage moyen du matériau :

$$\left(\sum_{k=0}^{\lambda} p_k D_k^T E_k \right) + \mu'_{\lambda_{\max}}(\omega_i, \omega_o)$$

Où $p_k = \min(\max(1 - \lambda + k, 0), 1)$ est le poids caractérisant la contribution du niveau k dans le résultat final. Ce poids permet d'avoir une

transition continue entre les niveaux de mip-mapping, évitant ainsi des artefacts visuels de transition brusque.

5. A la dernière étape, on retransforme l'espace de couleur YCrCb vers RVB puis on affecte la valeur du pixel dans le *frame-buffer* de la carte graphique.

7.4.3 Détails d'implémentation

Pour calculer les variables permettant d'accéder à la BTF, il est nécessaire de définir des espaces locaux aux sommets de la géométrie (TBN : Tangent, Binormal, Normal). Ils permettent de calculer les angles des vecteurs lumineux et d'observation. Ces espaces ne sont généralement pas présents dans les graphes de scène 3D classiques qui ne fournissent que la normale à la surface et les coordonnées de texture. Il est donc nécessaire de les calculer. La principale contrainte réside en la continuité entre les espaces TBN de deux sommets voisins : à partir de la normale à la surface, il est possible de définir une infinité de vecteurs tangents dans le plan perpendiculaire à la normale. Si l'on veut qu'il n'y ait pas de sauts dans la reconstitution de la BTF, il faut que les vecteurs calculés en chaque sommet soient continus (cohérence dans la progression des angles). Une méthode pour calculer un espace TBN consiste à tenir compte des coordonnées de textures en chaque sommet qui fournit une information de direction entre les sommets. Certaines API de graphe de scène (comme OpenSceneGraph) prennent en charge le calcul de l'espace TBN.

7.5 Résultats

Nous allons maintenant présenter les résultats en montrant des exemples de rendu de BTF en temps réel pour différentes conditions d'éclairage et d'observation. Nous présentons également les résultats en terme de performances (fréquence image) et discutons de la qualité d'image (taux d'erreur introduit par la compression).

7.5.1 Résultats en image

La figure 7.25 montre une image calculée par l'algorithme présenté ci-dessus à partir d'un échantillon de BTF mesurée (échantillon "Impalla" de l'Université de Bonn). On peut y remarquer que la réflexion spéculaire est préservée. Les ombres et les réflexions sur les bords des carreaux sont restitués et donnent une bonne impression de relief. Rappelons que seul un plan est dessiné, qu'il n'y a aucune information géométrique, l'impression de relief provient des textures de la BTF qui correspondent en chaque pixel à la

condition d'éclairage et d'observation correspondante (avec une interpolation linéaire pour les conditions n'appartenant pas à la BTF source).

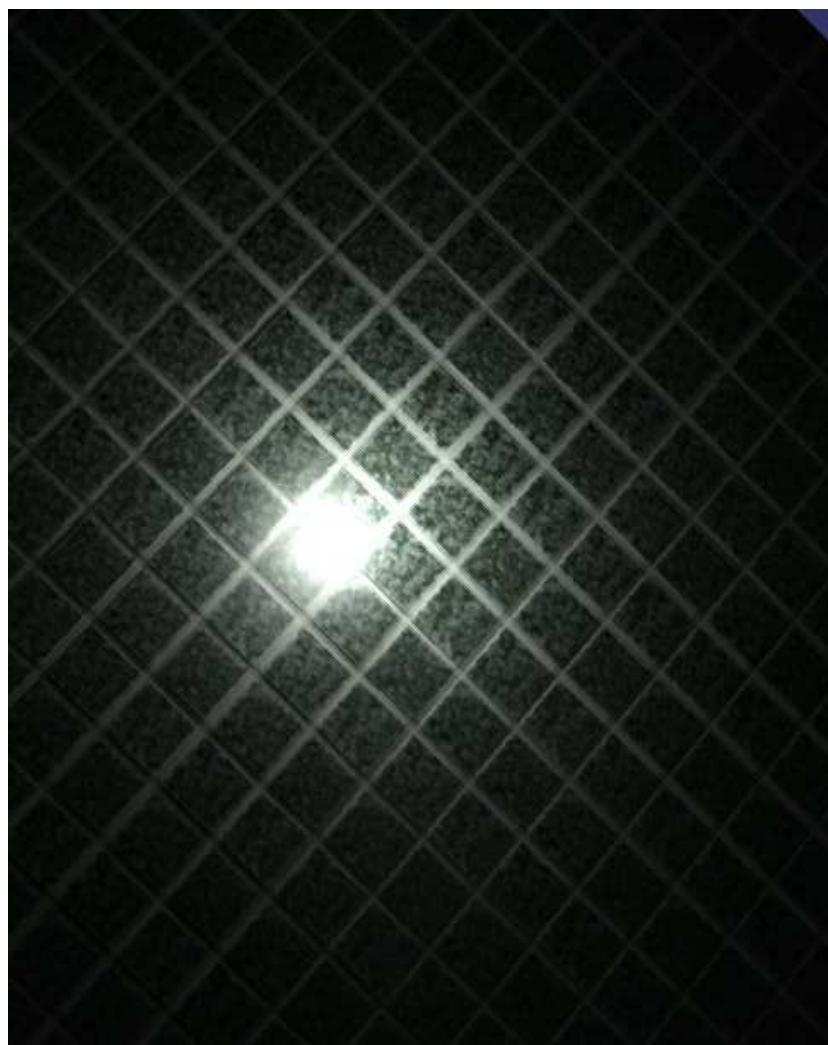


FIG. 7.25 – Rendu temps réel de BTF mesurée (échantillon "Impalla" mesuré par l'Université de Bonn).

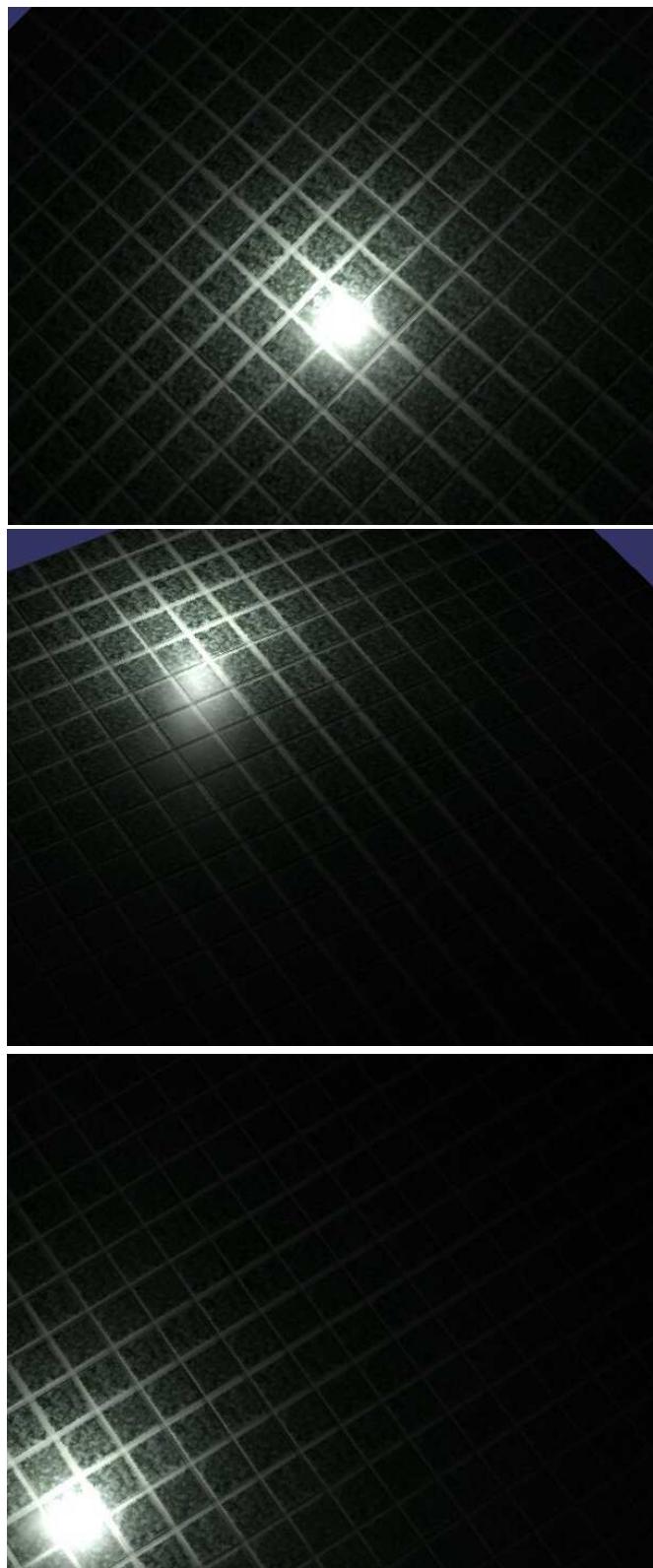


FIG. 7.26 – Rendu temps réel de BTF mesurée sous plusieurs conditions (échantillon "Impalla").

Les figures 7.27 et 7.28 présentent respectivement le rendu temps réel de BTF synthétique de pavés pour plusieurs conditions de vue et d'éclairage. Comme pour le matériau mesuré précédent, la réflexion spéculaire est préservée (ce qui est important étant donné que ce phénomène est beaucoup plus fin (haute fréquence) que la composante diffuse et risque d'être plus fortement affectée par la compression). L'impression de relief est également satisfaisante, même pour des angles d'observation rasants (rappelons que la technique de bump-mapping donnent des résultats médiocres dans ces conditions). Ce résultat est obtenu grâce à la technique de BTF qui contient l'auto-masquage de la surface qui évite une sensation d'aplatissement du matériau.

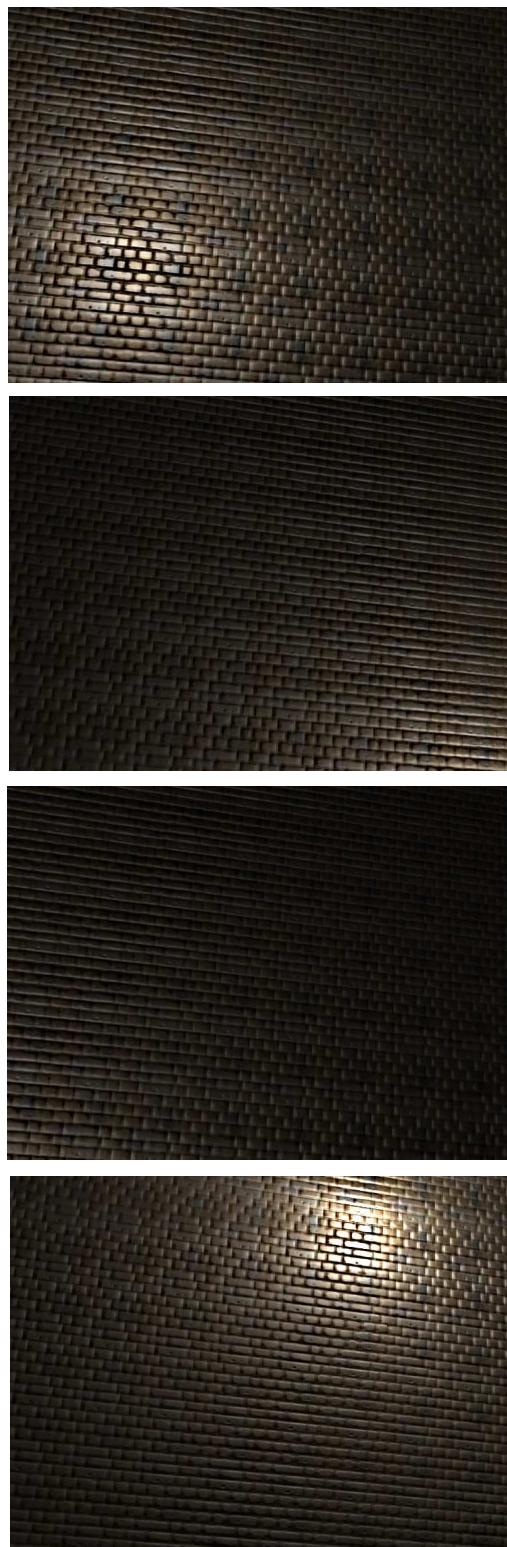


FIG. 7.27 – Rendu temps réel de BTF synthétique de pavés sous plusieurs conditions d'éclairage

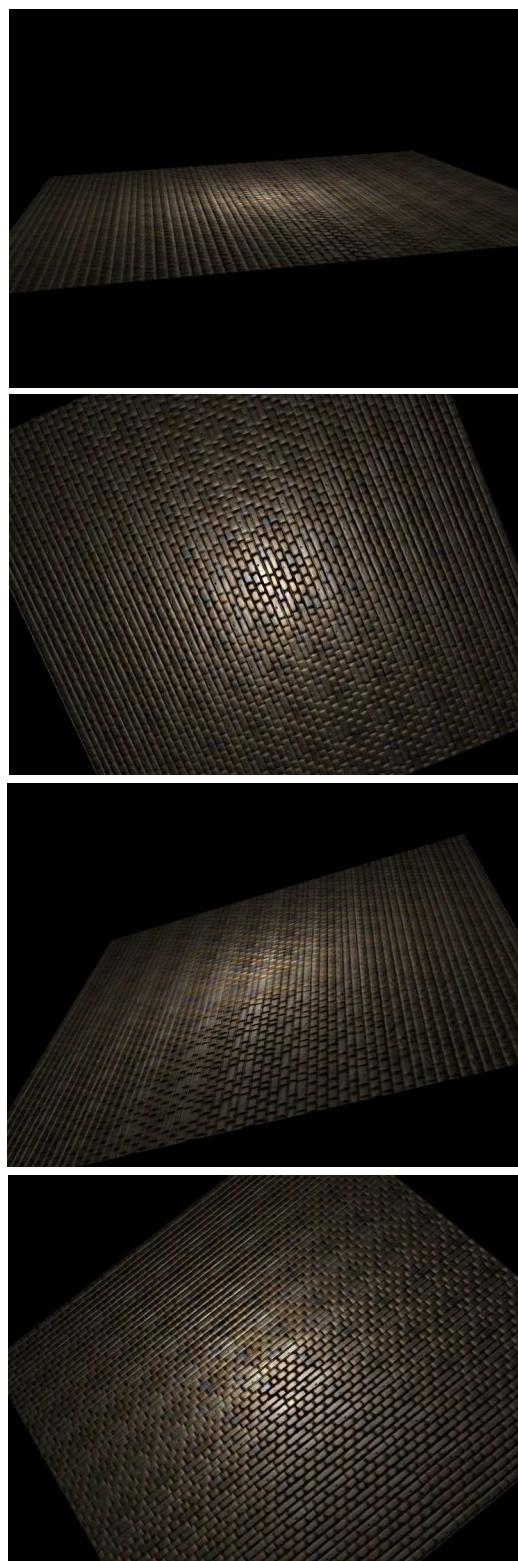


FIG. 7.28 – Rendu temps réel de BTF synthétique de pavés sous plusieurs conditions de vue

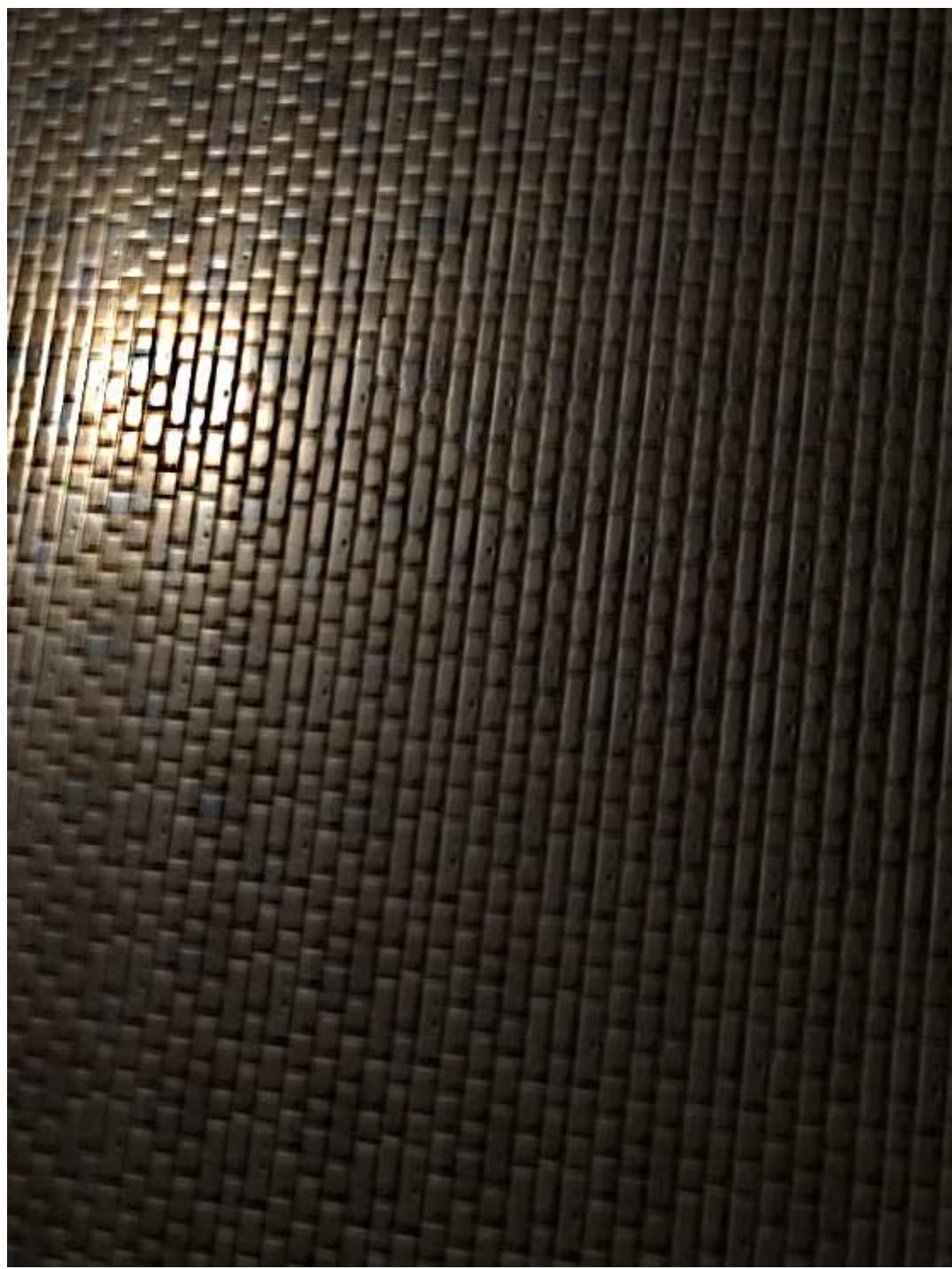


FIG. 7.29 – Rendu temps réel de BTF synthétique de pavés en vue rapprochée



FIG. 7.30 – Rendu temps réel de BTF de velours (échantillon "Corduroy" mesuré par l'Université de Bonn)



FIG. 7.31 – Rendu temps réel de BTF de velours (2)

7.5.2 Performances

Nous avons implémenté un prototype visuel pour le rendu temps réel de BTF. Il est basé sur l'API open source OpenSceneGraph. La machine cible est un PC dont la configuration est la suivante :

- Système d'exploitation Linux (Debian).
- Processeur : Intel Xeon 2,6 Ghz.
- Carte graphique : Nvidia Quadro FX 4400.
- Écran LCD (1280x1024) à 60 Hz.

Sur cette configuration, les fréquences obtenues oscillent entre 16 et 110 Hz en fonction du nombre de pixels à calculer. La majorité des calculs nécessaires à la reconstruction de la BTF se faisant au niveau du pixel shader, les fréquences obtenues sont très dépendantes du nombre de pixel représentant des surfaces modélisées par BTF. La surface de route ne couvre généralement qu'une proportion faible de l'écran (environ 30%), le nombre de pixel à calculer reste donc raisonnable : en résolution 1280x1024, la fréquence image obtenue est de l'ordre de 72 Hz (bornée à la résolution de l'écran qui est de 60 Hz). Les résultats en terme de fréquence sont acceptables pour une simulation temps réel, d'autant plus que la dernière génération de carte graphique (Nvidia GeForce 8800 GTX) est environ trois fois plus performante [Nvi07] en terme de calculs GLSL que nous utilisons de manière intensive pour la reconstruction de la BTF (taux de remplissage des pixels de 36,8 milliards par seconde pour cette carte contre 13,2 pour celle que nous utilisons) .

Il est possible d'optimiser le compromis entre la qualité et la fréquence image obtenue car la qualité est modifiable en temps réel lors du rendu, en paramétrant le nombre de composantes principales utilisées pour la reconstruction de la BTF.

7.6 Validation

Comme nous l'avons déjà exposé, l'algorithme présenté permettant d'aboutir à une représentation compacte de la BTF engendre une perte d'informations sur les textures de la BTF dans la mesure où le nombre de composantes principales utilisées pour les reconstruire est réduit. Afin d'estimer la qualité de la BTF recomposée, nous allons comparer visuellement et par calcul la différence entre la BTF d'origine et la BTF recomposée à partir des données issues de l'algorithme de compression.

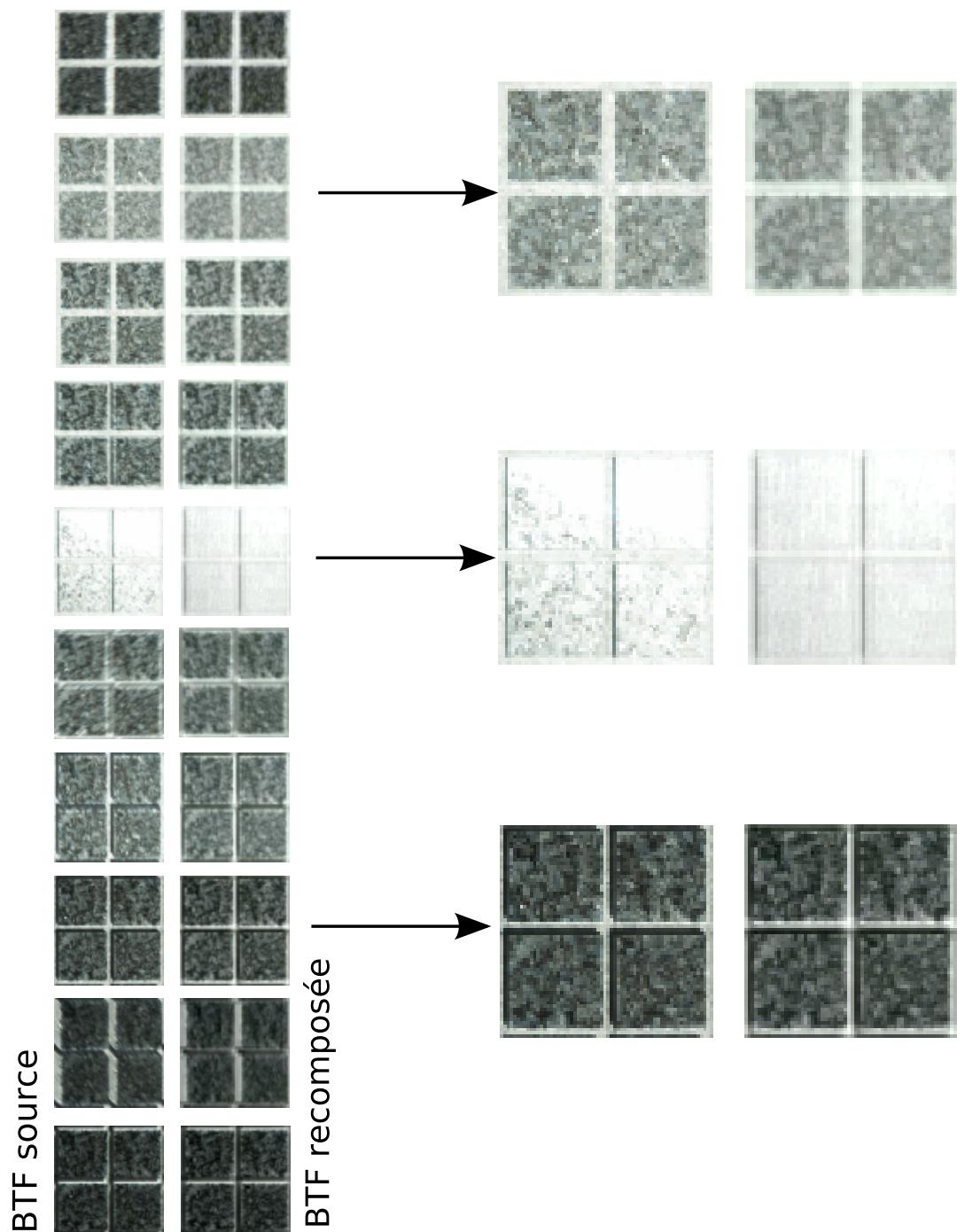


FIG. 7.32 – Comparaison entre la BTF source (colonne de gauche) et la BTF recomposée (colonne de droite) avec 8 composantes principales

Sur la figure 7.32, la comparaison visuelle de la BTF d'origine et de la BTF recomposée à partir de 8 composantes principales montrent qu'il est possible de reconstruire les textures de la BTF tout en préservant les différences entre les échantillons pour différentes conditions d'éclairage et d'observation qui sont caractéristiques de la BTF. Pour certaines d'entre-elles, il est même difficile de percevoir une différence visuelle entre les textures sources et recomposées. Bien que l'aspect général de la BTF semble être préservé, on peut toutefois remarquer que les textures sont un peu "lissées", c'est à dire que les détails de plus hautes fréquences spatiales sont perdus. Ces détails auraient été restitués par les composantes principales que nous avons éliminer lors de la reconstruction. Il convient de mesurer mathématiquement le taux d'erreur global sur toute la BTF (la figure 7.32 ne montre que quelques échantillons) afin de quantifier la perte exacte engendrée par notre méthode. La comparaison visuelle est elle aussi importante car il n'est pas toujours facile d'imaginer comment se traduit visuellement un taux d'erreur calculé.

Pour estimer la perte de données due à la compression, nous avons choisi de calculer l'erreur RMSE relative ("Root Mean Square Error"). Cette méthode consiste à calculer, pour chaque texture de la BTF d'origine $F^{\omega_i, \omega_o}(p)$, la moyenne des différences au carré des pixels avec la texture recomposée après compression $\tilde{F}^{\omega_i, \omega_o}(p)$ divisé par la valeur des pixels de la texture source $F^{\omega_i, \omega_o}(p)$ afin d'obtenir un pourcentage qui est plus intuitif que le taux d'erreur MSE classique. L'erreur RMSE relative pour n échantillons se calcule de la façon suivante :

$$RMSE = \sqrt{\frac{1}{n} \sum_{j=1}^n \left(\frac{\tilde{F}^{\omega_i, \omega_o}(p_j) - F^{\omega_i, \omega_o}(p_j)}{F^{\omega_i, \omega_o}(p_j)} \right)^2}$$

Nous avons calculé cette valeur pour chaque texture $F^{\omega_i, \omega_o}(p)$ pour plusieurs reconstructions pour lesquelles on modifie le nombre de composantes principales considérées. Sur la figure 7.33, on peut voir l'erreur RMSE relative moyenne sur toute la BTF avec une reconstruction utilisant 8 composantes principale ainsi que l'erreur RMSE pour chaque texture de la BTF. L'erreur moyenne relative obtenue est de 14%.

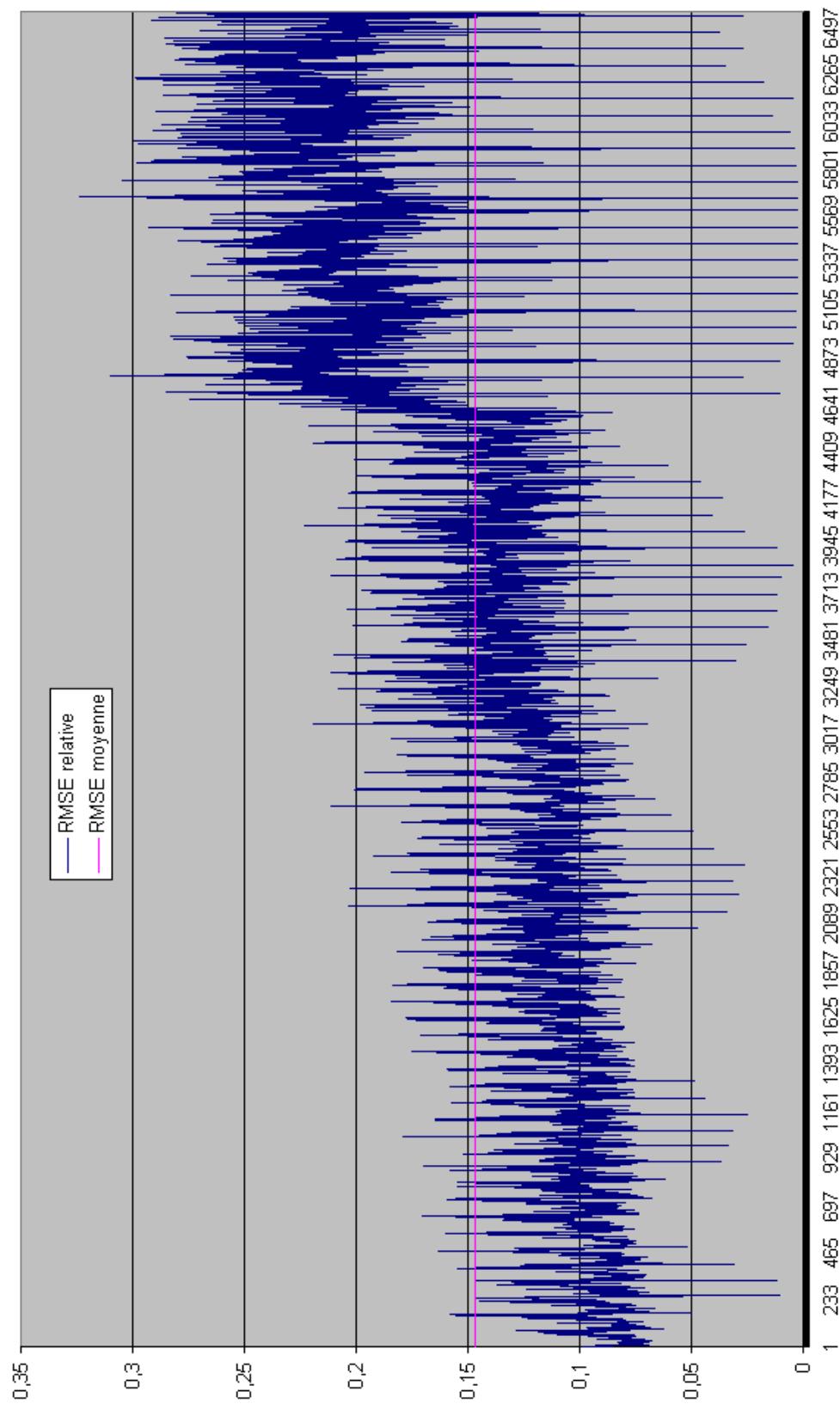


FIG. 7.33 – Erreur relative RMSE pour chaque texture de la BTF avec 8 composantes principales

Sur la figure 7.34, l'erreur donnée correspond à la reconstruction en utilisant 4 composantes principales. L'erreur moyenne obtenue est de 20%.

On remarque que l'erreur va en croissant, ce qui traduit que l'erreur est d'autant plus grande que l'angle d'incidence de la direction d'éclairage est petit. Ceci peut être compréhensible par le fait que plus la lumière est rasante, plus il y a de discontinuités dans l'éclairage de la surface, comme pour les ombres en particulier. De plus, les textures sont en moyenne plus sombres pour la lumière rasante, ainsi, le taux d'erreur relatif par rapport aux valeurs des pixels initiaux devient plus important. Les variations du taux d'erreur suggèrent qu'afin d'optimiser le temps de calcul nécessaire au rendu, il serait possible de choisir dynamiquement le nombre de composantes principales utilisées pour la reconstruction en fonction de l'angle d'observation et d'éclairage étant donné que ce taux varie selon ces variables.

Sur les figures 7.33 et 7.34, on remarque également que la variance est importante : toutes les 81 mesures, l'angle de vue devient rasant introduisant des discontinuités dans la BTF dues à l'auto-masquage de la surface ayant une géométrie non plane.

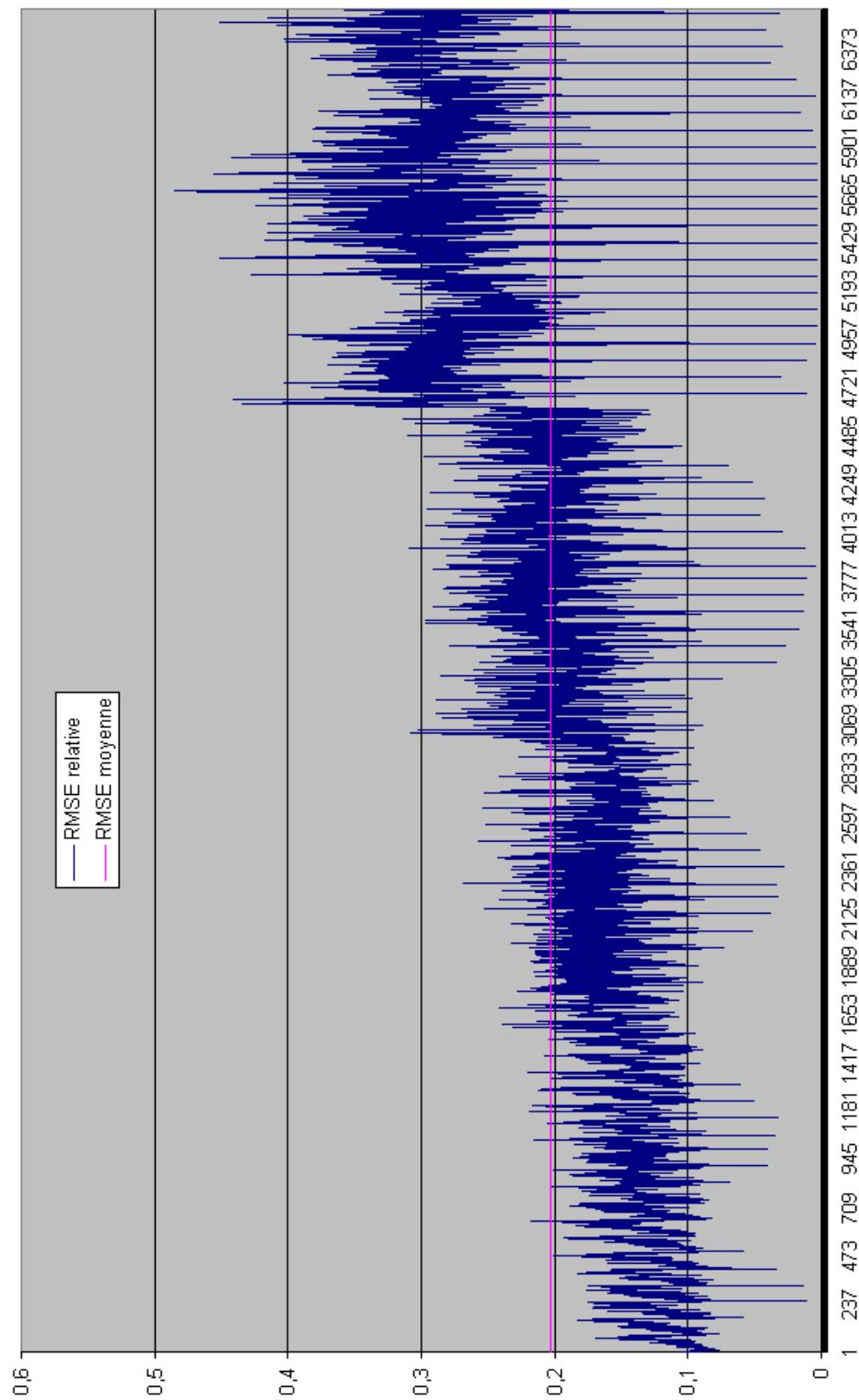


FIG. 7.34 – Erreur relative RMSE pour chaque texture de la BTF avec 4 composantes principales

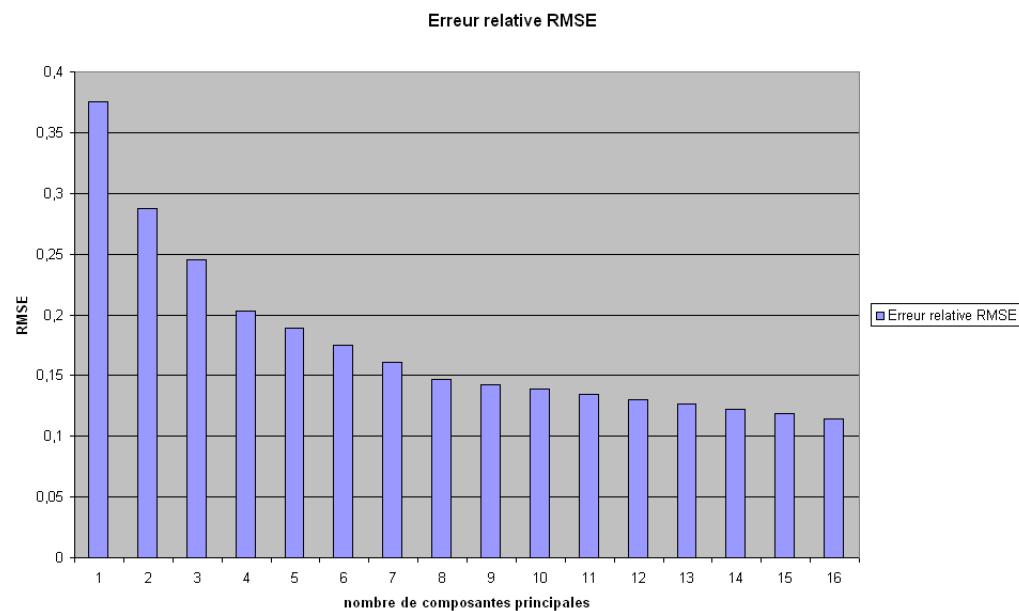


FIG. 7.35 – Erreur relative RMSE en fonction du nombre de composantes principales utilisée pour la reconstruction de la BTF

Afin de valider le choix du nombre de composantes principales à considérer pour le rendu, les taux d'erreur RMSE relatifs sont représentés sur la figure 7.35. On remarque que le taux d'erreur décroît rapidement au début (le taux d'erreur passe de 37% à 14% entre 1 et 8 composantes principales), puis le taux d'erreur décroît lentement pour une reconstruction entre 8 et 16 composantes principales, pour terminer à un peu plus de 10%. On peut donc considérer que le meilleur compromis entre le taux d'erreur et le nombre de composantes principales se situe entre 4 et 8 en fonction de la complexité de phénomènes contenues dans la BTF.

Chapitre 8

Visualisation de surfaces hétérogènes en mouvement

Le rendu réaliste de surfaces par des algorithmes de synthèse d'image avancés tels que le bump-mapping ou encore les BTF permet d'augmenter la richesse des images de part le grand nombre de détails simulés sur les matériaux hétérogènes, améliorant ainsi le réalisme visuel des images calculées en temps réel. Dans le cas de la simulation de conduite, l'observateur (généralement le conducteur) est principalement en mouvement. L'affichage dynamique d'images réalistes (qui comportent un grand nombre de détails) engendre des artefacts visuels (bruit) [LT06] lorsque la fréquence image n'est pas suffisante et que l'observateur où les objets ont un mouvement important. Ces problèmes montrent qu'il est nécessaire de tenir compte des propriétés du système de génération d'images (résolution , fréquence, ...) au delà des modèles utilisés pour la simulation réaliste de surfaces hétérogènes. Dans ce chapitre, nous expliquons l'origine des artefacts visuels générés, en particulier le lien entre la résolution spatiale et temporelle. Nous proposons une méthode de filtrage d'anti-aliasing temporel pour tenter de palier le problème et présentons les résultats obtenus.

8.1 Artefacts visuels

8.1.1 Présentation du problème

La simulation de surfaces hétérogènes par des méthodes de BTF se traduit par un grand nombre de détails ainsi qu'un contraste élevé dans les images. Si la simulation d'éclairage de tels matériaux ne pose pas de problèmes particuliers (hormis le filtrage nécessaire pour l'anti-aliasing spatial) lorsque l'observateur est fixe ou se déplace très lentement par rapport aux surfaces, il en est tout autre dans le cadre de la simulation de conduite où l'observateur (conducteur) est en mouvement.

Pour illustrer les problèmes rencontrés, prenons un exemple : pour une fréquence d'image donnée (60 Hz), et un véhicule virtuel se déplaçant à une vitesse constante de 130 km/h, la distance parcourue par l'observateur entre deux images successives affichées est de 60 cm (1,2 m si la fréquence image est de 30Hz). En prenant un revêtement routier (asphalt) dont le grain a une dimension de l'ordre de 5mm, et si l'on considère un pixel particulier affichant un point sur cette surface de route, il n'y a aucune corrélation entre les points affichées en ce pixel pour les deux images successives affichées. En réalité, 120 points distincts de la surface sont virtuellement "passés" par le pixel entre les deux images mais n'ont pas été affichés. Cette perte d'informations se traduit par l'affichage de points distincts à une fréquence donnée qui est perçue comme du bruit et non pas comme un déplacement par l'observateur.

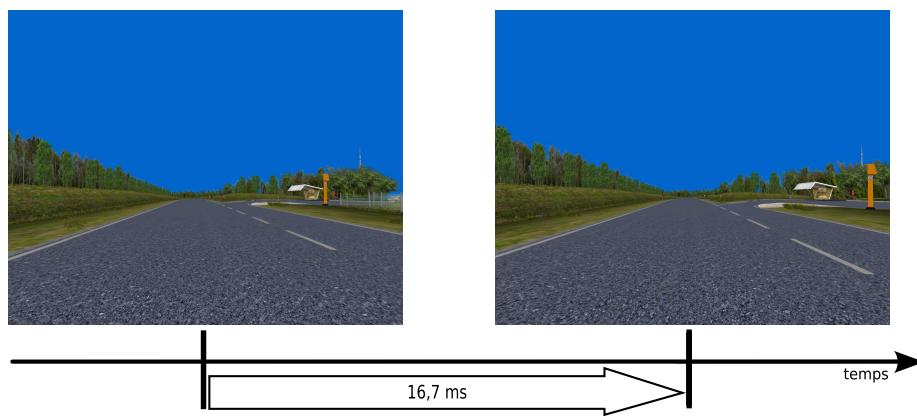


FIG. 8.1 – Temps écoulé entre deux frames

Plus généralement, pour les images affichées à chaque pas de temps, la texture (ou BTF) de la route est différente et non corrélée à cause du déplacement de l'observateur, générant en quelque sorte un bruit visuel rendant

la simulation fortement non réaliste. Cet artefact ne peut être donné en exemple par des images statiques car il n'est visible qu'en situation dynamique où l'observateur est en mouvement. Ceci montre que pour effectuer une simulation réaliste, la seule représentation de matériaux hétérogènes par des techniques de synthèse d'image spécifiques (BTF) n'est pas suffisante. Il est également nécessaire de tenir compte des limitations du système de génération d'image (carte graphique, système d'affichage) ainsi que des caractéristiques de la perception visuelle humaine pour aboutir à un résultat satisfaisant.

Après avoir présenté les artefacts visuels, nous allons faire une analyse plus approfondie du problème afin de proposer une solution visant à simuler de manière satisfaisante des surfaces hétérogènes lorsque l'observateur est en mouvement afin de garantir le réalisme visuel dans ces conditions.

8.1.2 Analyse du problème

Intuitivement, on imagine bien que le problème provient des différents échantillonnages : les artefacts visuels dépendent de la fréquence image (échantillonnage temporel) et de la résolution des détails présents sur la surface (échantillonnage spatial). En effet, plus la fréquence image est faible, plus la distance parcourue entre deux images est grande (à vitesse constante), et plus la surface est détaillée, plus le nombre de données non affichées (perte d'information) sera grand. Aussi, plus la vitesse du conducteur augmente, plus la distance parcourue entre deux images est grande, et ainsi plus les artefacts visuels sont importants. Si l'observateur est statique, aucun artefact visuel ne perturbe la simulation.

L'analyse de ce problème fait intervenir des questions sur le rapport entre la fréquence image et la fluidité perçue de la simulation. Il y a très souvent une grande confusion entre les questions posées à ce sujet, en particulier : "Combien d'images par seconde l'oeil humain peut-il percevoir ?" n'est pas équivalent à "Quelle fréquence image est nécessaire pour qu'une simulation paraisse fluide ?".

En outre, si l'observateur visualise du brouillard diffus se déplacement très lentement sans bordure nette, la simulation sera perçue fluide pour une fréquence image faible (10 images par seconde) étant donné que le déplacement entre deux images est très petit. Le cas extrême consiste à visualiser un mur fixe, dans ce cas, une seule image par seconde est suffisante. Ceci montre que la réponse à la question du rapport entre la fréquence image et la perception de la fluidité n'est pas absolue mais est dépendante de l'animation visualisée (fréquence spatiale de l'image, contraste, vitesse de déplacement des objets, etc.).

Une autre question concerne les différences entre l'industrie cinématographique et le domaine du jeu vidéo : pourquoi un film visualisé à une fréquence de 24 images par seconde paraît fluide alors d'un jeu vidéo semble saccadé à cette même fréquence ?



FIG. 8.2 – Exemple de flou de déplacement (*motionblur*) dans une image capturée d'un film

Ce phénomène résulte d'une différence de méthode entre la production d'images de l'industrie cinématographique (figure 8.2) et du monde de la simulation temps réel. Une caméra vidéo acquiert chaque image durant un certains temps, ainsi une intégration des déplacements est effectuée. Chaque image représentant une scène où il y a du mouvement (des objets de la scène) comporte donc du flou traduisant le déplacement effectué par les objets en mouvement pendant le temps d'acquisition de l'image. Au contraire, dans la synthèse d'image temps réel, chaque image affichée correspond à l'état de la scène à un instant t précis, et ne contient pas d'informations sur les déplacements des objets depuis l'image précédente au temps $t - 1$. La différence d'informations de déplacement (souvent nommée "motion blur") fait qu'il est nécessaire d'avoir une fréquence image supérieure pour la simulation temps réel que pour un film vidéo pour que la fluidité perçue soit satisfaisante.

Le problème concerne donc la représentation d'un signal continu par un système numérique discret (spatial et temporel). Le théorème de Shannon-Nyquist [Sha49] stipule que la fréquence d'échantillonnage d'un signal doit être égale ou supérieure au double de la fréquence maximale contenue dans ce signal, afin de convertir sans pertes ce signal d'une forme analogique à

une forme numérique.

Pour un signal continu dans le temps $x(t)$, la transformée de Fourier $X(f)$ continue s'écrit :

$$X(f) = \int_{-\infty}^{+\infty} x(t)e^{-2\pi ift}dt$$

Le signal $x(t)$ a une bande passante B soit $X(f) = 0$ pour $|f| > B$. La condition pour une reconstruction exacte du signal $x(t)$ depuis les données échantillonnées à une fréquence f_s est $f_s > 2B$. La fréquence limite $f_s/2$ est appelée "fréquence de Nyquist".

Dans le cas où cette condition n'est pas respectée, on parle de repliement spectral ou plus couramment d'aliasing (crénelage).

Dans notre cas, la fréquence d'affichage est très inférieure à la fréquence du signal (variations des détails de la surface de route), le système d'affichage ne respecte donc pas la fréquence de Nyquist et engendre ainsi des problèmes d'aliasing.

Il existe deux solutions potentielles pour palier le problème. La première consisterait à augmenter de manière très importante la fréquence d'affichage pour restituer de manière exacte le signal mais ceci n'est pas réalisable tant en terme de puissance de calcul pour la génération d'images que de capacité matérielle (les systèmes les plus évolués ont aujourd'hui une fréquence d'affichage d'environ 120Hz). La seconde méthode consiste à appliquer un filtrage afin de réduire la fréquence du signal (filtre passe-bas).

Nous allons maintenant proposer une méthode de filtrage temps réel permettant de diminuer ces problèmes d'aliasing. Il est important de noter que nous n'essayons pas de restituer ce que l'oeil humain est capable de percevoir mais d'optimiser l'affichage d'un signal haute fréquence sur un système d'affichage connaissant ses limitations intrinsèques.

8.2 Filtrage anti-aliasing temporel en temps réel

Le filtrage anti-aliasing temporel consiste à intégrer les données affichées entre deux images successives afin de ne pas perdre de données et d'obtenir un affichage continu. La solution la plus intuitive et naturelle consiste à calculer n images supplémentaires et à intégrer le résultat entre deux images successives affichées. Bien entendu, cette méthode n'est pas applicable en temps réel car elle diviserait la fréquence image par n , ce qui n'est pas concevable. Elle peut toutefois être utilisée, comme l'on fait Brostow et al. [BE01] pour créer une animation à partir d'images prisent à intervalle

de temps régulier (technique connue sous le nom de *stop motion*). Il faut alors trouver une méthode pour calculer un filtre passe-bas sans calculer n fois plus d'images. Nous allons maintenant proposer un algorithme de rendu remplissant cet objectif.

8.2.1 Algorithme de rendu

L'algorithme appartient au domaine du "*post-processing*". Il consiste à appliquer un filtre après avoir calculé l'image normalement. L'algorithme est constitué des trois étapes suivantes :

1. Génération de l'image à l'instant t puis sauvegarde du buffer image dans une texture T .
2. Calcul des vecteurs vitesses projetés dans le plan de la caméra pour chaque pixel (nécessite de connaître la position de la caméra aux temps t et $t - 1$).
3. Rendu final en combinant les texels de la texture sauvegardée à l'étape 1, le long des vecteurs vitesse.

Parmi les trois étapes de cet algorithme, les deux dernières peuvent être réalisées dans la même passe de rendu. Cet algorithme requiert au total deux passes de rendu, c'est à dire qu'il est nécessaire de dessiner deux fois la même scène pour chaque image affichée. Nous prenons en compte le déplacement entre le temps t et le temps $t - 1$. D'autres solutions ([DK00]) proposent de tenir compte également de l'image au temps $t + 1$ afin que le filtrage soit "centré" sur le temps courant. Ceci pourrait être réalisé mais obligerait à extrapoler la trajectoire du véhicule.

8.2.2 Détails de l'algorithme de rendu

Calcul des vecteurs vitesses

Les vecteurs vitesses $d\vec{P}$ sont calculés en 3D puis projetés dans le plan de la caméra en fonction de la géométrie de la scène et de la position de la caméra aux temps t et $t - 1$:

$$d\vec{P} = M_{projection}.(M_{camera}(t).P_{sommet} - M_{camera}(t - 1).P_{sommet})$$

Sur la figure 8.3 est représentée la norme des vecteurs vitesse $\|d\vec{P}\|$ en couleur (en rouge pour la norme selon l'axe x et en vert pour la norme selon l'axe y). Plus on est dans l'axe du déplacement (centre de l'écran), plus la vitesse est longitudinale (vert), plus on se trouve sur les bords inférieurs de



FIG. 8.3 – Représentation en couleur des vecteurs vitesses

l'image, plus la vitesse devient latérale (rouge).

En fonction de la norme des vecteurs vitesse, on indique si la zone doit être filtrée ou non (zone représentée en blanc sur la figure 8.4) par un seuil permettant ainsi d'optimiser le nombre de calculs. En effet, pour des zones lointaines, le déplacement entre deux images successives est peu important et ne nécessite pas de filtrage. De plus, les zones éloignées subissent un filtrage d'anti-aliasing spatial important et ne génère donc pas d'artefacts visuels.



FIG. 8.4 – Optimisation des zones filtrées en fonction des vecteurs vitesses

Nombre d'échantillons

Les texels de la texture T sont sélectionnés le long des vecteurs vitesse. Afin d'optimiser à la fois la qualité et la performance de l'algorithme, il est judicieux de définir le nombre d'échantillons en fonction de la norme du vecteur vitesse. Sur la figure 8.5 est représenté le nombre d'échantillons pris en compte dans le calcul du filtre. Le nombre maximal d'échantillons pris en compte se situe entre 16 et 24 pour des raisons de performances.



FIG. 8.5 – Représentation du nombre d'échantillons en fonction de la vitesse

Pondération des texels

Chaque texel pris en compte le long du vecteur vitesse doit être pondéré par une fonction. En théorie, chaque texel a un poids équivalent dans la mesure où il est apparu le même temps que les autres pour un véhicule se déplaçant à vitesse constante. Si le véhicule subit une phase d'accélération entre deux images, il serait plus juste de pondérer chaque texel par son temps réel d'apparition bien que ce phénomène soit très mineur.

Problèmes de rendu

Cette méthode ne fonctionne pas dans les zones en bordures de l'image (zone rouge sur la figure 8.6). En effet, pour ces zones les vecteurs vitesses

sortent de l'image. Les données de l'image au temps $t - 1$ sont manquantes pour calculer le filtre au temps t . La méthode d'implémentation palliant ce problème, consiste à inverser le vecteur vitesse pour ces zones et ainsi de pouvoir utiliser les données de la texture T . Cette astuce se justifie en considérant qu'il n'y ait pas de changement majeur de direction entre deux images successives affichées et que les points de la route sont suffisamment nombreux et aléatoires pour que leur moyenne soit relativement équivalente quelque soit la direction de déplacement. La potentielle différence de rendu pour ces zones de l'image générée n'est pas perceptible.

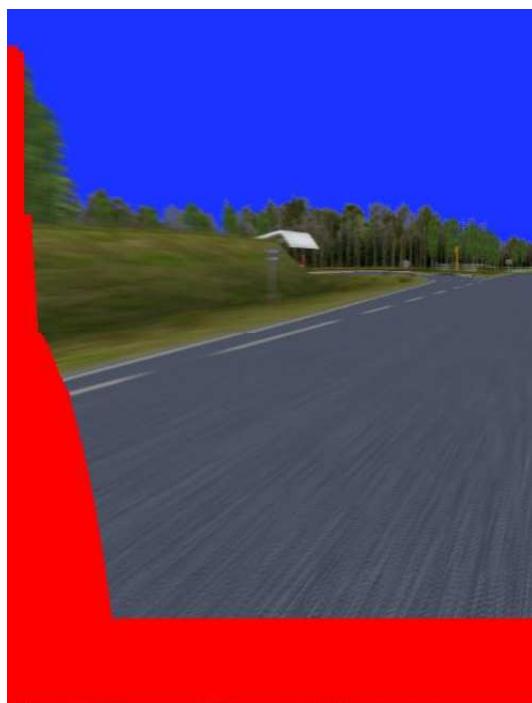


FIG. 8.6 – Filtrage dans les zones en bordure de l'image

8.3 Résultats

8.3.1 Résultats en image

Sur la figure 8.7, on peut apprécier visuellement la comparaison (statische) de la surface de route avec une partie filtrée et une partie non filtrée. La partie filtrée montre le résultat du filtre passe-bas qui génère des traits flous le long des vecteurs de vitesse. On peut remarquer que même pour une image statique, une information de vitesse est fournie par le filtrage.



FIG. 8.7 – Comparaison avec et sans filtrage

L'effet du filtrage est encore plus évident pour une vue de côté où le déplacement entre deux images successives est plus important, ce qui est appréciable pour les simulateurs de conduite immersif à large champ de vision où les artefacts visuels étaient très problématiques en vision périphérique.

L'algorithme permet d'utiliser des textures (ou BTF) de plus haute résolution (figure 8.9). En effet, les textures sont mises au point par les infographistes pour éviter les problèmes d'aliasing (spatial et temporel). En exploitant le filtrage, des textures plus détaillées peuvent être utilisées sans générer d'artefacts visuels lors de la visualisation en mouvement.

8.3.2 Performances

La fréquence image obtenue pour la configuration suivante varie entre 30 Hz et 60 Hz selon le nombre de pixels à traiter (la surface de la route varie en fonction des conditions (ligne droite, virage, hauteur de l'observateur par rapport à la route, etc.) :

- Système d'exploitation Linux (debian).
- Processeur : intel Xeon 2,6 Ghz.
- Carte graphique : Nvidia Quadro FX 4400.
- Ecran LCD (1280x1024) à 60 Hz.



FIG. 8.8 – Comparaison avec et sans filtrage



FIG. 8.9 – Filtrage de texture de route très détaillée (4 fois supérieure à celle utilisée initialement pour cette base de donnée)

8.3.3 Limitations

La méthode présentée n'est pas une réelle intégration 3D du déplacement entre deux images. Il est plus correct de parler d'algorithme "2.5D". Ainsi, si un objet apparaît entre deux images, il ne sera pas pris en compte, ce qui peut entraîner des problèmes d'occlusion.

Le déplacement utilisé pour notre intégration est linéaire, hors, si le véhicule a une trajectoire courbe, il serait plus convenable d'intégrer le long du déplacement réel du véhicule. Fort heureusement, la fréquence image et la vitesse du véhicule font que ces limitations n'engendrent que de manière très exceptionnelle des artefacts visuels, de plus, ceux-ci sont très peu perceptibles car très courts temporellement.

L'algorithme est basé sur la projection de l'image à l'écran et calcule ainsi le filtre par rapport à l'orientation de la pyramide de vision. Ceci a pour effet que le bas coté de la route sera perçu de manière très floue si l'observateur déplace son regard sur le bord de l'écran. Il serait judicieux d'intégrer un capteur de regard et de tenir compte de la direction d'observation réelle pour calculer le filtre en concordance avec les mouvements du regard de l'observateur réel.

8.4 Conclusion

Nous avons mis en place une méthode de filtrage visant à éliminer les problèmes d'aliasing temporel pouvant intervenir lors de simulation pour laquelle la scène visuelle contient beaucoup de détails et où l'observateur est en mouvement. Cet algorithme a été intégré dans le logiciel de simulation de conduite SCANeR II développé par Renault. Même si la méthode présente quelques limitations, elle est bien adaptée à la surface de route et donne des résultats visuels satisfaisant. Ironiquement, plus la fréquence image est affectée par le temps nécessaire au calcul du filtrage, plus les bénéfices du filtrage temporel sont appréciables. Nous avons également vu qu'en plus d'éliminer les problèmes d'artefacts visuels, cette méthode fournit des indices de vitesses souvent manquants en simulation de conduite. Toutefois, des études de validation seraient nécessaires afin de paramétriser le modèle plus finement (choix des surfaces filtrées, pondération du filtrage, etc.) mais aussi afin d'évaluer l'éventuel impact des indices visuels fournis sur la vitesse et la trajectoire du conducteur en simulation de conduite.



FIG. 8.10 – Filtrage complet de la texture de route

Chapitre 9

Discussion générale

Après avoir présenté nos différents travaux, nous allons dans ce chapitre exposer les démarches successives qui nous ont permis d'orienter notre recherche et réaliser une synthèse des résultats obtenus. Nous discutons également des aspects positifs ainsi que des pistes d'amélioration pour les différentes parties de nos travaux. Nous donnons les applications potentielles que ces travaux pourraient initiées pour l'industrie automobile.

9.1 Démarche et synthèse des résultats

L'évolution des technologies informatiques, en particulier en terme de puissance de calcul graphique a été extrêmement importante ces dernières années (figure 9.1). Ajoutées à l'ouverture du pipeline graphique OpenGL (technique de vertex et pixel shaders), les techniques de rendu temps réel permettent aujourd'hui de simuler des phénomènes physiques complexes avec une grande précision et deviennent ainsi de plus en plus présents dans les outils numériques d'étude et de conception pour l'industrie automobile.

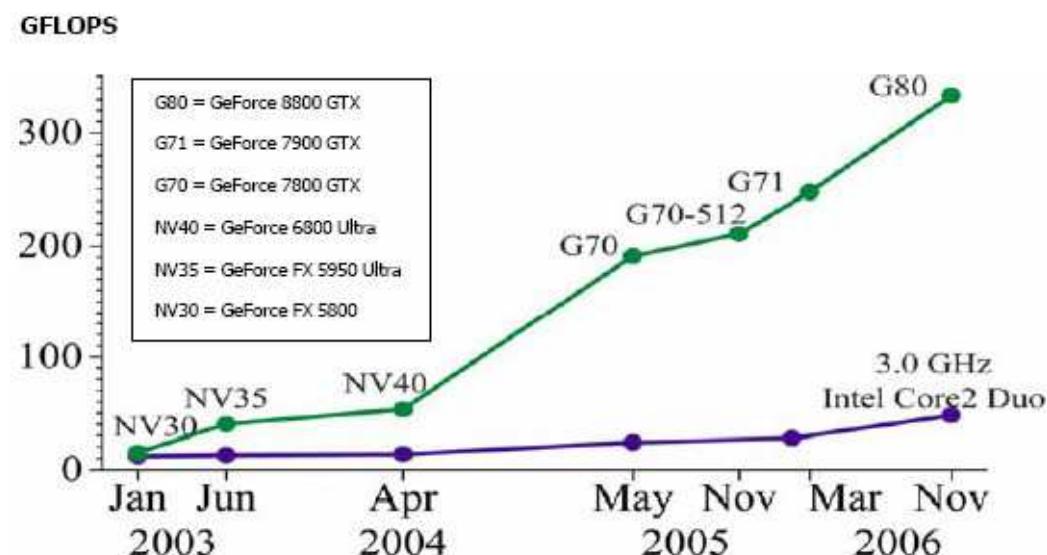


FIG. 9.1 – Comparatif de l'évolution de la puissance CPU / GPU (*source : Nvidia*)

L'objectif de cette thèse était d'exploiter les capacités des cartes graphiques pour améliorer le réalisme visuel des simulateurs de conduite, en particulier pour la simulation d'éclairage de nuit par des projecteurs automobile. La principale question à l'origine de ces travaux a donc été "*Quelles sont les pistes d'amélioration pour le réalisme de la simulation d'éclairage ?*".

Il est clairement apparu que la modélisation des projecteurs automobile en eux-mêmes était dores et déjà aboutie. En revanche, la réflexion de la lumière fournie par ces projecteurs sur les surfaces de la scène était modélisée par une fonction d'éclairage non représentative des différents matériaux constituant une scène routière. Nos travaux se sont donc orientés vers les techniques de simulation d'éclairage pour différents matériaux tels que l'as-

phalt, le marquage au sol, la végétation (herbe) ou encore les panneaux de signalisation. Deux catégories de matériaux se sont alors dégagées : les matériaux que l'on peut simuler de manière réaliste avec une fonction de réflectance (BRDF), tels que le marquage au sol ou encore les panneaux de signalisation et ceux dont l'aspect est beaucoup plus difficile à simuler comme l'asphalt ou encore la végétation de part leur hétérogénéité.

Nos travaux se sont alors concentrés sur le développement d'algorithme de rendu temps réel pour la représentation des matériaux hétérogènes. Nous avons commencé par réaliser un état de l'art sur les grandeurs physiques de la lumière, les algorithmes de synthèse d'image non temps réel ainsi que les techniques temps réel employées pour la simulation de conduite connaissant les besoins et contraintes de ce domaine particulier (fréquence image, systèmes d'affichage, etc.). Étant donné que nous nous intéressons à une catégorie de matériaux dont les variations géométrique sont de "petite" échelle (asphalt, herbe, etc.), nous avons écarté les méthodes basés sur des textures volumétriques (utilisées par exemple pour le rendu de forêt [DN04]). Nos travaux se sont focalisés sur les fonctions de textures bi-directionnelles (BTF) qui permettent de représenter un matériau par un ensemble de texture pour plusieurs conditions d'observation et d'éclairage.

En premier lieu, nous avons proposé et mis en place une méthode de génération de BTF synthétique à partir des algorithmes de rendu non temps réel étudiés afin de posséder une base de donnée de matériaux réalisistes à moindre coût. Nous avons respecté le format des échantillons mesurés par l'Université de Bonn afin de faciliter les tests sur les différents matériaux (synthétiques ou mesurés).

Ensuite nous avons développé un algorithme de rendu temps réel pour les BTF afin de pouvoir simuler une telle fonction qui a six dimensions et qui possède une taille mémoire très importante (jusqu'à plusieurs giga-octets). Cet algorithme se distingue en deux parties. La première partie, hors temps réel, permet de "compresser" la BTF afin d'aboutir à une représentation compacte dans le but de pouvoir charger les données dans la mémoire de la carte graphique, tout en limitant le nombre d'opérations nécessaires à la reconstruction de la BTF en temps réel. La deuxième partie a consisté à paramétriser et représenter les données sous forme de textures et à implémenter l'algorithme de reconstruction à l'aide des techniques de shaders (GLSL). Cet algorithme permet de simuler des matériaux hétérogènes en temps réel tout en respectant les contraintes imposées par la simulation de conduite en terme de fréquence image et de filtrage (mip-mapping) étant donné que les surfaces sont visibles pour des distances lointaines. La meilleure représentativité des surfaces donnée par cet algorithme permet d'améliorer significativement le réalisme des images calculées en temps réel.

La modélisation des surfaces par cette méthode améliorant le réalisme de la simulation, met en évidence les limitations des systèmes de production d'images temps réel, en particulier sur les échantillonnages spatial et temporel. En effet, nous avons vu que l'augmentation de la richesse des images (détails, contraste, etc.) engendre des artefacts visuels (bruit) lorsque l'observateur est en mouvement, ce qui est fréquemment le cas dans la simulation de conduite. Nous avons donc proposé et mis en place une technique de filtrage temporel afin d'éliminer ces artefacts visuels. Cette méthode améliore non seulement la qualité visuelle de la simulation mais fournit des indices de vitesse importants sur simulateur de conduite.

9.2 Discussion et perspectives

Nous avons mis en place une méthode pour la simulation d'éclairage réaliste de matériaux hétérogènes en temps réel basée sur les techniques de BTF et vu dans les résultats que le filtrage temporel était nécessaire pour aboutir à une simulation d'environnement réaliste lorsque l'observateur est en mouvement. Ceci nous amené à la question suivante : "*Est-il nécessaire de modéliser finement les surfaces puisque le filtrage temporel lisse le calcul d'éclairage ?*". Il est certain que la simulation des surfaces par les techniques de BTF couplée au filtrage nécessaire un grand nombre de calcul, mais pour aboutir à un réalisme visuel donnant une sensation d'immersion satisfaisante, il est important que la simulation soit correcte dès son lancement, en particulier quand le véhicule est à l'arrêt. La plupart des jeux vidéos utilisent des textures étirées pour donner une sensation de vitesse, ce qui pénalise le réalisme lorsque le véhicule est à l'arrêt. De plus, dans un soucis de validité des indices visuels et plus généralement de la simulation, il n'est pas correct d'étirer les textures qui donnent de fausses informations sur la vitesse et la direction réelle du véhicule. Simuler de manière fine les surfaces de la scène virtuelle et d'y appliquer un filtrage temporel est donc le seul moyen de garantir le réalisme indépendamment des conditions d'observation (à l'arrêt ou en mouvement).

La technique de filtrage temporel telle que nous l'avons envisagée porte sur la correction de phénomènes liés aux limitations des systèmes de production d'images. Toutefois, le paramétrage du modèle utilisé pourra tenir compte des caractéristiques de la perception humaine. En effet, la technique de filtrage fait intervenir des questions sur la perception du mouvement et fournit des indications sur la vitesse du conducteur en simulation de conduite.

Nous avons en particulier imaginer d'utiliser une fonction plus évoluée pour la génération du flou de mouvement, en particulier en tenant compte

de la persistance rétinienne afin de pondérer les texels pris en compte dans l'intégration du déplacement en fonction du moment auquel ils sont apparus. En effet, la persistance rétinienne dépend de la luminance de la scène visualisée, hors, comme nous l'avons vu, un système d'affichage offre une plage très restreinte de luminance comparée à la réalité. La perception du sujet en contexte de simulation de conduite est donc différente de la réalité pourtant la présence rétinienne joue un rôle dans la perception de la fluidité.

Comme nous l'avons vu, le filtrage permet d'éliminer les artefacts visuel, mais fournit également des indices de vitesse pour le conducteur. Nous avons débuté une étude sur la perception vitesse pour évaluer l'influence du filtrage sur la vitesse des conducteurs sur simulateur de conduite. L'exploitation des résultats est délicate dans la mesure où le potentiel effet sur la vitesse des conducteurs n'est pas suffisamment importante pour être détectée de manière significative. En particulier, les effets d'apprentissage (habitude à conduire sur simulateur) masquent les effets que l'on désire mesuré. Une solution possible consisterait à réaliser des expérimentations passives, c'est à dire où le conducteur ne contrôle pas la vitesse du véhicule mais doit en faire une estimation. Là encore, nous avons remarqué que l'estimation de la vitesse absolue par des indices visuels est une tâche très difficile et ne permet pas d'obtenir de résultats convaincants. La dernière solution, également passive, consiste à comparer les vitesses de véhicules affichés sur deux écrans différents, l'un avec filtrage et l'autre sans afin de déterminer l'écart de vitesse minimal perçu. Une étude menée selon ce principe par Breithecker et al. [BM06] montre qu'il n'y a pas d'écart significatif sur l'estimation des vitesses, ou seulement pour un filtrage très excessif qui dénature totalement l'aspect réaliste de la simulation. Toutefois, nous pensons que les indices fournis par le filtrage peuvent avoir une influence sur la perception de vitesse du conducteur sur simulateur immersif bien que nous n'avons pas pu mettre de protocole en place pour le démontrer.

En terme d'applications, ces travaux permettront d'étendre le périmètre de validité du simulateur d'éclairage : les résultats obtenus pourront augmenter le nombre de cas d'utilisation comme par exemple pour la simulation d'éclairage de différents types d'asphalt, de route humide, pour l'analyse de l'éblouissement provoqué par la réflexion sur la route des projecteurs des véhicules confrontants ou encore pour l'étude de stratégies d'éclairage intelligent. En terme de perspectives, ces travaux pourront être également une base pour des expérimentations sur la perception de vitesse et la vection (sensation du mouvement propre) qui sont autant de sujets étudiés par le Centre Technique de Simulation de Renault afin de garantir la validité des simulateurs de conduite.

Les travaux menés sur la simulation d'éclairage réaliste de matériaux hé-

térogènes en temps réel n'ont pas fait l'objet d'optimisations particulières liées au cas d'application initial envisagé. Par exemple, les plages d'angles contenus dans les Fonctions de texture bidirectionnelle auraient pu être diminuées dans le cadre de la simulation de conduite pour un observateur à une position donnée. Ainsi, la généralité des algorithmes de rendu développés pourrait permettre leurs utilisation pour d'autres domaines tels que le maquettage virtuel (logiciel P2V développé par Renault). Par exemple, le réalisme des matériaux constituant un intérieur de véhicule (cuir, plastique, bois, tissus, etc.) est difficile à obtenir et pourrait être améliorer à l'aide de ces travaux.

Enfin, la technique de BTF devrait devenir dans un proche avenir une méthode plus largement utilisée dans la mesure où elle améliore significativement le réalisme des simulations temps réel mais aussi des images calculées par des modèles plus complexes dans la mesure où elle peut être utilisé par des techniques non temps réel comme le lancer de rayon.

Annexe A

Code GLSL du filtrage temporel

A.1 Code du Vertex Shader

```
void main(void)
{
    // similar to gl_ProjectionMatrix * gl_ModelViewMatrix * gl_Vertex;
    gl_TexCoord[1] = ftransform();

    //gl_TextureMatrix[2] contains modelview for frame at time t-1
    gl_TexCoord[2] = gl_ProjectionMatrix * gl_TextureMatrix[2] * gl_Vertex;

    gl_Position = gl_TexCoord[1];
}
```

A.2 Code du Fragment Shader

```
uniform samplerRect sceneTexture;
uniform vec3 mb;
uniform vec3 res;

// mb.x => blurScale
// mb.y => divide coef for length of vector
// mb.z => max samples

vec4 blur(int samp, vec2 spVelocity)
{
    vec4 a = vec4(0.0,0.0,0.0,0.0);
    vec2 v, vTmp;
    float sum = 0.0;
```

```

for(int i = 0; i < samp; ++i)
{
    float t = float(i) / (float(samp)-1.0);
    sum += t;

    vTmp = spVelocity * t;

    v = vec2(gl_FragCoord) + vTmp;

    if( v.x < 0.0 || v.y < 0.0 || v.x > 2.0*res.x )
        v = vec2(gl_FragCoord) - vTmp;

    a += (1.0-t) * textureRect( sceneTexture, v);
}

return a/sum;
}

void main(void)
{
    vec4 tex = textureRect(sceneTexture, vec2(gl_FragCoord));

    // limit to select filtered surface in screen space
    if( gl_FragCoord.x > res.z-2.0 && gl_FragCoord.x < res.z+2.0)
    {
        gl_FragColor = vec4(1.0,0.0,0.0,0.0);
    }
    else
    {
        if(gl_FragCoord.x < res.z)
            gl_FragColor = tex;
        else
        {
            vec4 P      = gl_TexCoord[1];
            vec4 Pprev = gl_TexCoord[2];

            P.xy      = P.xy / P.w;
            Pprev.xy = Pprev.xy / Pprev.w;

            vec2 dP = vec2(res.x, res.y) * (P.xy - Pprev.xy);

            int samples = int(min(length(dP), mb.z ));

            if(samples <= 1)

```

```
    gl_FragColor = tex;
else
    gl_FragColor = blur(samples, dP);
}
}
}
```


Annexe B

Optimisation non-linéaire : Méthode de Levenberg Marquardt

B.1 Définition

Cette méthode apporte une solution au problème de minimisation d'une fonction ayant plusieurs paramètres (non-linéaires) ainsi que plusieurs variables. C'est une méthode hybride basée sur les méthodes de Gauss-Newton et la méthode de descente de gradient.

Son application la plus courante est la régression linéaire par la méthode des moindres carrés : on cherchera l'ensemble des paramètres p afin de minimiser la somme des carrés des déviations entre les données y_i et la valeur de la fonction pour la variable d'entrée correspondante $f(x_i)$, pour n couples de données (x_i, y_i) :

$$S_p = \sum_{i=1}^n [y_i - f_p(x_i)]^2$$

La solution est calculée par un processus itératif : on attribue des valeurs initiales aux paramètres p , de préférence proches du résultat final afin qu'il y ait possibilité de convergence au cours des itérations (bien qu'il ne soit pas toujours aisément d'ajuster par intuition les valeurs des paramètres pour une fonction complexe de plusieurs dimensions). Puis à chaque itération, p est remplacé par une nouvelle estimation $p+q$. Afin de déterminer q , les fonctions $f_i(p+q)$ sont approximées par linéarisation :

$$f(p+q) \approx f(p) + Jq$$

Où J est le jacobien de f avec les paramètres p .
 S est minimale lorsque $\nabla_q S = 0$, ce qui donne $(J^T J)q = -J^T f$, permettant

166Optimisation non-linéaire : Méthode de Levenberg Marquardt

de calculer q en inversant $J^T J$. La méthode de Levenberg Marquart consiste à approcher cette équation en y ajoutant un facteur d'amortissement λ qui est ajusté à chaque itération :

$$(J^T J + \lambda I)q = -J^T f$$

Si la décroissance de S est grande entre deux itérations, la valeur de λ est diminuée, comme dans l'algorithme de Gauss-Newton, dans le cas inverse (si l'itération ne permet pas de diminuer significativement S), la valeur de λ est augmentée ce qui rapproche l'algorithme à celui de descente du gradient. L'algorithme s'arrête soit lorsqu'un nombre d'itérations maximal est atteint, soit si un seuil minimum est atteint et les valeurs des paramètres p sont alors renvoyées.

B.2 Code scilab

```
clc;
clear;

x1 = [1:1:81];
x2 = [1:1:81];

meanY = read('mean.txt',1,6561);

X = read('diffus.txt',81, 81);
Y = read('spec.txt',81, 81);

// convert to 2D tabular
for i=1:81
  for j=1:81
    meanY2D(i, j) = meanY(1, (i-1)*81+j);
  end
end

//solve
function e=f1(params,m)
  a=params(1);
  b=params(2);
  c=params(3);
  d=params(4);
  e= (meanY2D - (a + b*X + c*Y.^d)).^2;
endfunction
```

```
//call
[abcd,v,info]=lsqrssolve([0.5; 0.5; 0.5; 5.0],f1,size(X,1)*size(Y,1));
abcd
norm(v)
info

// plot function
for i=1:81,
for j=1:81,
Yopt(i,j) = abcd(1) + abcd(2)*X(i,j) + abcd(3)*Y(i,j).^abcd(4);
end
end

plot3d(x1, x2, Yopt);
```


Annexe C

Code GLSL du rendu temps réel de BTF

C.1 Code du Vertex Shader

```
uniform vec4 lightPos;
uniform vec4 eyePos;

void main(void)
{
    vec4 lPos          = gl_ModelViewMatrix * lightPos;
    vec4 gVertexPos   = gl_ModelViewMatrix * gl_Vertex; // pMv

    gl_Position       = ftransform();

    vec3 Id      = vec3(gl_ModelViewMatrixInverse * (lPos - gVertexPos));
    vec3 Ed      = vec3(gl_ModelViewMatrixInverse * (eyePos - gVertexPos));

    vec3 I       = normalize(Id);
    vec3 E       = normalize(Ed);

    // local TBN space

    vec3 N      = vec3(gl_Normal);
    // tangent is provided by vertex color
    vec3 T      = normalize(vec3(gl_Color));
    vec4 B      = vec4(cross(T, N), 1.0);

    vec3 IL = normalize(vec3( dot(I,B), dot(I,T), dot(I,N)));
    vec3 EL = normalize(vec3( dot(E,T), -dot(E,B), dot(E,N)));
}
```

```

vec3 H = normalize(I + E);

float dh    = dot(N, H);
float zi    = dot(I, N );
float aze   = 2.0*acos(dot(E, N ))/PI();
float azi   = 2.0*acos(zi)/PI() ;

gl_TexCoord[0]  = gl_MultiTexCoord0;
gl_TexCoord[1]  = vec4(azi, aze,0.0,0.0);
gl_TexCoord[2]  = vec4(length(Id),length(Ed),dh,zi);
gl_TexCoord[3]  = vec4(IL,1.0);
gl_TexCoord[4]  = vec4(EL,1.0);
gl_TexCoord[5]  = vec4(N,1.0);
}

```

C.2 Code du Vertex Shader

```

uniform sampler3D weightMap;
uniform sampler3D brdfMap;
uniform samplerCube cubeMap;
uniform float lp;

void main(void)
{
    // compute mipmap level
    int level = int(4.0-min( 4.0, max(log2(gl_TexCoord[2].y/100.0),0.0) ));

    vec3 sum; // Y, Cb, Cr

    vec4 w[3];
    vec4 eY, eY1, eY2;
    vec4 eCbCr, eCbCr1, eCbCr2;
    vec2 wio;
    vec3 rvb;

    // surface texture coordinate
    vec2 xy = vec2(1-gl_TexCoord[0].x, 1*(1-gl_TexCoord[0].y));

    // rebuild BTF for levels
    for(int i = 0; i<=3*level; i+=3)
    {
        float flev = 32.0*float(i/3);

```

```
// get values for wieghtmap with scale and bias
w[0] = 25.0 * ( texture3D(weightMap,
                           vec3(xy, float(i+2)/16.0))-0.5)*2.0 );
w[1] = 19.0 * ( texture3D(weightMap,
                           vec3(xy, float(i) /16.0 ) )-0.5)*2.0 );

float iz1 = 0.5*ste.y + 0.5/512.0;
float iz2 = 0.5*ste.x + 0.5/512.0;

vec2 lum = (flev + ((0.5*(gl_TexCoord[3].x+1.0))/256.0,
                      (0.5*(gl_TexCoord[3].y+1.0) ));

eY1 = 0.15 * (texture3D(brdfMap, vec3(lum, iz1) )-0.5)*2.0;
eY2 = 0.15 * (texture3D(brdfMap, vec3(lum, iz2) )-0.5)*2.0;

// manual linear interpolation
eY = (1.0-ste.w) * eY1 + ste.w * eY2;

eCbCr1 = 0.15 * (texture3D(brdfMap, vec3(lum, 0.5+iz1) )-0.5)*2.0;
eCbCr2 = 0.15 * (texture3D(brdfMap, vec3(lum, 0.5+iz2) )-0.5)*2.0;

eCbCr = (1.0-ste.w) * eCbCr1 + ste.w * eCbCr2;

// Y
for(int c = 0; c<4; c++)
    sum.r += wk*(w[0][c] * eY[c]);

// Cb
sum.g += wk*(w[1].r * eCbCr.r);
sum.g += wk*(w[1].g * eCbCr.g);

// Cr
sum.b += wk*(w[1].b * eCbCr.b);
sum.b += wk*(w[1].a * eCbCr.a);
}

// phong model parameters from levenberg-marquardt
sum.r += 0.188
        + 0.134 * gl_TexCoord[2].w
        + 0.353 * pow(gl_TexCoord[2].z, 10.94);
sum.g += 0.52;
sum.b += 0.48;
```

```
// YCrCb to RGB conversion
rvb[0] = sum[0] + 1.40200 * (sum[2] - 0.5);
rvb[1] = sum[0] - 0.34414 * (sum[1] - 0.5) - 0.71414 * (sum[2] - 0.5);
rvb[2] = sum[0] + 1.77200 * (sum[1] - 0.5);

// set final color
gl_FragColor = vec4(rvb.b, rvb.g, rvb.r,1.0)/(gl_TexCoord[2].x);
}
```

Bibliographie

- [AMH02] Tomas Akenine-Möller and Eric Haines. *Real-Time Rendering (2nd Edition)*. AK Peters, Ltd., July 2002.
- [Arv86] James R. Arvo. Backward Ray Tracing. In *ACM SIGGRAPH '86 Course Notes - Developments in Ray Tracing*, volume 12, 1986.
- [AS84] D. Ariel and R. Sivan. False cue reduction in moving flight simulators. *IEEE Transactions on Systems, Man and Cybernetics*, 14(4) :665–671, 1984.
- [BA87] Peter J. Burt and Edward H. Adelson. The laplacian pyramid as a compact image code. *Readings in computer vision : issues, problems, principles, and paradigms*, 1 :671–679, 1987.
- [BE01] Gabriel J. Brostow and Irfan Essa. Image-based motion blur for stop motion animation. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 561–566, 2001.
- [Bli78] James F. Blinn. Simulation of wrinkled surfaces. In *SIGGRAPH '78 : Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 286–292, New York, NY, USA, 1978. ACM Press.
- [BM06] Huesmann A. Stamminger M. Breithecker M., Lankes F. Increasing perceived velocity by means of texture-based motion blur. *Driving Simulation Conference Europe*, 1 :273–283, 2006.
- [BSW92] Christopher S. Bretherton, Catherine Smith, and John M. Wallace. An intercomparison of methods for finding coupled patterns in climate data. 5(6) :541–560, 1992.
- [CCWG88] Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. A progressive refinement approach to fast radiosity image generation. In *SIGGRAPH '88 : Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 75–84, New York, NY, USA, 1988. ACM Press.
- [CG85] Michael F. Cohen and Donald P. Greenberg. The hemi-cube : a radiosity solution for complex environments. In *SIGGRAPH '85 : Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 31–40, New York, NY, USA, 1985. ACM Press.
- [CT92] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. pages 42–59, 1992.

- [DBP01] Kemeny A. Michelin S. Arquès D. Dumont Bècle P., Ferley E. Multi-texturing approach for paint appearance simulation on virtual vehicles. *Proceedings of the Driving Simulation Conference, France*, pages 123–133, 2001.
- [DK00] Frank Dachille and Arie Kaufman. High-degree temporal antialiasing. In *Proceedings of Computer Animation 2000*, pages 49–54, May 2000.
- [dl24] Commission Internationale de l'Eclairage. Cie proceedings (1926). *Cambridge : Cambridge University Press*, 1, 1924.
- [dl51] Commission Internationale de l'Eclairage. *CIE Proceedings (1951)*, 1(4) :37, 1951.
- [DLHS01] Katja Daubert, Hendrik P. A. Lensch, Wolfgang Heidrich, and Hans-Peter Seidel. Efficient cloth modeling and rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 63–70, London, UK, 2001. Springer-Verlag.
- [dlR99] RENAULT Direction de la Recherche. Résultats des mesures comparatives sur simulateur d'éclairage et sur piste. *rapport interne*, 1999.
- [DN04] Philippe Decaudin and Fabrice Neyret. Rendering forest scenes in real-time. In *Rendering Techniques (Eurographics Symposium on Rendering - EGSR)*, pages 93–102, june 2004.
- [DvGNK99] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 18(1) :1–34, 1999.
- [EY36] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *j-PSYCHO*, 1(3) :211–218, 1936.
- [FH04] Jiri Filip and Michal Haindl. Non-linear reflectance model for bidirectional texture function synthesis. In *ICPR '04 : Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 1*, pages 80–83, Washington, DC, USA, 2004. IEEE Computer Society.
- [GTGB84] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. In *SIGGRAPH '84 : Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 213–222, New York, NY, USA, 1984. ACM Press.
- [Jen96] Henrik Wann Jensen. Global illumination using photon maps. In *Proceedings of the eurographics workshop on Rendering techniques '96*, pages 21–30, London, UK, 1996. Springer-Verlag.
- [KL97] Nandakishore Kambhatla and Todd K. Leen. Dimension reduction by local principal component analysis. *Neural Comput.*, 9(7) :1493–1516, 1997.
- [KM99] Jan Kautz and Michael D. McCool. Interactive rendering with arbitrary brdfs using separable approximations. In *SIGGRAPH '99 : ACM SIGGRAPH 99 Conference abstracts and applications*, page 253, New York, NY, USA, 1999. ACM Press.
- [Lew93] Robert R. Lewis. Making Shaders More Physically Plausible. In *Fourth Eurographics Workshop on Rendering*, number Series EG 93 RW, pages 47–62, Paris, France, 1993.

- [LFTG97] Eric P. F. Lafourture, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-linear approximation of reflectance functions. In *SIGGRAPH '97 : Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 117–126, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.
- [LP99] Kemeny A. Lecocq P., Kelada J-M. Interactive headlight simulation. *Proceedings of DSC99, Driving Simulation Conference*, 1 :173–180, 1999.
- [LP01] Arquès D. Kemeny A. Lecocq P., Michelin S. Simulation temps réel d'éclairage en présence de brouillard. *Numéro spécial de la revue CFAO et Informatique Graphique*, 16 :51–66, 2001.
- [LT05] Arquès D. Michelin S. Lefebvre T., Kemeny A. Rendering of complex materials for driving simulators. *Proceedings of Driving Simulation Conference North America (Orlando)*, 1 :313–322, 2005.
- [LT06] Arquès D. Michelin S. Lefebvre T., Kemeny A. Speed and motion cues provided by temporal anti-aliasing in driving simulator. *Proceedings of Driving Simulation Conference ASIA (Tsukuba)*, 1 :211–219, 2006.
- [MAA01] Michael D. McCool, Jason Ang, and Anis Ahmad. Homomorphic factorization of brdfs for high-performance rendering. In *SIGGRAPH '01 : Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 171–178, New York, NY, USA, 2001. ACM Press.
- [Mac67] J. MacQueen. Some methods for classification and analysis of multivariate observations. *in Proc. 5th Berkeley Symp. Math Stat. and Prob.*, 1(1) :281–296, 1967.
- [Mar63] Donald W. Marquardt. An algorithm for least-squares estimation of non-linear parameters. *SIAM Journal on Applied Mathematics*, 11(2) :431–441, 1963.
- [MDK02] G. Reymond M. Dagdelen and A. Kemeny. Analysis of the visual compensation in the renault driving simulator. *Driving Simulation Conference, Europe*, pages 109–119, 2002.
- [MLH02] David K. McAllister, Anselmo Lastra, and Wolfgang Heidrich. Efficient rendering of spatial bi-directional reflectance distribution functions. In *HWWS '02 : Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 79–88, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.
- [MLK03] P. N. Belhumeur D. J. Kriegman M. L. Koudelka, S. Magda. Acquisition, compression and synthesis of bidirectional texture functions. *3rd International Workshop on Texture Analysis and Synthesis*, pages 75–82, 2003.
- [MMK03a] J. Meseth, G. Müller, and R. Klein. Preserving realism in real-time rendering of bidirectional texture functions. In *OpenSG Symposium 2003*, pages 89–96. Eurographics Association, Switzerland, April 2003.
- [MMK03b] G. Müller, J. Meseth, and R. Klein. Compression and real-time rendering of measured btfs using local pca. In T. Ertl, B. Girod, G. Greiner, H. Niemann, H.-P. Seidel, E. Steinbach, and R. Westermann, editors, *Vision, Modeling and Visualisation 2003*, pages 271–280. Akademische Verlagsgesellschaft Aka GmbH, Berlin, November 2003.

- [MMS⁺] G. Müller, J. Meseth, M. Sattler, R. Sarlette, and R. Klein. Acquisition, synthesis and rendering of bidirectional texture functions.
- [MMWR93] J.P. Wann M. Mon-Williams and S. Rushton. Binocular vision in a virtual world : visual deficits following the wearing of a head-mounted display. *Ophthalmic Physiol Opt*, 13 :387–391, 1993.
- [MWT99] M. Mon-Williams and J.R. Tresilian. Some recent studies on the extraretinal contribution to distance perception. *Perception*, 28 :167–181, 1999.
- [NF77] Hsia J. Ginsberg I. Limperis T. Nicodemus F., Richmond J. Geometric considerations and nomenclature for reflectance. *Monograph*, page 71, October 1977.
- [NKON90] Eihachiro Nakamae, Kazufumi Kaneda, Takashi Okamoto, and Tomoyuki Nishita. A lighting model aiming at drive simulators. In *SIGGRAPH '90 : Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 395–404, New York, NY, USA, 1990. ACM Press.
- [Nvi07] Nvidia. Nvidia geforce 8 series, 2007.
- [PFTV88] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical recipes in C : the art of scientific computing*. Cambridge University Press, New York, NY, USA, 1988.
- [Pho75] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6) :311–317, 1975.
- [POJ05] Fabio Policarpo, Manuel M. Oliveira, and ao L. D. Comba Jo' Real-time relief mapping on arbitrary polygonal surfaces. In *I3D '05 : Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 155–162, New York, NY, USA, 2005. ACM Press.
- [Pov96] Povray. Povray - the persistence of vision ray tracer, 1996.
- [Ram02] Ravi Ramamoorthi. Analytic pca construction for theoretical analysis of lighting variability in images of a lambertian object. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(10) :1322–1333, 2002.
- [Ros05] Randi J. Rost. *OpenGL(R) Shading Language (2nd Edition)*. Addison-Wesley Professional, 2005.
- [SC97] Peter Shirley and Kenneth Chiu. A low distortion map between disk and square. *J. Graph. Tools*, 2(3) :45–52, 1997.
- [Sci89] Scilab. <http://www.scilab.org>, 1989.
- [SdM87] S. J. Schein and F. M. de Monasterio. Mapping of retinal and geniculate neurons onto striate cortex of macaque. *J.Neurosci*, 7 :996–1009, 1987.
- [Sha49] C. E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1) :10–21, 1949.
- [Sut64] Ivan E. Sutherland. Sketch pad a man-machine graphical communication system. In *DAC '64 : Proceedings of the SHARE design automation workshop*, pages 6.329–6.346, New York, NY, USA, 1964. ACM Press.

- [SVLD03] F. Suykens, K. Vom, A. Lagae, and P. Dutr. Interactive rendering with bidirectional texture functions, 2003.
- [Tat06] Natalya Tatarchuk. Practical parallax occlusion mapping with approximate soft shadows for detailed surface rendering. In *SIGGRAPH '06 : ACM SIGGRAPH 2006 Courses*, pages 81–112, New York, NY, USA, 2006. ACM Press.
- [TZL⁺02] Xin Tong, Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, and Heung-Yeung Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. In *SIGGRAPH '02 : Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 665–672, New York, NY, USA, 2002. ACM Press.
- [War92] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *SIGGRAPH '92 : Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 265–272, New York, NY, USA, 1992. ACM Press.
- [Whi80] Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6) :343–349, 1980.

