即時資料驅動描繪之參數表示式與張量近似演算法

學生：蔡侑庭　　　　　　　　　　　　　　　　指導教授： 施仁忠 博士

國立交通大學資訊工程學系博士班

摘　　要

過去幾十年來，電腦繪圖領域之專家學者致力於開發各式各樣特殊新穎的視覺效果，以期能夠利用電腦合成出如同真實相片般之影像。為了達到高畫質擬真影像輸出，許多被稱為資料驅動描繪之進階繪圖演算法，會事先運用複雜程序處理輸入之三維場景以獲得必要資訊，抑或從真實世界當中預先擷取影像資訊，以便於執行時期能藉由這些視覺資訊快速重建出所希望描繪之視覺特效。然而隨著人類對於影像逼真程度之需求日益增高，預先採樣資訊之資料量也隨之水漲船高，不但耗費大量的儲存空間，也同時增加執行時期之資料存取時間以及描繪運算量。

為了解決上述問題，我們可以透過簡潔的表示式來有效率地描述預先採樣資訊，並應用精密複雜的壓縮方法以進一步減少資料儲存量。然而從以往相關研究成果卻可以觀察到，要能夠大幅度減少資料量，同時保有即時運算效能，是相當困難的一項挑戰，因為此兩項主要目標之間經常具有互斥性質。有鑑於此，我們於本論文當中將特別著重相關議題之探討，針對資料之表示式與近似演算法兩大主題，研究兩者於即時視覺資訊描繪之發展與應用。關於資料表示式，本論文將介紹兩項特殊參數表示式：單變量以及多變量球面輻射基底函數，主要用以描述視覺資訊中常見的輻射與照明度函數，同時更可以進一步拓展至各種於球表面採樣之資訊。而資料壓縮方面，本論文則提出兩項嶄新張量近似演算法：叢集以及稀疏叢集張量近似法，不但能夠充分利用視覺資訊之相依性關係，達到資料量減少與即時快速重建皆可兼顧之目的，也能夠延伸至各式高維度大型科學資料之壓縮、分析、或描述。

本論文最終更將深入研討如何透過所提出之方法來近似與重建預先採樣資訊，以期能於計算量、壓縮誤差、以及儲存空間三者之間達到最佳平衡點，同時更選取於電腦繪圖與視覺領域中具代表性之數項應用作為實驗基礎。實驗成果充分驗證本論文所提方法之可行性與可塑性，不但能夠有效保留原始資訊當中之視覺特徵，同時也能輕易達到即時描繪速率。如此一來，高畫質之擬真影像合成便可以於現今一般個人電腦上實現，不再是超級電腦或電影製作所專屬之權益。

Parametric Representations and Tensor Approximation Algorithms for Real-Time Data-Driven Rendering

Student: Yu-Ting Tsai Advisor: Dr. Zen-Chung Shih

Department of Computer Science
National Chiao Tung University

ABSTRACT

Over the last decades, computer graphics and vision researchers have focused on developing novel visual effects for computer-generated photo-realistic images. To achieve high-quality output, many state-of-the-art rendering algorithms, which are known as data-driven rendering, pre-process an input three-dimensional scene with complex procedures to obtain necessary data, or pre-capture the real world with a set of images. The desired visual effects are then reconstructed from the pre-sampled observations for efficient run-time rendering. Nevertheless, with the increasing demand of more and more photo-realistic image synthesis, the amount of pre-sampled data expands accordingly. It not only consumes a great deal of storage space, but also increases data access time and rendering costs at run-time.

In order to solve this issue, we can adopt a compact representation to efficiently describe the pre-sampled observations and further apply sophisticated approximation methods to reduce the amount of data, but achieving real-time performance at the same time is frequently another challenging problem. In this dissertation, we thus focus on data representations and approximation algorithms for real-time rendering of visual data sets. Two novel parametric representations, univariate and multivariate *spherical radial basis functions* (SRBFs), and two sophisticated tensor approximation algorithms, *clustered tensor approximation* (CTA) and *K-clustered tensor approximation* (K-CTA), are proposed to exploit the coherence in visual data sets. The univariate and multivariate SRBFs are especially suitable for modeling radiance and illumination data sets that are common and ubiquitous in computer graphics and vision. Additionally, SRBFs also can be applied to represent various kinds of observations on the unit hyper-sphere. As for CTA and K-CTA, they are developed based on tensor approximation to simultaneously achieve high compression ratios and real-time rendering performance for multi-dimensional visual data sets. CTA and K-CTA are also general approximation algorithms that are not restricted to specific applications. They can be employed to compress, analyze, or represent other large-scale scientific data sets that intrinsically exhibit multi-dimensional structures.

Last but not the least, we also investigate how to approximate and reconstruct the pre-sampled observations with a trade-off among computational costs, compression errors, and storage space. The proposed methods are further applied to various representative applications in computer graphics and vision. Our experiments show promising results in which important features of visual data sets are well-preserved after approximation and rendered at real-time rates. As a result, high-quality photo-realistic image synthesis can be efficiently realized on modern personal computers, not privileged to supercomputers or film production anymore.

Acknowledgements

First of all, I appreciate my advisor, Dr. Zen-Chung Shih, for his support and guidance during the hard time of my stay at the Department of Computer Science, National Chiao Tung University. I would also like to thank the members of Computer Graphics Laboratory, especially Chih-Hao Chen for helping me with parts of the precomputed radiance transfer program, Shr-Ching Weng and Qing-Zhen Jiang for profound discussions on the proposed importance sampling algorithm, Yu-Pao Tsai for his advice on the applications of the proposed methods, Dr. Wen-Chieh Lin and Kuei-Li Fang for suggestions and implementation of the hierarchical fitting algorithm, Jia-Yin Ji for preparing the model *Cloth*, and to name a few.

Finally, special thanks are due to Ya-Mei Chen for her encouragement and assistance during the last few years of my Ph.D. program. I would also like to dedicate this dissertation to my parents for their selfless love and support. Without all of you, I would never receive the Ph.D. degree in computer science, and this dissertation would not have been written.

Contents

Abstract (in Chinese) **i**

Abstract (in English) **ii**

Acknowledgements **iv**

Contents **v**

List of Figures **ix**

List of Tables **xii**

List of Algorithms **xiii**

List of Symbols **xiv**

1 Introduction **1**

1.1 Background 1

1.2 Motivations and Methods 2

1.3 Applications of Data-Driven Rendering 4

1.4 Contributions 5

1.5 Dissertation Organization 7

2 Literature Review **9**

2.1 Data Representations 9

2.1.1 Functional Models 9

2.1.2 Spherical Radial Basis Functions 11

2.1.3 Parameterization 11

2.2 Dimensionality Reduction 12

2.2.1 Linear Models 12

2.2.2 Non-Linear Models 13

2.2.3 Multi-Linear Models 14

2.3 Data-Driven Rendering 15

2.3.1 Spatially-Varying Appearance Models 16
2.3.2 Radiance Transfer 17

3 Univariate Spherical Radial Basis Functions 20
3.1 Mathematical Formulation 20
3.2 Characteristics of Univariate SRBFs 23
3.2.1 Spherical Singular Integral 23
3.2.2 Spatial Localization 24
3.2.3 Univariate Gaussian SRBFs and Von Mises-Fisher Distributions 24
3.3 Uniform Univariate SRBF Representation 26
3.3.1 Ordinary Least-Squares Projection 26
3.3.2 Regularized Least-Squares Projection 27
3.4 Scattered Univariate SRBF Representation 27
3.4.1 Fitting Algorithm 27
3.4.2 Initial Guess 28
3.5 Mathematical Proofs 30
3.5.1 Convolution of Univariate Gaussian SRBFs 30
3.5.2 Relation between Univariate Gaussian SRBF and Von Mises-Fisher Distribution 32

4 Multivariate Spherical Radial Basis Functions 33
4.1 Mathematical Formulation 33
4.1.1 Basic Definitions 33
4.1.2 Example 35
4.2 Scattered Multivariate SRBF Representation 35
4.2.1 Fitting Algorithm 35
4.2.2 Initial Guess 36
4.3 Parameterized Multivariate SRBF Representation 37
4.3.1 Overview 37
4.3.2 Example 39
4.3.3 Fitting Algorithm 40
4.4 Hierarchical Fitting Algorithm 40
4.4.1 Upsampling Stage 43
4.4.2 Optimization Stage 44

5 Clustered Tensor Approximation 46
5.1 Preliminaries and Background 46
5.1.1 Basic Definitions 47
5.1.2 Tensor Approximation 48
5.2 Mathematical Formulation 49

5.3 Algorithm . . . 51
5.3.1 Overview . . . 51
5.3.2 Clustering Stage . . . 53
5.3.3 Update Stage . . . 54
5.4 Implementation Issues . . . 54
5.4.1 Initial Guess . . . 54
5.4.2 Global Basis Matrices . . . 55
5.5 Mathematical Proofs . . . 56

6 K-Clustered Tensor Approximation 59
6.1 Mathematical Formulation . . . 60
6.2 Algorithm . . . 61
6.2.1 Overview . . . 61
6.2.2 Clustering Stage . . . 62
6.2.3 Update Stage . . . 66
6.3 Implementation Issues . . . 67
6.3.1 Initial Guess . . . 67
6.3.2 Degeneracy and Convergence . . . 68
6.4 Mathematical Proofs . . . 69
6.4.1 Greedy Search . . . 69
6.4.2 Optimal Projection . . . 70

7 Application I: Illumination and Reflectance Functions 73
7.1 High Dynamic Range Environment Maps . . . 73
7.1.1 Problem Formulation . . . 73
7.1.2 Experimental Results . . . 74
7.2 Bidirectional Reflectance Distribution Functions . . . 78
7.2.1 Univariate SRBFs . . . 79
7.2.2 Multivariate SRBFs . . . 80
7.2.3 Experimental Results . . . 81
7.3 Importance Sampling of Direct Illumination . . . 83
7.3.1 Problem Formulation . . . 88
7.3.2 Importance Sampling Algorithm . . . 90
7.3.3 Experimental Results . . . 90

8 Application II: Spatially-Varying Appearance Models 95
8.1 Bidirectional Texture Functions . . . 95
8.1.1 Tensor Approximation Algorithms . . . 96
8.1.2 Multivariate SRBFs . . . 102
8.1.3 Comparisons and Discussions . . . 111

8.2 View-Dependent Occlusion Texture Functions 118
8.2.1 Problem Formulation 118
8.2.2 Rendering Issues 120
8.2.3 Experimental Results 122

9 Application III: Radiance Transfer 127
9.1 Precomputed Radiance Transfer 127
9.1.1 Problem Formulation 128
9.1.2 Algorithm 130
9.1.3 Experimental Results 133
9.1.4 Discussions 135
9.2 Bi-Scale Radiance Transfer 139
9.2.1 Problem Formulation 140
9.2.2 Algorithm 144
9.2.3 Experimental Results 146

10 Conclusions and Future Work 150
10.1 Conclusions 150
10.1.1 Spherical Radial Basis Functions 150
10.1.2 Tensor Approximation Algorithms 151
10.1.3 Summary 151
10.2 Future Work 152
10.2.1 Flexible Data Representations 152
10.2.2 Powerful Approximation Algorithms 153
10.2.3 General Data-Driven Applications 154

Bibliography 156

Vita 172

List of Figures

3.1 Two-dimensional and three-dimensional plots of univariate Gaussian spherical radial basis functions. 21

3.2 A univariate spherical function in univariate spherical radial basis function expansions. 22

4.1 Approximate a bidirectional texture function using the proposed hierarchical fitting algorithm. 42

5.1 The decomposed core tensor and basis matrices of a third order tensor based on tensor approximation. 48

5.2 Decompose a third order tensor using clustered tensor approximation. 50

6.1 Decompose a third order tensor using K-clustered tensor approximation. 61

7.1 Examples of high dynamic range environment maps. 74

7.2 Squared error ratio plots for high dynamic range environment maps. 75

7.3 Reconstructed images of the high dynamic range environment map *St. Peter's Basilica*. 76

7.4 Reconstructed images of the high dynamic range environment maps *Eucalyptus Grove*, *Grace Cathedral*, and *The Uffizi Gallery*. 77

7.5 Examples of bidirectional reflectance distribution functions. 78

7.6 Squared error ratio plots for bidirectional reflectance distribution functions. . . 83

7.7 Reconstructed images of the bidirectional reflectance distribution function *Blue Metallic Paint*. 84

7.8 Reconstructed images of the bidirectional reflectance distribution function *Krylon Blue*. 85

7.9 Reconstructed images of the bidirectional reflectance distribution function *Nickel*. 86

7.10 Rendered images for bidirectional reflectance distribution functions. 87

7.11 Rendered images based on the proposed importance sampling algorithm for *Bunny* with the bidirectional reflectance distribution function *Blue Metallic Paint*. 92

7.12 Rendered images based on the proposed importance sampling algorithm for the model *Buddha* with the bidirectional reflectance distribution function *Krylon Blue*. 93

7.13 Rendered images based on the proposed importance sampling algorithm for the model *Venus* with the bidirectional reflectance distribution function *Nickel*. . . . 94

8.1 An example of a bidirectional texture function. 96
8.2 Squared error ratio plots based on different tensor approximation algorithms for bidirectional texture functions. 102
8.3 Rendered images based on different tensor approximation algorithms for bidirectional texture functions. 103
8.4 Reconstructed images of the bidirectional texture function for the material *Fiber* based on different tensor approximation algorithms. 104
8.5 Reconstructed images of the bidirectional texture function for the material *Rough-Hole* based on different tensor approximation algorithms. 105
8.6 Reconstructed images of the bidirectional texture function for the material *Sponge* based on different tensor approximation algorithms. 106
8.7 Reconstructed images of the bidirectional texture function for the material *Wool* based on different tensor approximation algorithms. 107
8.8 Rendered images based on the parameterized multivariate spherical radial basis function representation for bidirectional texture functions. 112
8.9 Reconstructed images of the bidirectional texture function for the material *Carpet* based on the parameterized multivariate spherical radial basis function representation. 113
8.10 Reconstructed images of the bidirectional texture function for the material *Impalla* based on the parameterized multivariate spherical radial basis function representation. 114
8.11 Reconstructed images of the bidirectional texture function for the material *Sponge* based on the parameterized multivariate spherical radial basis function representation. 115
8.12 Reconstructed images of the bidirectional texture function for the material *Wool* based on the parameterized multivariate spherical radial basis function representation. 116
8.13 Examples of the proposed view-dependent occlusion texture functions. 119
8.14 Comparison between different encoding schemes for view-dependent occlusion texture functions 121
8.15 Error texel ratio plots based on different encoding schemes for view-dependent occlusion texture functions. 123
8.16 Error texel ratio plots based on different tensor approximation algorithms for view-dependent occlusion texture functions. 124
8.17 Reconstructed images of the view-dependent occlusion texture function for the material *Fiber*. 125

8.18 Reconstructed images of the view-dependent occlusion texture function for the material *RoughHole*. 126

9.1 System diagram of the proposed all-frequency precomputed radiance transfer algorithm. 130
9.2 Tensor representation for radiance transfer matrices in the proposed all-frequency precomputed radiance transfer algorithm. 132
9.3 Rendered images based on the proposed all-frequency precomputed radiance transfer algorithm. 136
9.4 Rendered images with various configurations of clustered tensor approximation based on the proposed all-frequency precomputed radiance transfer algorithm. . 137
9.5 Comparison between the proposed all-frequency precomputed radiance transfer algorithm and all-frequency clustered principal component analysis. 138
9.6 The proposed all-frequency bi-scale radiance transfer algorithm determines visibility values at the meso-scale. 141
9.7 System diagrams of the proposed all-frequency bi-scale radiance transfer algorithm. 143
9.8 Rendered images with different tensor approximation algorithms for meso-structures based on the proposed all-frequency bi-scale radiance transfer algorithm. 147
9.9 Rendered images with/without indirect illumination based on the proposed all-frequency bi-scale radiance transfer algorithm. 148
9.10 Rendered images based on the proposed all-frequency bi-scale radiance transfer algorithm. 149

List of Tables

7.1 Statistics and timing measurements for bidirectional reflectance distribution functions. 81
7.2 Rendering performance for bidirectional reflectance distribution functions. . . . 82

8.1 Statistics and timing measurements of different tensor approximation algorithms for bidirectional texture functions. 100
8.2 Rendering performance of different tensor approximation algorithms for bidirectional texture functions. 101
8.3 Statistics and timing measurements of the parameterized multivariate spherical radial basis function representation for bidirectional texture functions. 110
8.4 Rendering performance of the parameterized multivariate spherical radial basis function representation for bidirectional texture functions. 111
8.5 Feature comparisons between tensor approximation algorithms and multivariate spherical radial basis functions for bidirectional texture functions. 117
8.6 Statistics and timing measurements of different tensor approximation algorithms for view-dependent occlusion texture functions. 123

9.1 Statistics and timing measurements of the proposed all-frequency precomputed radiance transfer algorithm. 134
9.2 Comparisons between the proposed all-frequency precomputed radiance transfer algorithm and all-frequency clustered principal component analysis. 135
9.3 Feature comparisons of the proposed all-frequency bi-scale radiance transfer algorithm with various meso-scale appearance models. 140
9.4 Statistics and timing measurements of the proposed all-frequency bi-scale radiance transfer algorithm. 146

List of Algorithms

3.1 Fitting algorithm for the scattered univariate spherical radial basis function representation. 28

3.2 Modified soft von Mises-Fisher clustering algorithm for preconditioning. . . . 31

4.1 Fitting algorithm for the scattered multivariate spherical radial basis function representation. 36

4.2 Initial guess for the scattered multivariate spherical radial basis function representation. 38

4.3 Fitting algorithm for the parameterized multivariate spherical radial basis function representation. 41

4.4 Hierarchical fitting algorithm based on the scattered multivariate spherical radial basis function representation. 43

5.1 N-mode singular value decomposition. 49

5.2 Static and iterative clustered tensor approximation. 52

6.1 Iterative K-clustered tensor approximation. 63

List of Symbols

General

$\mathbb{N}$	set of natural numbers
$\mathbb{R}$	set of real numbers
$\mathbb{R}^m$	m-dimensional Euclidean space
$a, b, \ldots$	scalars (italic roman lowercase letters)
$\mathbf{a}, \mathbf{b}, \ldots$	column vectors (boldface roman lowercase letters)
$\mathbf{A}, \mathbf{B}, \ldots$	matrices (boldface roman capitals)
c, i, j, k, n	indices
C, I, J, K, N	maximum values of indices c, i, j, k, n
e	base of the natural logarithm
π	ratio of the circumference of a circle to its diameter
$\lvert\cdot\rvert$	absolute value of a real number or cardinality of a set
$\lVert\cdot\rVert_0$	ℓ^0 norm of a vector
$\lVert\cdot\rVert_2$	ℓ^2 norm of a vector
$\{\cdot\}$	a set
$\setminus$	set minus operator
$\times$	scalar multiplication operator or Cartesian product operator of sets
$\Gamma(\cdot)$	gamma function
$P_n(\cdot)$	normalized Legendre polynomial of degree n so that $P_n(1) = 1$
$\ell_n^{(F)}$	degree-n Legendre coefficient of a function $F(\cdot)$
$\mathcal{I}_n(\cdot)$	order-n modified Bessel function of the first kind
$\mathbf{A}^{-1}$	inverse of a square matrix $\mathbf{A}$
$\mathbf{A}^{+}$	Moore-Penrose pseudo-inverse of a matrix $\mathbf{A}$
$\mathbf{I}_n$	identity matrix of size $n \times n$
$\mathbf{A}^T$	transpose of a matrix $\mathbf{A}$
$(\mathbf{A})_{ij}$	entry in row i and column j of a matrix $\mathbf{A}$
$(\mathbf{A})_{i*}$	i-th row of a matrix $\mathbf{A}$
$(\mathbf{A})_{*j}$	j-th column of a matrix $\mathbf{A}$

Spherical Radial Basis Functions

$\mathbb{S}^m$	unit hyper-sphere embedded in $\mathbb{R}^{m+1}$ (or unit m-sphere)
ϕ	geodesic distance on the unit hyper-sphere
ω	a point on the unit hyper-sphere
β	basis coefficient of a spherical radial basis function
ξ	center of a spherical radial basis function
λ	bandwidth of a spherical radial basis function
θ	a parameterization coefficient
$\omega \cdot \xi$	dot product of ω and ξ
$\psi(\cdot)$	a parameterization function
Ω	a set of points on the unit hyper-sphere(s)
Ξ	a set of spherical radial basis function centers
Λ	a set of spherical radial basis function bandwidths
Θ	a set of parameterization coefficients
Ψ	a set of parameterization functions
$F(\cdot)$	a univariate/multivariate spherical function
$G(\cdot)$	a spherical radial basis function
$\star_m$	spherical singular integral operator over $\mathbb{S}^m$
$\Omega(m)$	total surface area of $\mathbb{S}^m$
τ_G	center of gravity of a spherical radial basis function $G(\cdot)$
σ_G	variance of a spherical radial basis function $G(\cdot)$

Tensor Algebra

$\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \ldots$	tensors (boldface calligraphic capitals)
$\lVert\cdot\rVert_F$	Frobenius norm of a tensor (or a matrix)
$(\boldsymbol{\mathcal{A}})_{i_1 i_2 \cdots i_N}$	entry of an N-th order tensor $\boldsymbol{\mathcal{A}}$ indexed by $i_1, i_2, \ldots, i_N$
$\boldsymbol{\mathcal{A}}_{\langle n_i \rangle}$	the i-th mode-n sub-tensor of a tensor $\boldsymbol{\mathcal{A}}$
$\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle$	scalar product of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$
$uf_n(\cdot)$	mode-n unfolded matrix of a tensor
$\times_n$	mode-n product operator
$\otimes$	Kronecker product operator
$\bigtimes_n$	operator of a series of mode-n products
$\bigotimes$	operator of a series of Kronecker products
R_n	a mode-n reduced rank
$\boldsymbol{\mathcal{Z}}$	a core tensor
$\mathbf{U}_n$	a mode-n basis matrix
$\mathbf{V}_n$	a dual mode-n basis matrix

Applications

ω_l	an illumination direction on $\mathbb{S}^2$
ω_v	a view direction on $\mathbb{S}^2$
$L(\omega_l)$	a high dynamic range environment map
$\rho(\omega_l, \omega_v)$	a bidirectional reflectance distribution function
$\mathbf{p}$	a pixel or a point on an object surface
$\mathbf{n_p}$	surface normal at point $\mathbf{p}$
$L_{o,\mathbf{p}}(\omega_v)$	exitant radiance in view direction ω_v at point $\mathbf{p}$
$L_{in}(\omega_l)$	incident radiance in illumination direction ω_l
$V_\mathbf{p}(\omega_l)$	visibility function at point $\mathbf{p}$ in illumination direction ω_l
x	horizontal coordinate in the two-dimensional Cartesian coordinate system
y	vertical coordinate in the two-dimensional Cartesian coordinate system
$\mathbf{t} = [x, y]^T$	two-dimensional spatial coordinates, x and y, of a texel
$B(\omega_l, \omega_v, \mathbf{t})$	a bidirectional texture function
$O(\omega_v, \mathbf{t})$	a view-dependent occlusion texture function
S	a spatial coordinate texture for spatially-varying materials

Chapter 1

Introduction

1.1 Background

Synthesizing photo-realistic images is a significant and ambitious goal in computer graphics and vision. It has not only been extensively studied in the academic community, but also become ubiquitous in our daily life. Special visual effects and vivid animations are nowadays essential to many computer applications as well as film production, and could even be found on various consumer electronic devices. In real-time rendering applications, visual realism has always been a more challenging problem, since realistic image synthesis is not the only concern, but the human-computer interaction is also of great importance. The difficulty lies in that the interactivity usually dominates other objectives, hence the desired visual effects can only be realized with limited computational power.

Researchers have conventionally focused on developing analytic models and simulation-based algorithms to achieve photo-realistic image synthesis in real time. Nevertheless, real-world object shape, surface reflectance, micro-scale appearance, and natural illumination effects are frequently too complicated to be synthesized using analytic models or simple simulations. Over the last decades, there have been tremendous advances in this field. First, the emergence of programmable *graphics processing units* (GPUs) [103, 141] and *general-purpose computation on GPUs* (GP-GPUs) [131, 157] introduced flexible programming models whose compiled code could be executed and accelerated by graphics hardware. The fast pace of progress in the computational power of GPUs not only enables many rendering methods that were previously impractical to operate in real time, but also has stimulated the development of highly parallel algorithms in many scientific communities beyond computer graphics and vision.

Second, while traditional rendering algorithms generate images using computationally expensive procedures, state-of-the-art methods, which are known as *data-driven rendering* or *inverse rendering*, handle the same problem from a different standpoint – if we can not afford the computational costs of complex procedures at run-time, why not perform rendering from *cached* or *pre-sampled* data that represent the results of complex procedures or even real-world measurement? A common example is the *bidirectional reflectance distribution functions* (BRDFs) for local illumination reflection. In data-driven rendering, the whole process begins by measuring data samples from real-world object surfaces [113, 114]. The data collections are then modeled with an appropriate representation for fast run-time rendering from novel illumination and view directions. This simple idea obviously reverses the conventional 'forward' rendering process that develops a representation from theories or heuristics, and also ushers in a new era for real-time rendering.

1.2 Motivations and Methods

Although data-driven rendering avoids computationally expensive procedures at run-time, they are usually subject to cumbersome pre-sampled observations that consume a large amount of storage space and memory bandwidth. As the demand of more and more photo-realistic computer-generated images increases, this problem has become even worse since we have to record more information to account for more degrees of freedom and more detailed descriptions of the desired visual effects. Nowadays, the amount of pre-sampled data often exceeds tens or hundreds of gigabytes, and tends to increase in the future. Since memory bandwidth is a major bottleneck of GPUs in modern real-time graphics and vision applications, the performance of data-driven rendering may be, in the worst case, downgraded to be slower than directly employing the complex procedures.

To solve this problem, researchers have been devoted to developing effective representations and approximation algorithms for handling the pre-sampled observations. While an expressive representation provides an efficient and meaningful description of data, a powerful compression method results in compact storage space and fast data reconstruction on GPUs. Tight cooperation between them may be a remedy for real-time photo-realistic image synthesis. In this dissertation, we thus focus on data representations and approximation algorithms for real-time rendering of visual data sets. Related topics and methods are extensively surveyed and studied. We also discuss how to achieve an appropriate trade-off among computational costs, approximation errors, and storage space for real-time applications. Our investigation leads to the following methods:

Data Representations

We introduce two parametric representations for radiance and illumination functions, which are named univariate and multivariate *spherical radial basis functions* (SRBFs). A univariate SRBF is a circularly axis-symmetric and rotation-invariant function defined on the unit hypersphere. Radiance functions thus can be modeled in their intrinsic spherical domain, rather than a cubic or planar domain as in previous articles [22, 125]. This can avoid false boundaries and distortions that may result from improper resampling strategies in a non-intrinsic domain. The spatial localization property of univariate SRBFs especially allows the high-frequency signals within local regions to be handled efficiently. On the other hand, a multivariate SRBF is a function constructed by the product of multiple univariate SRBFs. This breaks the limitation of univariate SRBFs, as they only account for a single variable on the unit hyper-sphere. Since the surface appearance of a real-world object is frequently an effect of different physical factors, multivariate SRBFs are particularly suitable for modeling the complex behaviors of measured reflectance data. Moreover, we also demonstrate that a linear combination of either univariate or multivariate SRBFs can generalize SRBFs into a powerful analysis tool for heterogenous radiance functions.

To obtain a compact representation, it is also well-known that transforming the parameters of a multivariate function into another parametric space, which we refer to as *parameterization*, can improve approximation efficiency [77, 90, 151, 213, 214]. However, previous articles have considered only heuristic or fixed transformation functions, little attention have been paid to a data-dependent parameterization method. Therefore, we propose to learn a set of optimized parameterization functions for a given visual data set to overcome the main disadvantage of conventional fixed parameterization. By using a parametric representation to model the transformation functions, the parameterization process can be tightly integrated into the proposed multivariate SRBF representation. Previous heuristic parameterizations, such as the half-way vector for reflectance functions, thus become special cases in our general framework.

Approximation Algorithms

For data compression, we propose two dimensionality reduction algorithms, namely *clustered tensor approximation* (CTA) and *K-clustered tensor approximation* (K-CTA), for large-scale multi-dimensional visual data sets. A major drawback of traditional tensor approximation (also called multi-linear models or multi-way analysis) [33, 165] is that reconstruction costs may be still too high for real-time rendering applications when the data variations in the input tensor are large [185, 192]. The proposed CTA algorithm therefore aims at overcoming this drawback by classifying the input tensor into disjoint clusters, so that the member data within each cluster form another tensor with much lower variations and can be more efficiently decomposed using tensor approximation. An iterative technique for updating cluster members is also introduced to derive a locally optimal solution for CTA.

Based on CTA, we further develop K-CTA to classify the input tensor into 'overlapped' regions so that inter-cluster coherence can be exploited by mixing the decomposed results of more than one cluster. Since the maximum number of mixture clusters is guaranteed in the proposed model, K-CTA provides a sparse multi-linear representation in which the sparsity is totally under user control. This especially leads to predictable run-time reconstruction costs and an easy-to-optimize shader program on GPUs.

1.3 Applications of Data-Driven Rendering

To demonstrate the effectiveness of our methods, intensive experiments have been conducted on various applications of data-driven rendering and visual data sets in computer graphics and vision. They are summarized as follows:

Illumination and Reflectance Functions

Lighting and shading effects are considered as one of the main contributing factors in human visual perception. A large number of attempts have been made to develop realistic illumination and reflectance models in computer graphics and vision. In our experiments, two common kinds of illumination and reflectance data, including *high dynamic range* (HDR) environment maps [34, 36] and BRDFs, are modeled with univariate or multivariate SRBFs. One of the major advantages of SRBFs is their 'non-uniformity'. Specifically, SRBFs can be non-uniformly distributed based on the input observations to obtain a compact representation for the original function. This non-uniformity particularly captures the irregularity and high-frequency signals of real-world illumination and reflectance functions, which are difficult to be modeled efficiently using conventional data representations, for example, spherical harmonics and wavelets.

Furthermore, ray tracing is a promising framework that allows photo-realistic image synthesis. However, it frequently relies on time-consuming Monte Carlo sampling to simulate numerous illumination effects. We thus focus on the direct illumination computation in ray tracing, and demonstrate that the univariate SRBF representation for BRDFs can be seamlessly integrated with existing Monte Carlo sampling methods. The proposed importance-driven method especially exploits the non-uniformity of univariate SRBFs to efficiently generate sampling directions from the distributions of BRDFs.

Spatially-Varying Appearance Models

Meso-scale surface appearance is often difficult to be faithfully captured using analytic models. The shape details and spatially-varying reflectance distributions owing to complex meso-structures may need millions of polygons to model. Traditional *bidirectional texture functions* (BTFs) [28] only consider meso-scale illumination effects, but completely ignore the micro-geometry of object surfaces. We thus introduce a new spatially-varying appearance model, which is named *view-dependent occlusion texture function* (VOTF), to render view-dependent meso-scale occlusions from precomputed visibility information. The proposed VOTFs can be easily implemented on GPUs and further combined with existing appearance models for visualizing meso-scale silhouettes.

In addition, modeling spatially-varying surface appearance is a more challenging problem than approximating illumination and reflectance functions, since the large amount of appearance data frequently prevents sophisticated data representations and approximation algorithms. In this dissertation, we specifically demonstrate the experimental results of two different approximation approaches for spatially-varying appearance models: one is a non-parametric multi-linear model based on CTA or K-CTA; the other is a parametric representation based on multi-variate SRBFs and optimized parameterization.

First, the multi-linear framework provides a computationally efficient approximation scheme for large-scale multi-dimensional appearance models. By organizing an appearance data set as a high-order tensor, both CTA and K-CTA allow a compact representation for meso-structures and efficient run-time reconstruction on GPUs with just a few low-order factors and reduced multi-dimensional core tensors. Second, by using the proposed multivariate SRBF representation, we show that BTFs can be accurately approximated and efficiently rendered on GPUs. A hierarchical fitting algorithm is developed to exploit the spatial coherence in BTFs and reduce the computational costs of non-linear optimization.

Radiance Transfer

Rendering global illumination effects is one of the most computationally expensive problems in computer graphics. The light transport is a highly coupled effect of scene geometry, lighting environments, and object materials, such that its real-time simulation is difficult to achieve. Our experimental results demonstrate that based on *precomputed radiance transfer* (PRT) [125, 161, 162] and *bi-scale radiance transfer* (BRT) [163], real-time global illumination effects for a static scene in dynamic lighting environments can be easily realized by combining the univariate SRBF representation and CTA (or K-CTA). Furthermore, all-frequency signals in the rendered images, such as sharp shadow boundaries and specular reflections, also can be well-preserved with high compression ratios.

1.4 Contributions

In brief, this dissertation makes the following contributions to data representations and approximation algorithms for real-time data-driven rendering:

Data Representations

- A compact parametric representation for univariate spherical functions based on a linear combination of non-uniformly distributed univariate SRBFs is proposed. It particularly allows efficient spherical integrals and reconstruction on GPUs.

- A novel parametric representation based on a linear combination of multivariate SRBFs is introduced to efficiently model multivariate spherical functions that result from complex effects of various physical factors.

- An parameter transformation method based on parametric models is presented to automatically derive the optimal parameterization functions from a given visual data set. It can seamlessly cooperate with the proposed multivariate SRBF representation to improve approximation efficiency for multivariate spherical functions.

- The proposed univariate and multivariate SRBF representations as well as optimized parameterization are hardware-friendly and easy-to-implement on modern GPUs. Since they are all continuous functions, no additional interpolation or filtering techniques are required for smooth run-time rendering.

Approximation Algorithms

- A new algorithm for analyzing large-scale multi-dimensional visual data sets, namely CTA, is proposed. It iteratively re-classifies the input tensor into disjoint regions, so that the member data within each cluster can be more efficiently decomposed using traditional tensor approximation.

- A novel multi-linear model, which is called K-CTA, is introduced to permit accurate and compact approximation of large-scale multi-dimensional visual data sets. K-CTA not only extends CTA to exploit intra- and inter-cluster coherence at the same time, but also provides a sparse multi-linear representation in which the sparsity is totally under user control.

- Both CTA and K-CTA are efficient and GPU-friendly for run-time reconstruction. They can be further combined with other data representations and approximation algorithms to derive a more powerful model for real-time data-driven rendering.

Applications

- Novel approaches for modeling illumination and reflectance functions, including HDR environment maps and BRDFs, with non-uniform univariate or multivariate SRBFs are demonstrated. Although they are based on time-consuming non-linear optimization, GPUs can be employed to reduce the processing time from hours to minutes.

- The proposed univariate SRBF representation for reflectance functions is also promising for the importance sampling of direct illumination in ray tracing. By exploiting the rotational invariance of the univariate SRBF representation, the distribution of a BRDF can be efficiently analyzed to determine sampling directions for Monte Carlo integration.

- A new spatially-varying appearance model, namely the VOTF, is introduced to enable per-pixel rendering of view-dependent occlusions without actually modifying the geometric shape of objects at the meso-scale.

- A multi-linear framework for compressing spatially-varying appearance materials, such as BTFs and VOTFs, by using CTA or K-CTA is presented. It can be easily integrated into other data representations and approximation algorithms for efficient run-time rendering.
- A view-dependent signed-distance representation for VOTFs is introduced to preserve sharp features at the meso-scale even after approximation using multi-linear models.
- An extended texture synthesis method, based on the decomposed results of CTA or K-CTA, is demonstrated to account for the synthesis of view-dependent meso-structures over arbitrary surfaces.
- A hierarchical fitting algorithm for BTFs based on multivariate SRBFs and optimized parameterization is proposed to exploit the spatial coherence in visual data sets. It significantly accelerates the approximation process and is particularly suitable for multi-resolution data analysis and practical rendering applications owing to the inherent mipmap pyramid construction.
- A novel all-frequency PRT framework based on univariate SRBFs and CTA (or K-CTA) is introduced. It permits a compact data representation and real-time rendering of complex objects with global illumination effects in high-frequency lighting environments. The PRT application especially demonstrates the flexibility and potential of univariate SRBFs and CTA (or K-CTA). Since they do not conflict with each other, their combination is possible to provide a more powerful analysis tool for visual data sets.
- An all-frequency BRT algorithm is proposed to allow real-time rendering of objects with spatially-varying surface appearance and various global illumination effects on GPUs.

1.5 Dissertation Organization

The remainder of this dissertation is organized as follows. **Chapter 2** reviews the literature on data representations, dimensionality reduction, and data-driven rendering in computer graphics and vision. Next, **Chapters 3** and **4** introduce two parametric data representations, univariate and multivariate spherical radial basis functions, to model spherical radiance and illumination information. **Chapter 4** also describes the main concept of optimized parameterization that can be applied to improve approximation efficiency for modeling multivariate functions on the unit hyper-sphere. **Chapters 5** and **6** then develop two novel dimensionality reduction algorithms, namely clustered tensor approximation and K-clustered tensor approximation, to compress large-scale multi-dimensional visual data sets.

In addition, applications and experimental results of the proposed methods are demonstrated in **Chapters 7–9**. We choose various representative applications and visual data sets in computer graphics and vision, including high dynamic range environment maps (**Section 7.1**), bidirectional reflectance distribution functions (**Section 7.2**), importance sampling of direct illumination in ray tracing (**Section 7.3**), bidirectional texture functions (**Section 8.1**), view-dependent occlusion texture functions (**Section 8.2**), precomputed radiance transfer (**Section 9.1**), and bi-scale radiance transfer (**Section 9.2**). Finally, **Chapter 10** discusses the conclusions of this dissertation and sheds lights on future research directions.

Chapter 2

Literature Review

2.1 Data Representations

The data representation is a fundamental element of algorithms which can not be simply ignored or depreciated. It significantly affects the way in which we handle problems. Since various representations have entirely different advantages and disadvantages, there is no general solution to all problems. To derive a satisfactory representation, we need to analyze and exploit the special attributes of the given problem. The following three sections thus briefly review contemporary functional models (Section 2.1.1), spherical radial basis functions (Section 2.1.2), and parameterization methods (Section 2.1.3) that have been widely adopted for data representations in computer graphics and vision.

2.1.1 Functional Models

Scientists frequently rely on finite representation sets to establish an effective basis for further data analysis or processing. In computer graphics and vision, common forms of the finite sets may include:

- **Primitive elements:** Points, piecewise elements, solid primitives, and so forth.
- **Parametric splines:** Parametric curves, surfaces, volumes, and so forth.
- **Topological structures:** Meshes, graphs, hierarchical structures, and so forth.
- **Functional models:** Harmonic functions, wavelets, radial basis functions, and so forth.

Functional models have recently received great attention due to their efficiency and flexibility. In this category, data sets are analyzed and transformed into the space spanned by a finite set of basis functions. The combination of these functions thus 'implicitly' describes or approximates the raw data. Since this function set is often compact and sparse, the computational complexity of data analysis can be greatly reduced. Furthermore, the flexible ability of handling different problems with appropriate functions is also an appealing property. Applications of functional models in computer graphics and vision include image-based modeling and rendering [113, 119, 203, 213, 214], point-based graphics [122, 134, 136, 152], photo-realistic illumination computation [22, 49, 50, 125, 162], multi-resolution techniques [39, 106, 168], and to name a few.

Harmonic functions are important in physics, applied mathematics, and engineering communities. In computer graphics and vision, one major harmonic function, *spherical harmonics*, has been employed to represent illumination and reflectance functions for efficient rendering, for example, scattering functions [74], reflectance distributions [143, 145, 146, 160, 201], and environment maps [144, 145]. Light transport data sets have also been modeled with spherical harmonics to render self-shadowing, self-inter-reflection, subsurface scattering, and caustics effects [99, 161, 162]. Similar to Fourier series, spherical harmonics form a complete set of orthonormal basis functions on the unit sphere, so that a square-integrable univariate spherical function can be decomposed as a linear combination of the spherical harmonic basis. In addition to spherical harmonics, Ren et al. [149] and Sloan et al. [164] further employed *zonal harmonics*, which are special cases of spherical harmonics, to account for the radiance transfer observations of dynamic scenes and deformable objects.

Wavelets possess the characteristics of multi-resolution decompositions and hierarchical structures to provide an adaptive basis for data analysis. Although originated in applied mathematics and signal processing, wavelets have a great impact on several topics of computer graphics and vision [168], including image processing [12, 66], geometric modeling [39, 43, 106], global illumination [48], and so forth. As data-driven rendering becomes prevalent, wavelets have been applied to derive a compact representation for precomputed radiance transfer [125, 126, 170, 195, 196] and measured reflectance data sets [22, 87, 113].

Radial basis functions (RBFs) are also among the most popular basis functions for data representations except harmonic functions and wavelets. A RBF is a kernel function that depends on the 'distance' with respect to its center. The simplicity and powerful capability of RBFs have led to many applications in computer graphics and vision, such as geometric modeling [17, 38, 182], point-based rendering [136, 152, 215], point-based animation [122, 134], and so forth. For data-driven rendering, spatially-varying reflectance functions [213, 214] and view-dependent light transport [49, 50] have also been approximated with RBFs.

In this dissertation, we seek to develop an appropriate data representation for modeling radiance and illumination functions. Since radiance data are highly related to directions on the unit hyper-sphere, it would be better to preserve their intrinsic nature in the spherical domain. Additionally, this representation should be compact, rotation-invariant, and efficient in computing the spherical integral of the rendering equation [73]. In contrast to previous approaches, we introduce univariate and multivariate SRBFs that satisfy the desired objectives.

2.1.2 Spherical Radial Basis Functions

Spherical radial basis functions (also called *spherical basis functions* in [123]) are special RBFs defined on the unit hyper-sphere. Their intrinsic nature in the spherical domain and other appealing properties, such as rotational invariance and positive definiteness, make them appropriate for modeling spherical functions without introducing any artificial boundaries or distortions. When combined with multi-resolution approaches, such as spherical wavelets [44, 123], SRBFs become a powerful mathematical tool for analyzing scattered data on the unit hyper-sphere, including information measured by satellites [41] and observed stations on the entire globe [129].

In fact, computer graphics and vision researchers are not unfamiliar with SRBFs. The generalized cosine lobe [86] and the isotropic Gaussian kernel of Ward model [198] are two special cases of SRBFs. In both articles, SRBFs were utilized to model *bidirectional reflectance distribution functions* (BRDFs), and usually lead to a compact, expressive, and physically plausible representation for reflectance functions. Green et al. [49, 50] also employed spherical Gaussian functions, which can be viewed as a variant of SRBFs, to model all-frequency glossy and mirroring effects with self-occlusions.

In this dissertation, we apply traditional SRBFs, which are referred to as *univariate* SRBFs, to model radiance and illumination functions, and propose a novel category of SRBFs, namely *multivariate* SRBFs, to overcome the major drawbacks of univariate SRBFs. We also present practical fitting algorithms based on non-linear optimization to estimate the parameters of univariate/multivariate SRBF representations. Besides, it is worth noted that a special form of univariate SRBFs is highly related to the well-known von Mises-Fisher distribution [112] in statistics and various computer science applications, including document clustering [6], orientation distribution approximation [55, 117], and illumination estimation [56]. We will discuss this relation in more details in Section 3.2.3.

2.1.3 Parameterization

For data representations, identifying or deciding the parameters of a model, which is often called parameterization, is an essential pre-process. The identified parameters may be of particular interests for a specific application, or define a proper coordinate system for the given observations. An intuitive parameterization approach is to employ the same coordinate system during data acquisition, where each parameter corresponds to a single acquisition condition. Nevertheless, there are two problems with this approach. First, we may not be able to identify and control all the conditions that influence the measured data. Second, even all of them are controllable during acquisition, they may not form an effective coordinate system for the observations. A common solution to the above issues is to transform the acquisition conditions into another set of more powerful parameters. The overall effect thus is equivalent to representing the observations with a suitable coordinate system.

In computer graphics and vision, half-way [151] and reflected vector [145] parameterizations for BRDFs have been shown to be effective in modeling highly specular materials. Stark et al. [167] also proposed several physically-interpretable methods for isotropic BRDFs based on coordinate systems with triangular supports. In recent years, various heuristic parameterizations have been applied to approximate spatially-varying surface appearance [90, 213, 214] and radiance transfer functions [49], further demonstrating the promising potentials of parameterization. In general, parameterization is beneficial to reduce the dimensionality of reflectance functions, which leads to a compact and low-dimensional representation for surface appearance. It also greatly increases data coherence that can be subsequently exploited by different approximation algorithms.

Nevertheless, previous parameterization approaches are limited to fixed or heuristic transformation functions. Without intensive experiments, it is difficult to predict which parameter transformation performs the best for the given data sets. By contrast, we propose to learn the optimal parameterization functions from data, within restricted functional forms, by introducing additional parameters in the proposed multivariate SRBF representation. The resulting model thus successfully combines parameterization and multivariate SRBF approximation in a unified framework to fill the gap between these two problems that have been solved separately in conventional data representations.

2.2 Dimensionality Reduction

The high-dimensional 'curse' has driven the advances of dimensionality reduction techniques for a long time. Scientists generally assume that low-dimensional manifolds are embedded in their high-dimensional observations, and could be estimated by linear or non-linear transformations. The result of this estimation should preserve the structures of the embedded low-dimensional manifolds, and therefore implies an appropriate approximation (or representation) of the high-dimensional observations. In the following sections, we briefly review three categories of dimensionality reduction algorithms: linear (Section 2.2.1), non-linear (Section 2.2.2), as well as multi-linear (Section 2.2.3) models, and summarize their applications in computer graphics and vision.

2.2.1 Linear Models

Dimensionality reduction algorithms have been widely adopted to analyze and compress various types of data sets. Perhaps the most popular approach is *principal component analysis* (PCA) [72], which is a linear model and often computed using *singular value decomposition* (SVD). In PCA, data samples are transformed from a high-dimensional space into another low-dimensional sub-space spanned by only a few *principal components* (PCs). The original samples thus can be approximated by projecting them onto these PCs. Variants of PCA have also been proposed to incorporate special considerations in practical applications, including non-negative matrix factorization [93, 94, 132] and non-negative sparse PCA [63, 210]. Nevertheless, a major drawback of PCA, together with its variants, is that observations must be re-arranged into a standard two-mode matrix before analysis. In real cases, data are frequently sampled under different conditions, and can be naturally classified into more than two modes. The original structures and important information of data sets may be lost after the re-arrangement.

In addition to traditional PCA, some linear models, such as probabilistic PCA [174, 175] and Bayesian PCA [13], derive PCs and projection coefficients in the linear sub-space of observations from a probabilistic standpoint. These linear models thus can be made more tolerant to noise and outliers than conventional PCA, and easily integrated with other probabilistic methods, for example, the mixture of Gaussians [13, 174]. There are also other popular linear models, for instance, factor analysis [8], classical multi-dimensional scaling [26], independent component analysis [64], and to name a few. Comprehensive surveys of linear dimensionality reduction algorithms can be found in [7] and references therein.

In general, linear models are computationally efficient and easy to implement, but they are inadequate to analyze data sets with non-linear structures. Due to the discrepancy between the model assumption and the intrinsic non-linear nature of some real-world data sets, we may obtain fallacious results from linear analysis. In computer graphics and vision, applications of linear models include lighting information approximation [20, 67, 99, 113, 114], spatially-varying appearance models [153, 194], texture synthesis [96, 97], and motion tracking or estimation [104, 127].

2.2.2 Non-Linear Models

Apart from linear models, numerous dimensionality reduction algorithms have been proposed to explore non-linear correlations among data. Although the structures of real-world observations are complicated and globally non-linear, local PCA [75] assumed that local correlations are approximately linear, so that the observations could be 'locally' transformed into low-dimensional linear sub-spaces without a significant loss of information. Locally linear embedding [150, 154] and Laplacian eigenmaps [9] aimed at preserving the local geometry of the embedded manifolds by transforming nearby data samples to nearby points in the low-dimensional sub-space. Isomap [173] instead attempted to preserve both local and global structures of the embedded manifolds, in terms of geodesic distance, when transforming the observations into the low-dimensional sub-space. Moreover, kernel PCA [155] took higher-order statistics among data samples into account by performing PCA in the reproducing kernel Hilbert space. If a proper set of kernel functions is employed, the non-linear structures of the embedded manifolds in the original space may be mapped into linear structures in the reproducing kernel Hilbert space. Generalized PCA [188] further extended this concept by estimating an unknown number of sub-spaces from the observations, where the identified sub-spaces could be modeled with a set of homogeneous polynomials whose degree is the number of subspaces. It thus overcomes one of the major drawbacks of kernel PCA in which learning appropriate kernels is still an open problem.

There are also other non-linear models, for example, auto-associative neural networks [82], generative topographic mapping [14], self-organizing maps [80], permuted PCA [68], Gaussian process latent variable model [92], and to name a few. Their detailed descriptions and comparisons are beyond the scope of this dissertation. Interested readers may refer to [95] for a comprehensive review of non-linear dimensionality reduction algorithms.

Although non-linear models allow complex analysis on observations, not all of them are generative. Data reconstruction may not be possible from the derived results. This frequently prevents some of them from practical applications of data-driven rendering. Additionally, non-linear models are computationally more expensive than linear models, and sometimes may be intractable for large-scale data sets. Despite these disadvantages, applications of non-linear models are still prevalent in computer graphics and vision, for instance, precomputed radiance transfer [161], texture synthesis [97], and material modeling [113, 114, 193].

2.2.3 Multi-Linear Models

In recent years, multi-linear models (also called tensor approximation or multi-way analysis) [33, 165] have become widespread and caught a lot of attention. They can be regarded as a generalization of PCA, where data samples are processed in their intrinsic form as a multi-dimensional array, and separate reduction is allowed along each dimension. Unlike linear and non-linear models, tensor approximation relies on decomposing a high-dimensional space into multiple low-dimensional sub-spaces that are respectively associated with each mode of observations to remove the curse of dimensionality. The extracted low-order factors in each sub-space then can be combined to effectively model the original high-dimensional space. In this way, multi-linear models successfully preserve the intrinsic structures and important information of observations, and thus overwhelm one of the main disadvantages of previous dimensionality reduction techniques. Two primary categories of traditional multi-linear models are Tucker models [180, 181] and parallel factor analysis [57] (or canonical decomposition [18]). Both of them play important roles in chemometrics and psychometrics, and have recently become prevalent in signal processing [78] and computer science [1].

In computer graphics and vision, multi-linear models have also been successfully extended [159, 204, 205, 207] and applied to various applications, such as data-driven rendering [171, 185, 192], human facial processing [183, 184, 189, 190, 191], and image analysis [59, 60]. Even some matrix factorization methods [90, 124, 172, 208, 209] are implicitly related to multi-linear models. Vasilescu and Terzopoulos [185] organized a bidirectional texture function as a high-order tensor and applied tensor approximation to decompose it. Wang et al. [192] introduced an out-of-core and block-wise tensor approximation technique based on N-mode SVD [33]. Xu et al. [205] instead focused on the least-squares reconstruction, from the same sub-spaces, for a set of objects represented as high-order tensors. Shashua and Hazan [159] extended non-negative matrix factorization to derive positive and sparse factors from a general multi-dimensional array. Sun et al. [171] proposed a tensor representation to model the light transport of inter-reflections for dynamic BRDFs. Yan et al. [207] suggested re-arranging each element of a tensor so that data coherence is maximized for tensor-based subspace learning. Furthermore, Wu et al. [204] presented a hierarchical tensor decomposition model to expose the multi-scale structures among multi-dimensional visual data sets. This thus successfully incorporated multi-resolution analysis with existing framework of tensor approximation.

Although multi-linear models allow higher compression ratios than PCA, directly employing them to data-driven rendering would be inadequate for real-time applications. Even after applying the popular N-mode SVD algorithm [33] to derive an optimal approximation of the input tensor, the amount of compressed data is still cumbersome. It is also difficult to achieve real-time performance for run-time rendering or analysis in computer graphics and vision applications. In this dissertation, we therefore propose two novel multi-linear models, clustered tensor approximation and K-clustered tensor approximation, to solve these problems. The proposed methods rely on combining the merits of clustering and tensor approximation to form new mathematical tools for data analysis, and can be regarded as integrating the concept of some non-linear models into multi-linear models.

2.3 Data-Driven Rendering

Rendering from data has a long history in computer graphics and vision. Maybe the most well-known method is texture mapping. Recently, data-driven rendering approaches have enhanced and glorified this concept to synthesize photo-realistic images from large-scale measured or precomputed visual data sets. In this way, the output quality and rendering performance of data-driven rendering are independent of scene complexity, but depend on the acquisition, encoding, and decoding schemes of the pre-sampled data. Numerous variants and extensions of data-driven rendering have been proposed over the last decade. The light fields [101] and the lumigraphs [47] employed densely-sampled images to render novel views from arbitrary camera positions without additional information. Shade et al. [158] proposed to synthesize the requested view from a layered depth image set that stores depth information of a scene. Dana et al. [28] proposed *bidirectional texture functions* (BTFs) that combine BRDFs and texture maps to account for the illumination- and view-dependent variations in texels. The progress of sensor technologies has also stimulated the development of many data-driven rendering techniques, such as material modeling [21, 113, 114, 135], time-varying appearance [51, 58, 169, 193], and so forth.

Two primary challenges of data-driven rendering are the accurate measurement of natural phenomena and the efficient manipulation of observations. Nowadays, the acquisition, representation, and compression of visual data sets for complicated phenomena become even more difficult to be handled. The purpose of this dissertation is to overcome the latter challenge of data-driven rendering by applying effective data representations and powerful approximation algorithms to measured observations. Although there are numerous related articles, we only briefly review spatially-varying appearance models (Section 2.3.1) and radiance transfer (Section 2.3.2) techniques that are directly related to this dissertation.

2.3.1 Spatially-Varying Appearance Models

Appearance Models

For detailed surface appearance, BTFs [28] extended texture maps and BRDFs to describe spatially-varying local reflectance distributions owing to micro-geometry that may need millions of polygons to model. Thus, a BTF is a six-dimensional function that can be regarded as a two-dimensional texture in which each texel records the exitant radiance of different view directions with respect to incident illumination variations[1]. Koudelka et al. [81] and Sattler et al. [153] also discussed the acquisition, compression, and rendering issues of BTFs and made their measured BTF databases publicly available. Although BTFs effectively model complex meso-scale reflectance behaviors with multiple images, they are still restricted to local illumination responses.

Apart from BTFs, various appearance models have also been developed to render shadows and complex illumination effects from precomputed meso-scale data or special fields, such as visibility information [30, 61], view-dependent displacement mapping [194, 197], shell texture functions [21], and relief mapping [130, 138, 139]. However, even if these methods permit efficient representations for complicated surface appearance and meso-structures, inter-reflections can not be rendered at real-time rates.

Approximation Algorithms

For the approximation of appearance models, we summarize three main categories of algorithms: functional linear models, non-parametric models, and probabilistic models.

Functional linear models are of the most popular representations for data-driven rendering. Their main concept is to approximate a complex appearance function as a linear combination of simple basis functions. In this category, the choice of an appropriate basis is one of the major research issues, which significantly influences approximation efficiency and quality. Previous articles have proposed to model various appearance functions using parametric kernels [46, 86, 115, 198], polynomials [111, 118], radial basis functions [213, 214], spherical harmonics [145, 163], and wavelets [107, 156]. For all-frequency appearance data sets, parametric kernels and radial basis functions generally provide the best trade-off between rendering per-

[1]For a single texel, the actual intersection point on meso-structures may be different from each view direction.

formance and image quality. Nevertheless, they usually rely on time-consuming non-linear optimization to derive model parameters, and thus are impractical for approximating spatially-varying materials.

Non-parametric models can be regarded as functional models that do not have pre-defined forms of basis functions. In this category, an appropriate basis is learnt from appearance data sets, rather than the prior information specified by researchers. The most popular approaches in computer graphics and vision include clustering and dimensionality reduction techniques, such as variants of principal component analysis [27, 77, 81, 121, 153], matrix factorization [89, 90, 116, 135, 172], tensor approximation [185, 192], and vector quantization [42, 100]. Although non-parametric models are entirely data-dependent methods that provide accurate and flexible representations for appearance functions, the amount of compressed data and run-time rendering costs are still high when compared to other categories of approximation algorithms. Additionally, special interpolation or estimation techniques are frequently required to synthesize surface appearance from novel illumination and view directions.

In **probabilistic models**, spatial correlations in appearance data are described with probability density functions, so that similar materials can be synthesized from the estimated parameters of distributions and noise maps [52, 54]. Recently, Haindl and Filip [53] proposed a multi-scale probabilistic BTF model based on the casual autoregressive random field, and further combined range maps to enhance the surface roughness of rendered objects. Although probabilistic models can achieve very high compression ratios, their main goal is efficient and seamless appearance synthesis, not an optimal reconstruction of the original appearance data. Additionally, the run-time rendering process is rather slow and currently not GPU-friendly.

2.3.2 Radiance Transfer

Precomputed Radiance Transfer

Recently, *precomputed radiance transfer* (PRT) has received a growing interest owing to its ability of rendering complex illumination and shadowing effects, such as self-inter-reflections, sub-surface scattering, caustics, and self-shadows, in dynamic lighting environments at real-time rates. The key concept is to precompute and model light scattering between an object and its surroundings by representing both incident radiance and light transport functions in the spherical harmonic basis. Thus, run-time rendering of exitant radiance can be reduced to a simple dot product for a diffuse object, or a matrix-vector multiplication for a glossy one. Since the spherical harmonic basis is inadequate to approximate high-frequency signals, PRT methods based on spherical harmonics are also called *low-frequency* PRT [99, 161, 162].

Beyond low-frequency radiance transfer, the *all-frequency* PRT methods [105, 125, 126, 195, 196] pre-sampled high-resolution light transport data to accurately capture hard and soft shadows in all-frequency lighting environments. The densely-sampled PRT data were then compressed using sophisticated compression techniques, for instance, non-linear wavelet approximation [125, 126, 195, 196] and matrix factorization [105, 195, 196]. Green et al. [50] further introduced a hybrid PRT method for static scenes. While view-independent effects, including direct and indirect diffuse terms, were modeled with spherical harmonics or wavelets, high-frequency view-dependent signals, such as direct and indirect glossy terms, were approximated with Gaussian functions using non-linear optimization. The success of PRT has stimulated the development of sophisticated approximation algorithms for large-scale light transport data sets [49, 108, 109, 206] and practical graphics applications such as editing systems [10, 11, 171]. Interested readers may refer to [98] for a comprehensive survey of recent progress of PRT.

Apart from static scenes and distant illumination, PRT has been extended to deformable objects [67, 149, 164], dynamic scenes [76, 149, 170, 211], and local lights [85, 149, 211]. Sloan et al. [164] adopted rotation-invariant zonal harmonics to model radiance transfer functions using non-linear optimization, so that rotating the transfer functions becomes simple and trivial. Although the zonal harmonic basis may yield a more compact representation than spherical harmonics, it is still restricted to low-frequency signals and lighting environments. Moreover, Kristensen et al. [85] introduced unstructured light clouds by precomputing radiance transfer functions with respect to densely-sampled local lights and then clustering the results based on heuristic metrics. Zhou et al. [211] also presented a shadowing approach, namely precomputed shadow fields, for dynamic scenes and local lights. The shadow field was built from concentric shells that surround a light source or an object. At run-time, the shadow fields of scene entities were combined to obtain incident radiance distributions for rendering.

In this dissertation, we focus on static PRT since previous representations and approximation algorithms may be inadequate for harnessing the power of PRT. Available methods either only handle low-frequency information, or suffer from unwieldy amount of data even after compression. For dynamic scenes, the amount of PRT data sets further expands to an impractical degree for real-time applications. The enormous data sets often prohibit high-quality rendering, and subsequently restrict the practical use of PRT. Therefore, we employ the proposed univariate SRBFs and tensor approximation algorithms to permit real-time performance and compact storage space for view-dependent all-frequency PRT in a unified framework. Extending the proposed PRT approach to cope with dynamic scenes remains as the future work.

Bi-Scale Radiance Transfer

While most PRT algorithms tackled spatially-uniform surface appearance, *bi-scale radiance transfer* (BRT) [163] effectively extended PRT with spatially-varying materials in low-frequency lighting environments. By generalizing light transport into macro-scale and meso-scale radiance transfer, BRT effectively combined coarsely-sampled global illumination data with detailed surface appearance. Thus, different sampling rate, precomputation technique, and data representation can be utilized at each scale to efficiently approximate the full global illumination solution.

Based on the same concept, radiance transfer volume [197] and shell radiance texture functions [166] were proposed to respectively model the meso-scale radiance transfer with generalized displacement maps [197] and shell texture functions [21]. In this way, rendered image quality can be greatly improved, while considerably reducing precomputation costs. Nevertheless, previous articles on BRT are restricted to low-frequency light transport since the spherical harmonic basis is employed to approximate the precomputed data. Thus, hard self-shadows and specular highlights at both scales due to the geometric details of meso-structures can not be accurately rendered. By contrast, we focus on the issue of combining all-frequency radiance transfer data at both scales, and propose an all-frequency BRT framework based on the proposed univariate SRBFs and tensor approximation algorithms to achieve real-time rendering on GPUs.

Chapter 3

Univariate Spherical Radial Basis Functions

In computer graphics and vision, previous functional models for radiance functions seem to be insufficient for solving related problems. Based on harmonic functions, it may take tens of thousands of terms to represent regional illumination and shadowing effects. As for wavelet-based methods, there is usually no analytic solution for efficient rotation of wavelet coefficients of a univariate spherical function. In this chapter, we therefore introduce a functional representation, namely univariate *spherical radial basis functions*[1] (SRBFs), to overcome the above-mentioned issues.

This chapter is a more detailed version of Section 4.1 and Appendix A in our published paper [179]. Unlike the published paper that only briefly presented the background of univariate SRBFs, here we additionally discuss many important characteristics of univariate SRBFs, and describe more implementation details of different univariate SRBF representations.

3.1 Mathematical Formulation

Univariate SRBFs are circularly axis-symmetric functions defined on $\mathbb{S}^m$, where $\mathbb{S}^m$ is the unit hyper-sphere embedded in the $(m+1)$-dimensional Euclidean space $\mathbb{R}^{m+1}$. Let ω and ξ denote two points on $\mathbb{S}^m$, $\omega \cdot \xi$ represent the dot product of ω and ξ, and $\phi \in \mathbb{R}$ be the geodesic distance between ω and ξ, namely $\phi = \arccos(\omega \cdot \xi)$. A univariate SRBF is defined as a function that depends on ϕ, and can be expressed in terms of expansions in Legendre polynomials as

$$G(\cos\phi) = G(\omega \cdot \xi) = \sum_{n=0}^{\infty} \ell_n^{(G)} P_n(\omega \cdot \xi), \tag{3.1}$$

[1]Throughout this dissertation, the univariate SRBF is referred to as the original SRBF that was proposed by Freeden et al. [44].

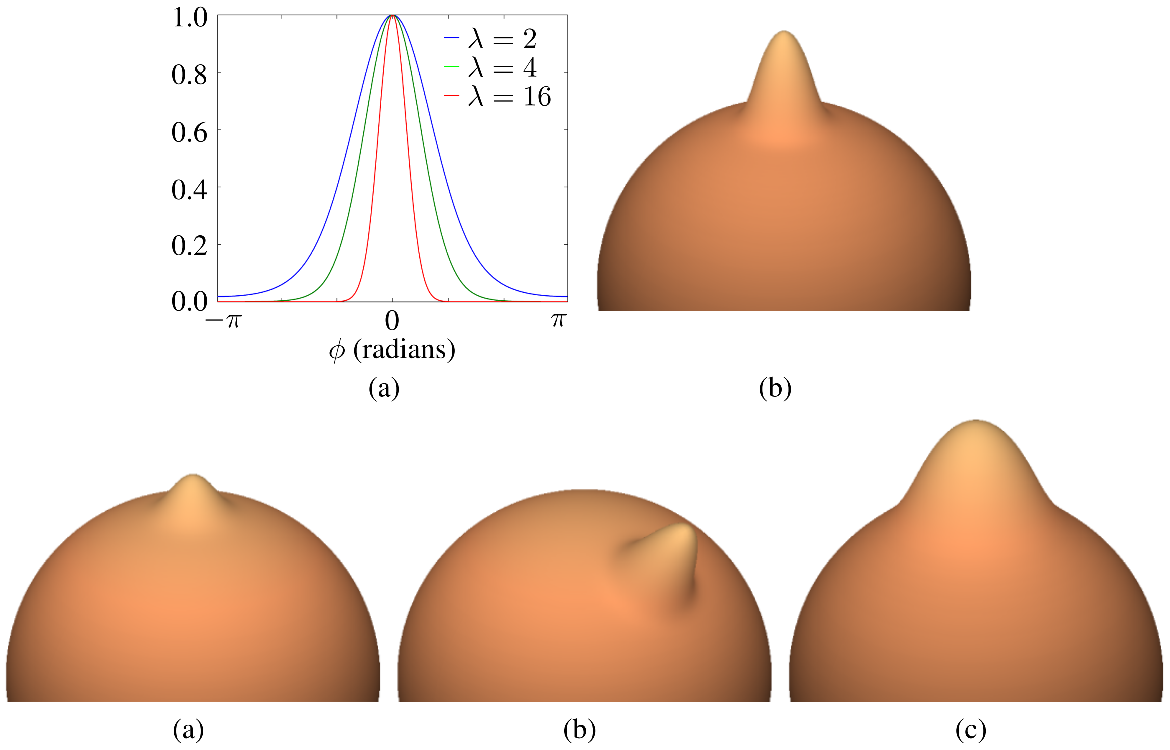


Figure 3.1: (a) A two-dimensional plot of univariate Gaussian SRBFs with different bandwidths. Three-dimensional plots of (b) a univariate Gaussian SRBF and the results by changing its (c) coefficient, (d) center, and (e) bandwidth.

where $P_n(\omega \cdot \xi) \in \mathbb{R}$ is the *normalized Legendre polynomial of degree* n so that $P_n(1) = 1$, the *Legendre coefficients* of $G(\omega \cdot \xi)$, namely $\{\ell_n^{(G)} \in \mathbb{R}\}_{n=0}^{\infty}$, satisfy $\ell_n^{(G)} \geq 0$ for each n as well as $\sum_{n=0}^{\infty} \ell_n^{(G)} < \infty$, and ξ is also known as the center of a univariate SRBF.

Two common examples of univariate SRBFs are the univariate Abel-Poisson SRBF kernel (Equation 3.2) and the univariate Gaussian SRBF kernel (Equation 3.3):

$$G^{(Abel)}(\omega \cdot \xi | \lambda) = \frac{1 - \lambda^2}{\left(1 - 2\lambda(\omega \cdot \xi) + \lambda^2\right)^{\frac{3}{2}}} \quad (0 < \lambda < 1), \tag{3.2}$$

$$G^{(Gau)}(\omega \cdot \xi | \lambda) = e^{-\lambda} e^{\lambda(\omega \cdot \xi)}, \tag{3.3}$$

where $\lambda \in \mathbb{R}$ denotes the bandwidth parameter that controls the coverage of a univariate SRBF. By choosing an appropriate value for λ, a univariate SRBF can be adaptive to the spatial variation of local region. Therefore, univariate SRBFs not only overcome one of the major disadvantages of spherical harmonics, but also possess more degrees of freedom than zonal harmonics. Different types of univariate SRBFs define distinctive distributions on $\mathbb{S}^m$, and may exhibit diverse behaviors in various applications. The two-dimensional and three-dimensional plots of univariate Gaussian SRBFs are illustrated in Figure 3.1, and more examples of univariate SRBFs can be found in [44].

Similar to RBFs in the Euclidean space, given a set of distinct points $\Xi = \{\xi_j \in \mathbb{S}^m\}_{j=1}^{J}$,

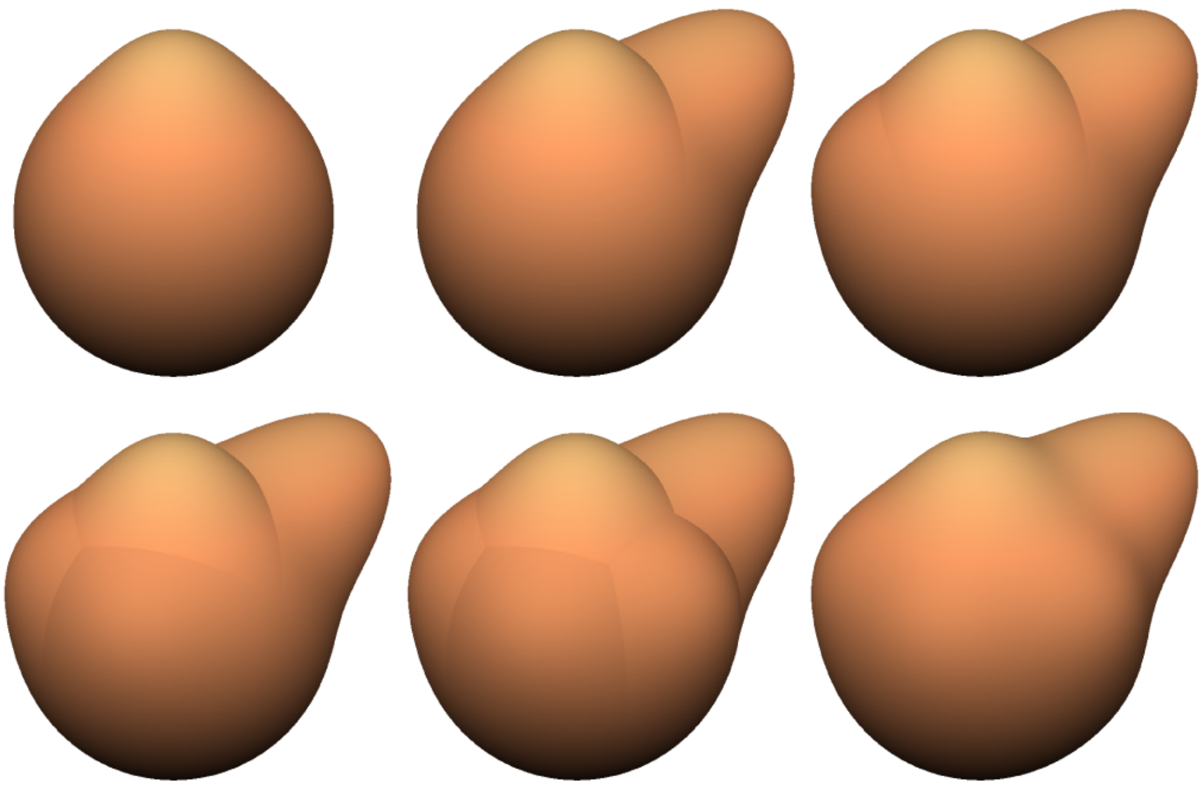


Figure 3.2: A univariate spherical function in univariate SRBF expansions. From left to right, top to bottom: the result after incorporating the first univariate SRBF, the second one, and so forth. Right-most bottom: the final summed result with six univariate SRBFs.

which is called the SRBF center set, and another set of real scalars $\Lambda = \{\lambda_j \in \mathbb{R}\}_{j=1}^{J}$, which is named the SRBF bandwidth set, a univariate spherical function $F(\omega) \in \mathbb{R}$ can be approximated with a linear combination of J univariate SRBFs as

$$F(\omega) \approx \hat{F}(\omega) = \sum_{j=1}^{J} \beta_j G(\omega \cdot \xi_j | \lambda_j), \tag{3.4}$$

where $\beta_j \in \mathbb{R}$ denotes the basis coefficient of the j-th univariate SRBF. Furthermore, $\hat{F}(\omega)$, whose dependence on the parameters of univariate SRBFs is omitted for notational simplicity, specifies the approximate univariate SRBF representation of $F(\omega)$. Univariate SRBFs thus behave as reproducing kernels for interpolating $F(\omega)$ on $\mathbb{S}^m$. An example of a univariate spherical function in univariate SRBF expansions is illustrated in Figure 3.2.

To learn the parameters in Equation 3.4, we can formulate an unconstrained least-squares optimization problem as follows:

$$\min_{\{\beta_j, \xi_j, \lambda_j\}_{j=1}^{J}} \frac{1}{2} \int_{\mathbb{S}^m} \left(F(\omega) - \hat{F}(\omega) \right)^2 d\omega. \tag{3.5}$$

In Sections 3.3 and 3.4, we will further discuss this subject and present practical algorithms for solving Equation 3.5.

3.2 Characteristics of Univariate SRBFs

3.2.1 Spherical Singular Integral

Based on the orthogonal property of Legendre polynomials in the interval $[-1, +1]$, the Legendre expansions of a univariate SRBF especially facilitate the convolution of two univariate SRBFs over $\mathbb{S}^m$, namely the *spherical singular integral*, by

$$\begin{aligned}(G \star_m H)(\xi_g \cdot \xi_h) &= \int_{\mathbb{S}^m} G(\omega \cdot \xi_g) H(\omega \cdot \xi_h) d\omega \\ &= \sum_{n=0}^{\infty} \ell_n^{(G)} \ell_n^{(H)} \frac{\Omega(m)}{d_{m,n}} P_n(\xi_g \cdot \xi_h),\end{aligned} \tag{3.6}$$

where

$$\Omega(m) = \frac{2\pi^{\frac{m+1}{2}}}{\Gamma\left(\frac{m+1}{2}\right)} \tag{3.7}$$

is the total surface area of $\mathbb{S}^m$ which is related to the gamma function $\Gamma(\cdot) \in \mathbb{R}$, $\star_m$ specifies the spherical singular integral operator over $\mathbb{S}^m$, $d_{m,n} \in \mathbb{N}$ denotes the dimension of the space of degree-n spherical harmonics on $\mathbb{S}^m$, and $d\omega$ is the differential surface element on $\mathbb{S}^m$. For more details about Equation 3.6, please refer to [45] and [123].

Furthermore, the spherical singular integral of two univariate Abel-Poisson SRBFs remains another univariate Abel-Poisson SRBF, which is formally given by

$$\left(G^{(Abel)} \star_m H^{(Abel)}\right)(\xi_g \cdot \xi_h | \lambda_g, \lambda_h) = \frac{1 - (\lambda_g \lambda_h)^2}{\left(1 - 2(\lambda_g \lambda_h)(\xi_g \cdot \xi_h) + (\lambda_g \lambda_h)^2\right)^{\frac{3}{2}}}. \tag{3.8}$$

We omit the mathematical proof of Equation 3.8 since it can be easily verified from Equation 3.6 (or from Equations 2.15 and 2.16 in [123]).

Unlike univariate Abel-Poisson SRBFs, the convolution of two univariate Gaussian SRBFs is more complicated, but can be efficiently evaluated for small m. We thus introduce the following theorem for computing the spherical singular integral of two arbitrary univariate Gaussian SRBFs:

Theorem 3.1: *The spherical singular integral of two arbitrary univariate Gaussian SRBFs, $G^{(Gau)}(\omega \cdot \xi_g | \lambda_g)$ and $H^{(Gau)}(\omega \cdot \xi_h | \lambda_h)$, is*

$$\left(G^{(Gau)} \star_m H^{(Gau)}\right)(\xi_g \cdot \xi_h | \lambda_g, \lambda_h) = e^{-(\lambda_g + \lambda_h)} \Omega(m) \Gamma\left(\frac{m+1}{2}\right) \mathcal{I}_{\frac{m-1}{2}}\left(\|r\|_2\right) \left(\frac{2}{\|r\|_2}\right)^{\frac{m-1}{2}}, \tag{3.9}$$

where $r = \lambda_g \xi_g + \lambda_h \xi_h$, $\mathcal{I}_n(\cdot)$ denotes the order-n modified Bessel function of the first kind, and $\|\cdot\|_2$ represents the ℓ^2 norm of a vector.

Corollary 3.2: *For the special case of $m = 2$, the spherical singular integral of two arbitrary univariate Gaussian SRBFs is*

$$\left(G^{(Gau)} \star_2 H^{(Gau)}\right)(\xi_g \cdot \xi_h|\lambda_g, \lambda_h) = 4\pi e^{-(\lambda_g+\lambda_h)} \frac{\sinh \|r\|_2}{\|r\|_2}. \tag{3.10}$$

For the mathematical proof and details about Theorem 3.1, interested readers may refer to Section 3.5.1. We omit the proof of Corollary 3.2 since it is easy to verify Equation 3.10 from Equation 3.9.

3.2.2 Spatial Localization

For a univariate SRBF, we are often interested in how well it localizes on the unit hyper-sphere $\mathbb{S}^m$. This spatial localization property particularly distinguishes univariate SRBFs from harmonic functions. For example, since spherical harmonics are global functions on $\mathbb{S}^m$, high-frequency signals within local regions may need thousands of spherical harmonics to model. Nevertheless, real-world data sets are frequently heterogeneous fields with a few localized high-frequency signals. The lack of spatial localization thus is a fatal drawback of harmonic functions in practical applications.

A primary measure of localization for a univariate SRBF $G(\omega \cdot \xi|\lambda)$ is the *variance* σ_G [44, 123]. Let $G'(\omega \cdot \xi|\lambda)$ be the normalized univariate SRBF of $G(\omega \cdot \xi|\lambda)$ over $\mathbb{S}^m$ so that $(G' \star_m G')(\xi \cdot \xi|\lambda, \lambda) = 1$, which is given by

$$G'(\omega \cdot \xi|\lambda) = \frac{G(\omega \cdot \xi|\lambda)}{(G \star_m G)(\xi \cdot \xi|\lambda, \lambda)}. \tag{3.11}$$

The variance of $G(\omega \cdot \xi|\lambda)$ is then defined as

$$\sigma_G = \int_{\mathbb{S}^m} \|\omega - \tau_G\|_2^2 \left(G'(\omega \cdot \xi|\lambda)\right)^2 d\omega. \tag{3.12}$$

where

$$\tau_G = \int_{\mathbb{S}^m} \omega \left(G'(\omega \cdot \xi|\lambda)\right)^2 d\omega. \tag{3.13}$$

Note that we may interpret $\left(G'(\omega \cdot \xi|\lambda)\right)^2$ as a density distribution on $\mathbb{S}^m$, hence τ_G corresponds to the *center of gravity* of $G(\omega \cdot \xi|\lambda)$.

3.2.3 Univariate Gaussian SRBFs and Von Mises-Fisher Distributions

In this section, we identify an interesting connection between the univariate Gaussian SRBF and the widely-adopted von Mises-Fisher distribution in directional statistics [112], which particularly implies the practical effectiveness of univariate Gaussian SRBFs. This relation is described as the following remark:

Remark 3.3: *Let $G^{(Von)}(\omega\cdot\xi|\lambda)$ be the von Mises-Fisher distribution on $\mathbb{S}^m$, which is defined as*

$$G^{(Von)}(\omega\cdot\xi|\lambda)=\frac{\lambda^{\frac{m-1}{2}}}{(2\pi)^{\frac{m+1}{2}}\mathcal{I}_{\frac{m-1}{2}}(\lambda)}e^{\lambda(\omega\cdot\xi)}\quad(\lambda\geq 0). \tag{3.14}$$

The relation between the univariate Gaussian SRBF and the von Mises-Fisher distribution is identified by the following equation:

$$G^{(Von)}(\omega\cdot\xi|\lambda)=\frac{G^{(Gau)}(\omega\cdot\xi|\lambda)}{g^{(Gau)}(\lambda)}\quad(\lambda\geq 0), \tag{3.15}$$

where

$$\begin{aligned}g^{(Gau)}(\lambda)&=\int_{\mathbb{S}^m}G^{(Gau)}(\omega\cdot\xi|\lambda)d\omega\\&=e^{-\lambda}\Omega(m)\Gamma\Big(\frac{m+1}{2}\Big)\mathcal{I}_{\frac{m-1}{2}}(\lambda)\Big(\frac{2}{\lambda}\Big)^{\frac{m-1}{2}}.\end{aligned} \tag{3.16}$$

Corollary 3.4: *The von Mises-Fisher distribution is equivalent to the normalized univariate Gaussian SRBF with non-negative bandwidth, whose integral over $\mathbb{S}^m$ is 1, and $g^{Gau}(\lambda)$ is known as the normalizing constant of $G^{(Gau)}(\omega\cdot\xi|\lambda)$.*

Corollary 3.5: *For the special case of $m=2$, the normalizing constant of $G^{(Gau)}(\omega\cdot\xi|\lambda)$ is given by*

$$g^{(Gau)}(\lambda)=4\pi e^{-\lambda}\frac{\sinh\lambda}{\lambda}. \tag{3.17}$$

Interested readers may refer to Section 3.5.2 for the mathematical proof of Remark 3.3. Corollaries 3.4 and 3.5 are direct consequences of Remark 3.3. A significant implication of Remark 3.3 is that we may regard (or transform) a univariate spherical function as a density distribution on $\mathbb{S}^m$. Then, various statistical analysis techniques developed for von Mises-Fisher distributions can be applied to univariate Gaussian SRBFs with only minor modifications. In Section 3.4.2, we will discuss this subject in more details about how to employ the mixture model of von Mises-Fisher distributions as a preconditioning process to improve the initial guess of the proposed univariate SRBF representation.

3.3 Uniform Univariate SRBF Representation

3.3.1 Ordinary Least-Squares Projection

In Section 3.1, we described a functional model based on a weighted sum of univariate SRBFs and formulated a least-squares optimization problem (Equation 3.5) for learning model parameters. Nevertheless, searching for an optimal solution to Equation 3.5 requires time-consuming non-linear optimization, which is usually impractical for large-scale visual data sets or when the approximation process should be performed at run-time. Rather than learning all the parameters in Equation 3.4, we propose a *uniform* variant of the proposed univariate SRBF representation, where SRBF centers are uniformly distributed on $\mathbb{S}^m$, and bandwidths are either heuristically determined by users or estimated as a pre-process. In this way, only the basis coefficients need to be optimized. Equation 3.5 then can be re-formulated as an unconstrained linear least-squares problem to significantly reduce computational costs, and global optimum is additionally guaranteed.

More formally, by taking the first order partial derivative of the objective function in Equation 3.5 with respect to each basis coefficient and setting the resulting derivatives to zeros, we have the following set of J linear equations:

$$\begin{aligned}
\sum_{j=1}^{J} \beta_j \int_{\mathbb{S}^m} G(\omega \cdot \xi_j|\lambda_j) G(\omega \cdot \xi_1|\lambda_1) d\omega &= \int_{\mathbb{S}^m} F(\omega) G(\omega \cdot \xi_1|\lambda_1) d\omega, \\
\vdots \qquad\qquad & \qquad\qquad \vdots \\
\sum_{j=1}^{J} \beta_j \int_{\mathbb{S}^m} G(\omega \cdot \xi_j|\lambda_j) G(\omega \cdot \xi_J|\lambda_J) d\omega &= \int_{\mathbb{S}^m} F(\omega) G(\omega \cdot \xi_J|\lambda_J) d\omega.
\end{aligned} \tag{3.18}$$

When the center and bandwidth sets are fixed, all the spherical singular integrals in Equation 3.18 become constants. Equation 3.18 then can be rewritten in a matrix form as $\mathbf{A}\mathbf{b} = \mathbf{g}_F$, where

$$\mathbf{b} = \begin{bmatrix} \beta_1 & \cdots & \beta_J \end{bmatrix}^T, \tag{3.19}$$

$$\mathbf{A} = \begin{bmatrix} (G \star_m G)(\xi_1 \cdot \xi_1|\lambda_1, \lambda_1) & \cdots & (G \star_m G)(\xi_J \cdot \xi_1|\lambda_J, \lambda_1) \\ \vdots & \ddots & \vdots \\ (G \star_m G)(\xi_1 \cdot \xi_J|\lambda_1, \lambda_J) & \cdots & (G \star_m G)(\xi_J \cdot \xi_J|\lambda_J, \lambda_J) \end{bmatrix}, \tag{3.20}$$

$$\mathbf{g}_F = \begin{bmatrix} \int_{\mathbb{S}^m} F(\omega) G(\omega \cdot \xi_1|\lambda_1) d\omega & \cdots & \int_{\mathbb{S}^m} F(\omega) G(\omega \cdot \xi_J|\lambda_J) d\omega \end{bmatrix}^T, \tag{3.21}$$

and $\mathbf{A}$ is known as the interpolation matrix. The optimal basis coefficients thus is given by

$$\mathbf{b} = \mathbf{A}^{+} \mathbf{g}_F, \tag{3.22}$$

where the superscript '$+$' specifies the Moore-Penrose pseudo-inverse of a matrix. In addition, Equation 3.22 is the so-called ordinary least-squares projection [147].

3.3.2 Regularized Least-Squares Projection

It is well-known that sometimes the ordinary least-squares projection is subject to the overfitting problem, in which the variations of the derived basis coefficients tend to be unreasonably large in order to slightly decrease the sum of squared errors. This situation may lead to flickering artifacts and unsmooth animations in data-driven rendering applications. Instead, the regularized least-squares projection [147] can be applied to solve this problem by adding a penalty term to Equation 3.5 as follows:

$$\min_{\{\beta_j,\xi_j,\lambda_j\}_{j=1}^{J}} \frac{1}{2}\int_{\mathbb{S}^m}\left(F(\omega)-\hat{F}(\omega)\right)^2 d\omega+\frac{1}{2}\sum_{j=1}^{J}\mu_j\beta_j^2, \tag{3.23}$$

where μ_j is the user-specified weight of the j-th basis coefficients. As a result, the right-most term of the objective function in Equation 3.23 favors small values for basis coefficients, and thus reduces their variations. By following the same approach as in the derivation of Equation 3.22, the solution to Equation 3.23, when the center and bandwidth parameters are constant, is

$$\mathbf{f}=\left(\mathbf{A}+\mathbf{A}'\right)^{+}\mathbf{g}_F, \tag{3.24}$$

where $\mathbf{A}'\in\mathbb{R}^{J\times J}$ is a diagonal matrix whose j-th diagonal entry is μ_j.

3.4 Scattered Univariate SRBF Representation

3.4.1 Fitting Algorithm

Although a univariate spherical function $F(\omega)$ can be approximated by adopting a set of uniform univariate SRBFs as in Section 3.3, the power of univariate SRBFs over harmonic functions and wavelets mostly lies in the additional degrees of freedom which come from the center and bandwidth parameters. We therefore take a step further to obtain a compact set of scattered univariate SRBFs for approximating $F(\omega)$, where the center and bandwidth of each univariate SRBF are adaptive to $F(\omega)$. This implies that all the parameters in Equation 3.5 should be solved by a non-linear optimization process.

Instead of solving all the three sets of parameters at the same time, we propose an iterative alternating least-squares algorithm that optimizes only one set of parameters at each step, while leaving the others unchanged. This scheme often yields better results, since the three sets of parameters are highly coupled with each other. Algorithm 3.1 summarizes the pseudo-code of the overall fitting process for our scattered univariate SRBF representation. During each

Algorithm 3.1: Fitting algorithm for the scattered univariate SRBF representation.

```
Procedure: UniSRBFOptimize($F(\omega), J, \tilde{F}$)
Input: A univariate spherical function $F(\omega)$ on $\mathbb{S}^m$, the number of univariate SRBFs $J$,
       and the initial guess $\tilde{F} = \left\{\beta_j, \xi_j, \lambda_j\right\}_{j=1}^{J}$.
Output: The optimized parameters $\left\{\beta_j, \xi_j, \lambda_j\right\}_{j=1}^{J}$ in Equation 3.5.

begin
    repeat
        Update basis coefficient set $\{\beta_j\}_{j=1}^{J}$
        Update center set $\{\xi_j\}_{j=1}^{J}$
        Update bandwidth set $\{\lambda_j\}_{j=1}^{J}$
    until convergence
    Update all parameters to obtain a locally optimal solution to Equation 3.5
end
```

iteration, we apply the L-BFGS-B solver [16, 212] to separately update the three sets of parameters. In our current implementation, the gradient computation is performed on GPUs using NVIDIA *Compute Unified Device Architecture* (CUDA) [128]. The computed results are then transferred from GPUs to host memory for the L-BFGS-B solver to update model parameters on CPUs. Since the gradient computation is one of the main performance bottlenecks in the proposed fitting algorithm, this approach can reduce the computation time by a factor of 2 $\sim$ 5. Note that the basis coefficients also can be obtained by using ordinary least-squares projection (or regularized least-squares projection if Equation 3.23 is considered instead).

A special advantage of our univariate SRBF representation is that an incremental fitting algorithm is straightforward. One can always take the residual function $F'(\omega) = F(\omega) - \hat{F}(\omega)$ as the input univariate spherical function to Algorithm 3.1, and obtain the parameters of additional univariate SRBFs. Moreover, the above process can be further iteratively performed until a desired fitting error is reached.

3.4.2 Initial Guess

Overview

Since the approximation results significantly depend on the initial seeds of SRBF parameters, we propose a heuristic method to automatically determine a reasonable initial guess in the hope of achieving smaller approximation errors and reducing the computational costs. Experimental results indicate that this heuristic guess of initial parameters generally works better than taking a set of uniform univariate SRBFs to initialize the optimization process, and decreases the computation time by a factor of 2 $\sim$ 4.

Given a univariate spherical function $F(\omega)$, we first estimate the coverage of each direction in a dense set of unit directions $\Omega = \{\omega_i \in \mathbb{S}^m\}_{i=1}^{I_\Omega}$, where I_Ω denotes the total number of sampling directions in Ω. A priority queue is then constructed with respect to the derived coverage of each direction. To prevent the initial SRBF centers from concentrated in local regions, after selecting a direction ω_i as an initial SRBF center, all the directions which locate within the coverage of ω_i are marked so that they will not be chosen in subsequent steps. This process continues until all the initial SRBF centers are selected. If all directions are marked before all the SRBF centers are determined, we reduce the coverage of each direction in Ω and the selected centers, update the marks, and re-loop over the priority queue to choose the remaining centers. Finally, the initial SRBF bandwidths are derived from the coverage of the selected centers, and the initial basis coefficients are set to the function values of the corresponding selected centers. If univariate Gaussian SRBFs are employed, a preconditioning process based on the mixture model of von Mises-Fisher distributions can be additionally performed to update the initial centers and bandwidths without using non-linear optimization.

Coverage and Bandwidth Estimation

The coverage of a direction ω_i is estimated from its resemblance to nearby sampling directions, and can be computed by a heuristic approach. At first, we loop over the directions $\omega_{i'}$ in Ω, in the order from the nearest to the farthest direction with respect to ω_i, until $\left|F(\omega_i) - F(\omega_{i'})\right|^2$ exceeds a user-defined threshold or $\omega_i \cdot \omega_{i'} < 0$. These nearby directions, which satisfy the above criteria, form a cone-shaped region centered on ω_i. The coverage of ω_i is then defined as the geodesic distance between ω_i and the farthest $\omega_{i'}$ within this cone. Based on the coverage of each direction, the priority queue is constructed by sorting the coverage-weighted function value of each sampling direction.

The remaining problem is to compute the initial bandwidth of a univariate SRBF from the coverage of a selected center. A key observation is that this coverage is indeed related to the variance of a univariate SRBF (Figure 5.5.1 in [44]). Specifically, let ϕ_ξ be the derived coverage of a selected initial SRBF center ξ. We determine the initial bandwidth of a univariate SRBF by correlating ϕ_ξ to the variance of the univariate SRBF and solving the following equation:

$$\sin \phi_\xi = \int_{\mathbb{S}^m} \left\|\omega - \tau_G\right\|_2^2 \big(G(\omega \cdot \xi | \lambda)\big)^2 d\omega. \tag{3.25}$$

For univariate Abel-Poisson SRBFs, Equation 3.25 has an analytic solution. From Equation 4.27 in [123], it is easy to verify that the initial bandwidth is

$$\lambda = \sqrt{\frac{1 - \sin \phi_\xi}{1 + \sin \phi_\xi}}. \tag{3.26}$$

Nevertheless, a closed-form solution to Equation 3.25 usually does not exist for most univariate SRBFs, for example, univariate Gaussian SRBFs. For these univariate SRBFs, we instead approximate the initial bandwidth by a gradient decent approach.

Preconditioning

Remark 3.3 suggests that the estimation techniques for von Mises-Fisher distributions may be applicable to the parameter estimation of the univariate Gaussian SRBF representation. According to the experiments, we have found that the von Mises-Fisher clustering algorithm [6, 55], or the mixture model of von Mises-Fisher distributions, indeed improves the initial guess of centers and bandwidths of univariate Gaussian SRBFs. However, there are two principal issues that should be noted. First, the mixture model of von Mises-Fisher distributions aims at representing a probability distribution on $\mathbb{S}^m$, while the target of the univariate Gaussian SRBF representation is an arbitrary univariate spherical function on $\mathbb{S}^m$. Second, the relation described in Remark 3.3 only holds for univariate Gaussian SRBFs with non-negative bandwidths. For initial guess preconditioning, the second issue can be easily overcome by constraining the initial bandwidth to be non-negative when solving Equation 3.25, but the von Mises-Fisher clustering algorithm should be slightly modified to address the first issue.

Specifically, our key concept is to regard the function value of a sampling direction as its weight. The overall effect thus is equivalent to transforming the univariate spherical function $F(\omega)$ into a discrete probability distribution of sampling directions before applying the clustering algorithm. Based on this concept, Algorithm 3.2 summarizes the modified soft von Mises-Fisher clustering algorithm for preconditioning the initial centers and bandwidths of univariate Gaussian SRBFs. We omit the derivations of Equations in Algorithm 3.2, since it is easy to verify them by following the same approach as in the appendix of [6]. Note that the main purpose of Algorithm 3.2 is to improve the initial guess of centers and bandwidths. The initial basis coefficients are set to the function values of the corresponding SRBF centers at the end of the preconditioning process.

3.5 Mathematical Proofs

3.5.1 Convolution of Univariate Gaussian SRBFs

Recall that we discussed the convolution of two univariate SRBFs in Section 3.2.1. For the convolution of two single-scale univariate Gaussian SRBFs, namely the univariate Gaussian SRBFs with the same bandwidth, Narcowich and Ward [123] derived an analytic solution. However, single-scale univariate SRBFs become inadequate to model univariate spherical functions with scattered SRBFs, whose bandwidths should be adaptive to the given data sets. We thus introduce the spherical singular integral of two multi-scale univariate Gaussian SRBFs, and propose Theorem 3.1 for efficient computation.

Proof of Theorem 3.1: Suppose that we intend to compute the convolution of two multi-scale

Algorithm 3.2: Modified soft von Mises-Fisher clustering algorithm for preconditioning. Interested readers may refer to [6] for the original soft von Mises-Fisher clustering algorithm.

```
Procedure: UniGaussSRBFPrecondition($F(\omega), \Omega, J, \tilde{F}$)
Input: A univariate spherical function $F(\omega)$ on $\mathbb{S}^m$, a set of sampling directions
       $\Omega = \{\omega_i \in \mathbb{S}^m\}_{i=1}^{I_\Omega}$, the number of univariate SRBFs $J$, and the initial guess
       $\tilde{F} = \{\beta_j, \xi_j, \lambda_j\}_{j=1}^{J}$.
Output: The preconditioned initial guess $\{\beta_j, \xi_j, \lambda_j\}_{j=1}^{J}$ in Equation 3.5.

begin
    for $j \leftarrow 1$ to $J$ do $\beta_j \leftarrow \beta_j g^{(Gau)}(\lambda_j)$ (Equation 3.16)  // Initialization
    repeat
        // Expectation step
        for $i \leftarrow 1$ to $I_\Omega$ do
            for $j \leftarrow 1$ to $J$ do $F_j(\omega_i|\xi_j, \lambda_j) \leftarrow G^{(Von)}(\omega_i \cdot \xi_j|\lambda_j)$
            for $j \leftarrow 1$ to $J$ do $\Pr(j|\omega_i, \xi_j, \lambda_j) \leftarrow \frac{\beta_j F_j(\omega_i|\xi_j, \lambda_j)}{\sum_{k=1}^{J} \beta_k F_k(\omega_i|\xi_k, \lambda_k)}$
        end
        // Maximization step
        for $j \leftarrow 1$ to $J$ do
            $\beta_j \leftarrow \frac{1}{\sum_{i=1}^{I_\Omega} F(\omega_i)} \sum_{i=1}^{I_\Omega} F(\omega_i) \Pr(j|\omega_i, \xi_j, \lambda_j)$
            $\xi_j \leftarrow \sum_{i=1}^{I_\Omega} \omega_i F(\omega_i) \Pr(j|\omega_i, \xi_j, \lambda_j)$
            $\bar{r}_j \leftarrow \frac{\|\xi_j\|_2}{\sum_{i=1}^{I_\Omega} F(\omega_i) \Pr(j|\omega_i, \xi_j, \lambda_j)}$
            $\xi_j \leftarrow \frac{\xi_j}{\|\xi_j\|_2}$
            $\lambda_j \leftarrow \frac{\bar{r}_j(m+1) - \bar{r}_j^3}{1 - \bar{r}_j^2}$
        end
    until convergence
    for $j \leftarrow 1$ to $J$ do $\beta_j \leftarrow F(\xi_j)$  // Post-processing
end
```

univariate Gaussian SRBFs, which corresponds to

$$\int_{\mathbb{S}^m} G^{(Gau)}(\omega \cdot \xi_g|\lambda_g) H^{(Gau)}(\omega \cdot \xi_h|\lambda_h) d\omega = e^{-(\lambda_g+\lambda_h)} \int_{\mathbb{S}^m} e^{\omega \cdot (\lambda_g \xi_g + \lambda_h \xi_h)} d\omega. \tag{3.27}$$

Since this convolution is rotation-invariant, we can replace r, where $r = \lambda_g \xi_g + \lambda_h \xi_h$, with $r' = \left[\, \|r\|_2, 0, \ldots, 0 \right]^T$ without losing generality. Rewrite the integral on the right side of Equation 3.27 in terms of spherical coordinates in $\mathbb{R}^{m+1}$, and substitute r with r' to obtain

$$\begin{aligned}\int_{\mathbb{S}^m} e^{\omega\cdot(\lambda_g\xi_g+\lambda_h\xi_h)}d\omega &= \int_0^{2\pi}\int_0^{\pi}\cdots\int_0^{\pi} e^{\omega\cdot r'}(\sin\phi_1)^{m-1}(\sin\phi_2)^{m-2}\cdots\sin\phi_{m-1}d\phi_1 d\phi_2\cdots d\phi_m \\ &= \Omega(m-1)\int_0^{\pi} e^{\|r\|_2\cos\phi_1}(\sin\phi_1)^{m-1}d\phi_1 \\ &= \Omega(m-1)\int_{-1}^{1} e^{\|r\|_2 t}\big(1-t^2\big)^{\frac{m}{2}-1}dt \quad (t=\cos\phi_1).\end{aligned} \tag{3.28}$$

From Equation 3.71(9) in [199], the order-n modified Bessel function of the first kind can be written in an integral form as

$$\mathcal{I}_n(x) = \frac{\big(\frac{1}{2}x\big)^n}{\Gamma\big(n+\frac{1}{2}\big)\Gamma\big(\frac{1}{2}\big)}\int_{-1}^{1} e^{\pm xt}\big(1-t^2\big)^{n-\frac{1}{2}}dt \quad (n>-\frac{1}{2}). \tag{3.29}$$

By combining the last line of Equation 3.28 with Equation 3.29 and substituting the result into Equation 3.27, Equation 3.9 can be derived. □

3.5.2 Relation between Univariate Gaussian SRBF and Von Mises-Fisher Distribution

In Section 3.2.3, we mentioned about the equivalence of the normalized univariate Gaussian SRBF (with non-negative bandwidth) and the von Mises-Fisher distribution, which serves as a solid base for the preconditioning process in Section 3.4.2. In the following paragraph, we will analytically prove this intriguing relation.

Proof of Remark 3.3: Consider transforming a univariate Gaussian SRBF $G^{(Gau)}(\omega\cdot\xi|\lambda)$, with $\lambda\geq 0$, into a probability distribution function $\Pr^{(Gau)}(\omega\cdot\xi|\lambda)$ on $\mathbb{S}^m$ as

$$\Pr{}^{(Gau)}(\omega\cdot\xi|\lambda) = \frac{G^{(Gau)}(\omega\cdot\xi|\lambda)}{g^{(Gau)}(\lambda)} \quad (\lambda\geq 0), \tag{3.30}$$

where the normalizing constant $g^{(Gau)}(\lambda)$ is obtained by integrating $G^{(Gau)}(\omega\cdot\xi|\lambda)$ over $\mathbb{S}^m$ as

$$g^{(Gau)}(\lambda) = \int_{\mathbb{S}^m} G^{(Gau)}(\omega\cdot\xi|\lambda)d\omega = e^{-\lambda}\int_{\mathbb{S}^m} e^{\lambda(\omega\cdot\xi)}d\omega. \tag{3.31}$$

Since the integral on the right side of Equation 3.31 is similar to the integral on the left side of Equation 3.28, Equation 3.16 can be derived by substituting r and $\lambda_g+\lambda_h$ with λ into Equation 3.9. From Equations 3.7 and 3.16, Remark 3.3 is proved by simply comparing Equation 3.30 with the definition of the von Mises-Fisher distribution (Equation 3.14). □

Chapter 4

Multivariate Spherical Radial Basis Functions

A compact and efficient representation for large-scale visual data sets remains challenging in practice. The enormous amount of observations frequently becomes the performance bottleneck at run-time and prohibits further analysis in computer graphics and vision applications. In addition to data size, a real-world visual data set is usually a mixed effect of various types of physical factors. This high-dimensional nature is so complicated that simple analytic models often fail to describe the *multivariate* behavior of a data set.

In this chapter, we propose a novel functional representation, namely multivariate *spherical radial basis functions* (SRBFs), to solve this issue. The complex behaviors of a visual data set are described as a linear combination of multivariate SRBFs, while each multivariate SRBF is constructed from the product of more than one univariate SRBF. As a result, many popular analytic models and factorization-based approaches would fall within this general *weighted sum-of-products* representation.

4.1 Mathematical Formulation

4.1.1 Basic Definitions

In Chapter 3, we introduced a functional model based on univariate SRBFs for a univariate spherical function $F(\omega) \in \mathbb{R}$. As shown in Equation 3.4, $F(\omega)$ can be approximated in univariate SRBF expansions. Nevertheless, there are two major problems when applying Equation 3.4 to a model multivariate spherical function, such as the *bidirectional reflectance distribution function* (BRDF). First, a real-world visual data set may be a heterogenous field that results from various physical factors. Whether these factors are visible or invisible, the observed data distribution is frequently a function of at least two different variables, for example, illumination and view directions for a BRDF. However, Equation 3.4 is a univariate representation that can only account for a single direction on $\mathbb{S}^m$. This suggests that a multivariate representation may be more favorable to describe the complex behaviors of a multivariate spherical function. Second, one may suggest that after sampling a multivariate spherical function into multiple univariate spherical functions under different conditions, Equation 3.4 can be applied to separately model the resulting univariate functions. For instance, we can respectively consider the reflectance distribution of each illumination (or view) direction for a BRDF. However, the outcome is a discrete and often non-compact representation. It is also a non-trivial matter to generalize this representation to estimate the distributions under novel conditions, such as novel illumination/view directions.

To represent a multivariate spherical function under different physical conditions, we can construct a multivariate SRBF from the product of several circularly axis-symmetric univariate SRBFs. For a complex or heterogenous multivariate spherical function, multiple multivariate SRBFs can be further linearly mixed to derive a general weighted sum-of-products model. More formally, we introduce the following definition of a multivariate SRBF:

Definition 4.1: *Let $\Omega = \{\omega_n \in \mathbb{S}^{m_n}\}_{n=1}^{N}$ and $\Xi = \{\xi_n \in \mathbb{S}^{m_n}\}_{n=1}^{N}$ denote two N-element point sets, with ω_n and ξ_n on the n-th unit hyper-sphere $\mathbb{S}^{m_n}$ in $\mathbb{R}^{m_n+1}$. We define a multivariate SRBF on the Cartesian product space $\mathbb{S}^{m_1} \times \mathbb{S}^{m_2} \times \cdots \times \mathbb{S}^{m_N}$ as*

$$G\big(\Omega|\Xi\big) = G\big(\omega_1, \omega_2, \ldots, \omega_N | \xi_1, \xi_2, \ldots, \xi_N\big) = \prod_{n=1}^{N} G(\omega_n \cdot \xi_n). \tag{4.1}$$

The multivariate Abel-Poisson SRBF kernel thus corresponds to

$$G^{(Abel)}\big(\Omega|\Xi, \Lambda\big) = \prod_{n=1}^{N} \frac{1-\lambda_n^2}{\big(1 - 2\lambda_n(\omega_n \cdot \xi_n) + \lambda_n^2\big)^{\frac{3}{2}}} \quad (\forall n,\ 0 < \lambda_n < 1), \tag{4.2}$$

and the multivariate Gaussian SRBF kernel is given by

$$G^{(Gau)}\big(\Omega|\Xi, \Lambda\big) = e^{\sum_{n=1}^{N} \big(\lambda_n(\omega_n \cdot \xi_n) - \lambda_n\big)}, \tag{4.3}$$

where $\Lambda = \{\lambda_n \in \mathbb{R}\}_{n=1}^{N}$ is the set of bandwidth parameters of the involved univariate SRBFs, and Ξ is also known as the SRBF center set. Similar to the univariate SRBF representation, an N-variate spherical function $F(\Omega) \in \mathbb{R}$, with the n-th variable ω_n defined on $\mathbb{S}^{m_n}$, thus can be approximated based on a weighted sum-of-products representation as

$$F(\Omega) \approx \hat{F}(\Omega) = \sum_{j=1}^{J} \beta_j G\big(\Omega|\Xi_j, \Lambda_j\big) = \sum_{j=1}^{J} \beta_j \prod_{n=1}^{N} G\big(\omega_n \cdot \xi_{j,n} | \lambda_{j,n}\big), \tag{4.4}$$

where $\Xi_j = \{\xi_{j,n} \in \mathbb{S}^{m_n}\}_{n=1}^{N}$ and $\Lambda_j = \{\lambda_{j,n} \in \mathbb{R}\}_{n=1}^{N}$ are respectively known as the center set and the bandwidth set of the j-th multivariate SRBF. Moreover, $\hat{F}(\Omega)$, whose dependence on the parameters of multivariate SRBFs is omitted for notational simplicity, denotes the approximate multivariate SRBF representation of $F(\Omega)$. As we will present in Section 4.3, Equation 4.4 can be further extended into a more general model when combined with optimized parameterization.

4.1.2 Example

Consider a BRDF $\rho(\omega_l, \omega_v) \in \mathbb{R}$, where ω_l and ω_v respectively denote the illumination and view directions on $\mathbb{S}^2$. Based on Equation 4.4, we can approximate $\rho(\omega_l, \omega_v)$ as

$$\rho(\omega_l, \omega_v) \approx \hat{\rho}(\omega_l, \omega_v) = \sum_{j=1}^{J} \beta_j^{(\rho)} G\big(\omega_l \cdot \xi_{j,\omega_l}^{(\rho)} | \lambda_{j,\omega_l}^{(\rho)}\big) G\big(\omega_v \cdot \xi_{j,\omega_v}^{(\rho)} | \lambda_{j,\omega_v}^{(\rho)}\big). \tag{4.5}$$

where $\{\beta_j^{(\rho)} \in \mathbb{R}\}_{j=1}^J$ is the basis coefficient set, $\{\xi_{j,\omega_l}^{(\rho)} \in \mathbb{S}^2\}_{j=1}^J$ and $\{\lambda_{j,\omega_l}^{(\rho)} \in \mathbb{R}\}_{j=1}^J$ respectively denote the center set and the bandwidth set for illumination variations, and $\{\xi_{j,\omega_v}^{(\rho)} \in \mathbb{S}^2\}_{j=1}^J$ as well as $\{\lambda_{j,\omega_v}^{(\rho)} \in \mathbb{R}\}_{j=1}^J$ instead specify the center set and the bandwidth set for view variations.

Note that Equation 4.5 is similar to many factorization-based representations for BRDFs, especially principal component analysis [77] and non-negative matrix factorization [91], but the proposed multivariate SRBF representation, like other parametric models, is more compact and intuitive to interpret or edit the derived parameters. This relation to factorization-based methods also suggests the potential of our multivariate SRBF representation to approximate various reflectance functions.

4.2 Scattered Multivariate SRBF Representation

4.2.1 Fitting Algorithm

To learn the model parameters in Equation 4.4, an unconstrained least-squares optimization problem can be formulated as follows:

$$\min_{\{\beta_j, \Xi_j, \Lambda_j\}_{j=1}^J} \frac{1}{2} \int_{\mathbb{S}^{m_1}} \int_{\mathbb{S}^{m_2}} \cdots \int_{\mathbb{S}^{m_N}} \Big(F(\Omega) - \hat{F}(\Omega)\Big)^2 d\omega_1 d\omega_2 \cdots d\omega_N + E_{addl}, \tag{4.6}$$

where E_{addl} denotes the additional energy term that should also be minimized for a robust and satisfactory solution. Note that since E_{addl} may substantially vary in different applications, its mathematical formulation will not be precisely defined in this dissertation. E_{addl} thus is left as a flexible and application-specific term for readers, and a special case of E_{addl} will be discussed in Section 4.4.

Based on Algorithm 3.1, we propose an iterative alternating least-squares algorithm to solve Equation 4.6. The key concept is similar to Algorithm 3.1. At a time, we specifically optimize only one out of three types of parameters: basis coefficients, center sets, or bandwidth sets, while fixing the other types of parameters. Algorithm 4.1 lists the pseudo-code of the proposed fitting algorithm for the scattered multivariate SRBF representation. Note that the optimization

Algorithm 4.1: Fitting algorithm for the scattered multivariate SRBF representation.

```
Procedure: MultiSRBFOptimize($F(\Omega)$, $J$, $\tilde{F}$)
Input: An $N$-variate spherical function $F(\Omega)$ on $\mathbb{S}^{m_1}\times\mathbb{S}^{m_2}\times\cdots\times\mathbb{S}^{m_N}$, the number of
       multivariate SRBFs $J$, and the initial guess $\tilde{F}=\left\{\beta_j,\Xi_j,\Lambda_j\right\}_{j=1}^{J}$.
Output: The optimized parameters $\left\{\beta_j,\Xi_j,\Lambda_j\right\}_{j=1}^{J}$ in Equation 4.6.

begin
    repeat
        Update basis coefficient set $\{\beta_j\}_{j=1}^{J}$
        Update center sets $\{\Xi_j\}_{j=1}^{J}$
        Update bandwidth sets $\{\Lambda_j\}_{j=1}^{J}$
    until convergence
    Update all parameters to obtain a locally optimal solution to Equation 4.6
end
```

of centers and bandwidths for the same multivariate SRBF is not decoupled in Algorithm 4.1. According to the experiments, we found no obvious improvement in approximation errors when decoupling the estimation of parameters for the same multivariate SRBF, but the computational costs of this approach would certainly increase. Moreover, a GPU-based implementation and an incremental variant of Algorithm 4.1 are also feasible by following the same approaches as described in Section 3.4.1.

4.2.2 Initial Guess

Similar to many iterative algorithms, the initial guess of model parameters in Equation 4.6 plays an important role in Algorithm 4.1. We thus propose a technique to heuristically determine an effective initial guess in this section. The proposed approach mainly relies on viewing a multivariate spherical function as multiple univariate functions, and iteratively addresses one variable of the multivariate spherical function at a time.

More formally, consider the N-variate spherical function $F(\Omega)$ in Equation 4.4 and N sampling sets $\{\Omega_n\}_{n=1}^{N}$, where $\Omega_n=\{\omega_{i_n,n}\}_{i_n=1}^{I_{\omega_n}}$ is a set of I_{ω_n} sampling directions on $\mathbb{S}^{m_n}$. Let $F\big(\omega_n|\left\{\omega_{i_j,j}\right\}_{j=1,j\neq n}^{N}\big)\in\mathbb{R}$ be a univariate spherical function obtained by fixing the set of variables $\Omega\setminus\{\omega_n\}$ to $\left\{\omega_{i_j,j}\right\}_{j=1,j\neq n}^{N}$ for $F(\Omega)$, where $\setminus$ denotes the set difference operator. The sampled data of $F(\Omega)$ can be regarded as $I_{\omega_1}\times I_{\omega_2}\times\cdots\times I_{\omega_{n-1}}\times I_{\omega_{n+1}}\times\cdots\times I_{\omega_N}$ data sets, where each data set contains the observations of individual univariate spherical function. We then separately apply the scattered univariate SRBF representation to approximate the observations of each univariate spherical function, but additionally constrain that the representations for different data sets should employ the same sets of centers and bandwidths. After carefully examining the derived parameters, it is obvious that the basis coefficients form the observations of another multivariate spherical function $F_n(\Omega \setminus \{\omega_n\}) \in \mathbb{R}$ without dependence on ω_n. The above process thus can be repeatedly performed to remove a single variable in Ω at each iteration until all model parameters are obtained.

The pseudo-code of this heuristic method is summarized in Algorithm 4.2. Note that although Algorithm 4.2 removes the variables of $F(\Omega)$ in the order from ω_1, ω_2, ..., to ω_N, one can always address them in an arbitrary order. It is also feasible to find the optimal order for small N by a brute-force approach. However, we do not consider this issue in the current implementation. Developing an efficient technique for determining the optimal order is left as a possible research direction in the future.

4.3 Parameterized Multivariate SRBF Representation

4.3.1 Overview

Previous articles have reported that heuristic parameterizations of a reflectance function, such as the half-way and difference vectors [151], can make a great impact on the efficiency of approximation algorithms. Nevertheless, since a pre-defined parameterization method frequently relies on certain assumptions of data characteristics, it may be inadequate to handle various real-world visual data sets. For example, the half-way parameterization tends to align the specular peak of a reflectance function, but the shadowing and masking effects of micro-facets are ignored. This situation will become even worse for a real-world multivariate visual data set, since it is usually measured under diverse and complex physical conditions.

To overcome the disadvantages of fixed parameterization, we propose to learn a set of optimized transformation functions for a given visual data set. Since our goal is to obtain a compact representation, we choose to model the transformation functions using parametric equations. This particularly allows the parameterization process to be tightly integrated into the proposed multivariate SRBF representation. Although the derived optimal solution is constrained to a certain functional form, experimental results show that even a linear mixture of the parameters of a multivariate reflectance function, followed by projection onto the unit hyper-sphere, can be more effective than previous heuristic approaches. Finding the *truly* optimal parameterizations using non-parametric models thus is left as the future work.

Let $\psi(\Omega|\Theta) \in \mathbb{S}^m$ be a transformation function that depends on a given set of parameterization coefficients $\Theta = \{\theta_i \in \mathbb{R}\}_{i=1}^{I_\Theta}$, where I_Θ denotes the total number of parameterization coefficients in Θ and is specified by users. We would like to find an optimal solution to Θ so that a multivariate spherical function $F(\Omega)$ can be efficiently approximated by transforming it into a univariate spherical function $F'\big(\psi(\Omega|\Theta)\big) \in \mathbb{R}$ that is more suitable for univariate SRBF

Algorithm 4.2: Initial guess for the scattered multivariate SRBF representation.

```
Procedure: MultiInitial($F(\Omega), J, \{\Omega_n\}_{n=1}^{N}$)
Input: An $N$-variate spherical function $F(\Omega)$ on $\mathbb{S}^{m_1} \times \mathbb{S}^{m_2} \times \cdots \times \mathbb{S}^{m_N}$, the number of
       multivariate SRBFs $J$, and $N$ sets of sampling directions $\{\Omega_n\}_{n=1}^{N}$.
Output: The initial guess of parameters $\{\beta_j, \Xi_j, \Lambda_j\}_{j=1}^{J}$ in Equation 4.6.

begin
    for $j \leftarrow 1$ to $J$ do $\beta_{j,0}(\Omega) \leftarrow F(\Omega)$
    for $n \leftarrow 1$ to $N$ do
        // Initialization
        $\Omega \leftarrow \Omega \setminus \{\omega_n\}$
        Initialize $\{\xi_{j,n}, \lambda_{j,n}\}_{j=1}^{J}$ with respect to $\beta_{1,n-1}(\omega_n, \{\omega_{1,k}\}_{k=n+1}^{N})$ (Section 3.4.2)
        for $j \leftarrow 1$ to $J$ do $\beta_{j,n}(\Omega) \leftarrow \beta_{j,n-1}(\xi_{j,n}, \Omega)$

        // Model parameter estimation
        repeat
            for $i_n \leftarrow 1$ to $I_{\omega_n}$ do
                for $i_{n+1} \leftarrow 1$ to $I_{\omega_{n+1}}$ do
                    ⋮
                    for $i_N \leftarrow 1$ to $I_{\omega_N}$ do
                        for $j \leftarrow 1$ to $J$ do
                            Update basis coefficients $\beta_{j,n}(\{\omega_{i_k,k}\}_{k=n+1}^{N})$ with respect to
                            $\beta_{j,n-1}(\omega_n, \{\omega_{i_k,k}\}_{k=n+1}^{N})$
                        end
                    end
                end
            end
            Update center set $\{\xi_{j,n}\}_{j=1}^{J}$ with respect to $\{\beta_{j,n-1}(\omega_n, \Omega)\}_{j=1}^{J}$
            Update bandwidth set $\{\lambda_{j,n}\}_{j=1}^{J}$ with respect to $\{\beta_{j,n-1}(\omega_n, \Omega)\}_{j=1}^{J}$
        until convergence
        Update all parameters $\{\beta_{j,n}(\Omega), \xi_{j,n}, \lambda_{j,n}\}_{j=1}^{J}$ to obtain a locally optimal solution
        with respect to $\{\beta_{j,n-1}(\omega_n, \Omega)\}_{j=1}^{J}$
    end
    for $j \leftarrow 1$ to $J$ do $\beta_j \leftarrow \beta_{j,N}$
end
```

expansions:

$$F(\Omega) = F'\big(\psi(\Omega|\Theta)\big) \approx \hat{F}'\big(\psi(\Omega|\Theta)\big) = \sum_{j=1}^{J} \beta_j G\big(\psi(\Omega|\Theta)\cdot \xi_j|\lambda_j\big). \tag{4.7}$$

From Equation 4.7, it is also intuitive to extend the same concept to transform $F(\Omega)$ into an N'-variate spherical function $F'(\Psi) \in \mathbb{R}$ as

$$\begin{aligned} F(\Omega) = F'(\Psi) &= F'\big(\psi_1(\Omega|\Theta_1), \psi_2(\Omega|\Theta_2), \ldots, \psi_{N'}(\Omega|\Theta_{N'})\big) \\ &\approx \hat{F}'(\Psi) = \sum_{j=1}^{J} \beta_j \prod_{n=1}^{N'} G\big(\psi_n(\Omega|\Theta_n)\cdot \xi_{j,n}|\lambda_{j,n}\big), \end{aligned} \tag{4.8}$$

where $\Psi = \big\{\psi_n(\Omega|\Theta_n) \in \mathbb{S}^{m_n}\big\}_{n=1}^{N'}$ is a set of N' transformation functions, $\Theta_n = \{\theta_{i,n}\}_{i=1}^{I_{\Theta_n}}$ specifies the parameterization coefficient set with I_{Θ_n} elements for the n-th transformation function, and $\hat{F}'(\Psi)$ denotes the approximate multivariate SRBF representation of $F'(\Psi)$. Note that the number of variables of $F'(\Psi)$, namely N', is not necessarily identical to that of $F(\Omega)$, but rather can be specified by users. This flexibility particularly allows our paramerterized multivariate SRBF representation to accurately model various complex behaviors of a real-world multivariate spherical function.

In summary, we combine the scattered multivariate SRBF representation and optimized parameterization to derive a parametric model (Equation 4.8), and obtain its parameters by solving the following unconstrained least-squares optimization problem:

$$\min_{\Upsilon} \frac{1}{2} \int_{\mathbb{S}^{m_1}} \int_{\mathbb{S}^{m_2}} \cdots \int_{\mathbb{S}^{m_N}} \Big(F(\Omega) - \hat{F}'(\Psi)\Big)^2 d\omega_1 d\omega_2 \cdots d\omega_N + E_{addl}, \tag{4.9}$$

where $\Upsilon = \Big\{\{\beta_j, \Xi_j, \Lambda_j\}_{j=1}^{J}, \{\Theta_n\}_{n=1}^{N'}\Big\}$, and E_{addl} is the additional energy term similar to Equation 4.6. In Section 4.3.3, we will further present a general algorithm for solving Equation 4.9.

4.3.2 Example

We again take the BRDF $\rho(\omega_l, \omega_v)$ for instance. Based on Equation 4.8, Equation 4.5 can be extended into an N'-variate SRBF representation as

$$\rho(\omega_l, \omega_v) \approx \hat{\rho}'\big(\Psi^{(\rho)}\big) = \sum_{j=1}^{J} \beta_j^{(\rho)} \prod_{n=1}^{N'} G\Big(\psi_n^{(\rho)}\big(\omega_l, \omega_v|\Theta_n^{(\rho)}\big)\cdot \xi_{j,n}^{(\rho)}|\lambda_{j,n}^{(\rho)}\Big), \tag{4.10}$$

where $\Psi^{(\rho)} = \big\{\psi_n^{(\rho)}\big(\omega_l, \omega_v|\Theta_n^{(\rho)}\big) \in \mathbb{S}^2\big\}_{n=1}^{N'}$ is the set of transformation functions for $\rho(\omega_l, \omega_v)$, $\big\{\beta_j^{(\rho)} \in \mathbb{R}\big\}_{j=1}^{J}$ denotes the basis coefficient set, and $\big\{\xi_{j,n}^{(\rho)} \in \mathbb{S}^2\big\}_{j=1}^{J}$, $\big\{\lambda_{j,n}^{(\rho)} \in \mathbb{R}\big\}_{j=1}^{J}$, as well as $\Theta_n^{(\rho)}$ respectively represent the center set, the bandwidth set, and the parameterization coefficient set for the n-th transformed variable. Each transformation function in Equation 4.10 can be modeled with the normalization of a linear combination of ω_l and ω_v as follows:

$$\forall n,\ \psi_n\big(\omega_l, \omega_v | \Theta_n^{(\rho)}\big) = \frac{\theta_{1,n}^{(\rho)}\omega_l + \theta_{2,n}^{(\rho)}\omega_v}{\big\|\theta_{1,n}^{(\rho)}\omega_l + \theta_{2,n}^{(\rho)}\omega_v\big\|_2}, \tag{4.11}$$

where $\theta_{1,n}^{(\rho)} \in \mathbb{R}$ and $\theta_{2,n}^{(\rho)} \in \mathbb{R}$ are the parameterization coefficients in $\Theta_n^{(\rho)}$ for the n-th transformation function.

When $N' = 3$ and $J = 1$, Equation 4.10 is similar to the mathematical formulation of homomorphic factorization [116], but the parameterized multivariate SRBF representation allows a linear combination of multiple multivariate functions to model heterogeneous materials, which is particularly important for representing spatially-varying reflectance data sets. Moreover, Equation 4.11 was inspired by the fact that many common parameterizations for reflectance functions, such as the half-way, illumination, and view vectors, are its special cases. It is also remarkable that these three heuristic parameterizations were employed in the implementation of homomorphic factorization, which further implies the practical effectiveness of our methods.

4.3.3 Fitting Algorithm

Algorithm 4.3 presents the pseudo-code of a practical algorithm for solving Equation 4.9. At a time, we optimize only one out of four types of parameters, including basis coefficients, center sets, bandwidth sets, and parameterization coefficient sets, while the other types of parameters are fixed. Similar to Algorithm 4.1, the optimization of different parameterization coefficient sets is not decoupled for performance issues. Furthermore, it is also straightforward to develop a GPU-based implementation and an incremental variant of Algorithm 4.3.

As for the initial guess of parameters in Equation 4.9, since parameterization coefficient sets strongly depend on the functional form of transformation functions, we currently do not have a common solution to this issue. Nevertheless, previous heuristic parameterizations generally provide an appropriate starting point if they are special cases of the adopted transformation functions. Take Equation 4.11 for example, the initial guess of the first three parameterization coefficient sets can be explicitly set to the half-way, illumination, and view parameterizations, while the remainders are randomly generated. Once the initial guess of parameterization coefficient sets is determined, that of basis coefficients, center sets, and bandwidth sets then can be estimated using Algorithm 4.2.

4.4 Hierarchical Fitting Algorithm

A real-world visual data set is usually a heterogeneous field that results from different physical factors. In addition to directional factors, another common factor is the spatial location of an

Algorithm 4.3: Fitting algorithm for the parameterized multivariate SRBF representation.

```
Procedure: ParamMultiSRBFOptimize($F(\Omega)$, $J$, $\Psi$, $\tilde{F}$)
Input: An $N$-variate spherical function $F(\Omega)$ on $\mathbb{S}^{m_1} \times \mathbb{S}^{m_2} \times \cdots \times \mathbb{S}^{m_N}$, the number of
       multivariate SRBFs $J$, a set of transformation functions $\Psi = \left\{\psi_n(\Omega|\Theta_n)\right\}_{n=1}^{N'}$,
       and the initial guess $\tilde{F} = \left\{\left\{\beta_j, \Xi_j, \Lambda_j\right\}_{j=1}^{J}, \left\{\Theta_n\right\}_{n=1}^{N'}\right\}$.
Output: The optimized parameters $\left\{\left\{\beta_j, \Xi_j, \Lambda_j\right\}_{j=1}^{J}, \left\{\Theta_n\right\}_{n=1}^{N'}\right\}$ in Equation 4.9.

begin
    repeat
        Update basis coefficient set $\{\beta_j\}_{j=1}^{J}$
        Update center sets $\{\Xi_j\}_{j=1}^{J}$
        Update bandwidth sets $\{\Lambda_j\}_{j=1}^{J}$
        Update parameterization coefficient sets $\{\Theta_n\}_{n=1}^{N'}$
    until convergence
    Update all parameters to obtain a locally optimal solution to Equation 4.9
end
```

observation. For example, the *bidirectional texture function* (BTF) [28] is an appearance data set that describes spatially-varying and view-dependent illumination effects. Nevertheless, all of the univariate and multivariate SRBF representations in this dissertation can only address directional variables of a multivariate function. For a spatially-varying natural phenomenon, we may need to ignore its spatial correlations, and then separately apply one of the proposed SRBF representations to model the observations at a single location. When the spatial coherence in the observations is high, this approach seems to be inefficient and may cause serious problems if spatial interpolation techniques are directly applied to the derived parameters instead of the reconstructed data. To solve the above-mentioned issues, we propose a hierarchical fitting algorithm for spatially-varying visual data sets to accelerate the approximation process. It is particularly suitable for multi-scale analysis and data-driven rendering applications due to the inherent multi-resolution structure of derived parameters.

Specifically, consider a spatially-varying multivariate spherical function $F(\Omega, \mathbf{x}) \in \mathbb{R}$, where $\mathbf{x} = \left[x_1, x_2, \ldots, x_m\right]^T$ is an m-dimensional spatial location. Let $\left\{F_i(\Omega, \mathbf{x})\right\}_{i=0}^{I_h}$ denote a hierarchical set of I_h spatially-varying multivariate spherical functions, where $F_i(\Omega, \mathbf{x}) \in \mathbb{R}$ specifies the spatially-varying multivariate spherical function at level i and $F_{I_h}(\Omega, \mathbf{x}) = F(\Omega, \mathbf{x})$. For $i < I_h$, $F_i(\Omega, \mathbf{x})$ is obtained by spatially downsampling $F(\Omega, \mathbf{x})$ by a factor of $d^{I_h - i}$, where d is a user-specified constant. The proposed hierarchical fitting algorithm then operates on $\left\{F_i(\Omega, \mathbf{x})\right\}_{i=0}^{I_h}$ and the initial guess of the scattered multivariate SRBF representation for each spatial location at level 0. As illustrated in Figure 4.1, it also consists of a sequence of upsampling and optimization stages from the coarsest level 0 to the finest level I_h.

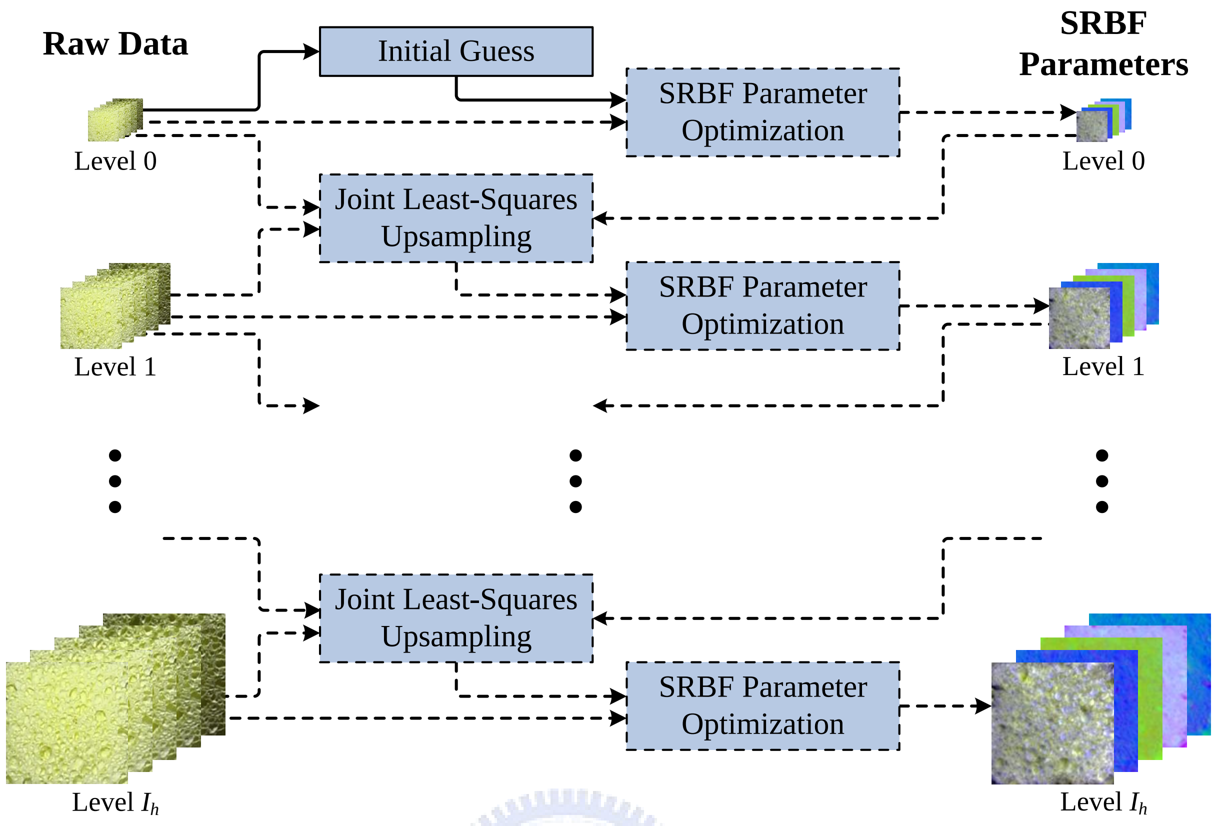

Raw Data
Initial Guess
SRBF Parameters
SRBF Parameter Optimization
Level 0
Level 0
Joint Least-Squares Upsampling
SRBF Parameter Optimization
Level 1
Level 1
Joint Least-Squares Upsampling
SRBF Parameter Optimization
Level I_h
Level I_h

Figure 4.1: Approximate a BTF using the proposed hierarchical fitting algorithm.

For level i ($i > 0$), the upsampling stage (Section 4.4.1) derives the initial solution of the scattered multivariate SRBF representation, for each spatial location at level i, from $\tilde{F}_{i-1}(\mathbf{x})$, where $\tilde{F}_{i-1}(\mathbf{x})$ represents the optimized results of level $i-1$. Instead of using traditional spatial interpolation techniques, such as cubic or Lanczos filtering, we propose a joint least-squares upsampling algorithm by exploiting the relation between $F_i(\Omega, \mathbf{x})$ and $F_{i-1}(\Omega, \mathbf{x})$ to assist the resampling of $\tilde{F}_{i-1}(\mathbf{x})$. After that, the optimization stage (Section 4.4.2) updates the initial solution for each spatial location at level i based on the scattered multivariate SRBF representation. To allow direct spatial interpolation on the optimized parameters without reconstruction, we introduce additional spatial smoothness energy terms in the objective function, which will constrain the parameter coherence of adjacent spatial locations. The pseudo-code of the overall fitting process is summarized in Algorithm 4.4. Note that although we only present a hierarchical algorithm based on the scattered multivariate SRBF representation in this section, extending it to support other SRBF representations is feasible and straightforward to implement.

Parts of this section are related to the approach presented in [40]. Fang [40] implemented a prototype of the hierarchical fitting algorithm on CPUs, and demonstrated its potentials for modeling spatially-varying materials. In this dissertation, we instead accelerate the time-consuming non-linear optimization process of this prototype with GPUs, and further improve its robustness by introducing advanced techniques for high-quality upsampling and the initial guess of SRBF parameters.

Algorithm 4.4: Hierarchical fitting algorithm based on the scattered multivariate SRBF representation.

```
Procedure: HierarchyMultiSRBFOptimize($\{F_i(\Omega, \mathbf{x})\}_{i=0}^{I_h}$, $J$)
Input: A hierarchical set of $I_h$ spatially-varying multivariate spherical functions
       $\{F_i(\Omega, \mathbf{x})\}_{i=0}^{I_h}$ and the number of multivariate SRBFs $J$.
Output: The optimized parameters of each level $\{\tilde{F}_i(\mathbf{x})\}_{i=0}^{I_h}$.
begin
    // Initialization
    foreach spatial location $\mathbf{x}$ at level 0 do
        Initialize $\tilde{F}_0(\mathbf{x})$ using Algorithm 4.2
        $\tilde{F}_0(\mathbf{x}) \leftarrow$ MultiSRBFOptimize($F_0(\Omega, \mathbf{x})$, $J$, $\tilde{F}_0(\mathbf{x})$) (Algorithm 4.1)
    end
    // Hierarchical optimization
    for $i \leftarrow 1$ to $I_h$ do
        Upsample $\tilde{F}_{i-1}(\mathbf{x})$ to obtain $\tilde{F}_i(\mathbf{x})$   // Upsampling stage
        // Optimization stage
        repeat
            foreach spatial location $\mathbf{x}$ at level $i$ do
                $\tilde{F}_i(\mathbf{x}) \leftarrow$ MultiSRBFOptimize($F_i(\Omega, \mathbf{x})$, $J$, $\tilde{F}_i(\mathbf{x})$) (Algorithm 4.1)
            end
        until convergence
    end
end
```

4.4.1 Upsampling Stage

Given the optimized parameters of pyramid level $i-1$, namely $\tilde{F}_{i-1}(\mathbf{x})$, an appropriate initial solution of each spatial location at level i is derived in the upsampling stage. This initial solution significantly influences the quality and computational costs for approximating $F_i(\Omega, \mathbf{x})$. However, since some details of $F_i(\Omega, \mathbf{x})$ may be lost when downsampled to $F_{i-1}(\Omega, \mathbf{x})$ during hierarchical set construction, traditional spatial interpolation techniques are inadequate for a high-quality upsampling from $\tilde{F}_{i-1}(\mathbf{x})$. The key insight is that because both $F_i(\Omega, \mathbf{x})$ and $F_{i-1}(\Omega, \mathbf{x})$ are available in this stage, their relation can be employed to 'jointly' derive the initial guess of $\tilde{F}_i(\mathbf{x})$ from $\tilde{F}_{i-1}(\mathbf{x})$. This relies on the assumptions that the relation between $F_i(\Omega, \mathbf{x})$ and $F_{i-1}(\Omega, \mathbf{x})$ is similar to that between $\tilde{F}_i(\mathbf{x})$ and $\tilde{F}_{i-1}(\mathbf{x})$, and $\tilde{F}_{i-1}(\mathbf{x})$ approximates $F_{i-1}(\Omega, \mathbf{x})$ with low reconstruction errors.

Specifically, the level-i multivariate spherical function at a spatial location $\mathbf{x}$ can be approximated with a linear combination of the level-$(i-1)$ multivariate spherical function at multiple spatial locations as follows:

$$F_i(\Omega, \mathbf{x}) \approx \sum_{\mathbf{x}' \in \mathcal{N}_{i-1}(\mathbf{x})} w_{\mathbf{x}'} F_{i-1}(\Omega, \mathbf{x}'), \tag{4.12}$$

where $\mathcal{N}_{i-1}(\mathbf{x})$ represents the set of participating spatial locations at level $i-1$ for $\mathbf{x}$, and $w_{\mathbf{x}'}$ denotes the blending weight of a spatial location $\mathbf{x}' \in \mathcal{N}_{i-1}(\mathbf{x})$. In current implementation, we heuristically determine $\mathcal{N}_{i-1}(\mathbf{x})$ as the neighboring spatial locations of $\mathbf{x}$ at level $i-1$ within a user-defined window. Since both $F_i(\Omega, \mathbf{x})$ and $F_{i-1}(\Omega, \mathbf{x})$ are known in this stage, the unknown blending weights thus can be derived by solving an unconstrained linear least-squares problem, and then applied to compute the initial solution of $\tilde{F}_i(\mathbf{x})$ as

$$\tilde{F}_i(\mathbf{x}) = \sum_{\mathbf{x}' \in \mathcal{N}_{i-1}(\mathbf{x})} w_{\mathbf{x}'} \tilde{F}_{i-1}(\mathbf{x}'). \tag{4.13}$$

It should be noted that reconstructing the interpolated parameters of participating spatial locations may not exactly correspond to interpolating their reconstructed values. For example, since the center and bandwidth sets are related to the exponents of multivariate Gaussian SRBFs, Equation 4.13 will linearly blend these two sets at different spatial locations in the logarithmic space, which is definitely not equivalent to the weighted sum of the multivariate Gaussian SRBFs at each spatial location. However, they would be close to each other if the spatial variations in the parameters of participating spatial locations are smooth. Since the model parameters of level $i-1$ were already updated with spatial smoothness energy terms in the optimization stage of previous iteration, we have found that the proposed joint least-squares upsampling algorithm works very well in practice.

4.4.2 Optimization Stage

In this stage, the initial guess of each spatial location at level i is individually updated to obtain a locally optimal solution to Equation 4.6. To guarantee spatial coherence in the derived parameters, we introduce additional smoothness energy terms of basis coefficients (Equation 4.15), centers (Equation 4.16), and bandwidths (Equation 4.17) to Equation 4.6 as follows:

$$E_{addl} = \mu_\beta E_\beta + \mu_\xi E_\xi + \mu_\lambda E_\lambda, \tag{4.14}$$

$$E_\beta = \sum_{\mathbf{x}' \in \mathcal{N}'_i(\mathbf{x})} \sum_{j=1}^{J} \frac{1}{2} \big(\beta_j(\mathbf{x}) - \beta_j(\mathbf{x}')\big)^2, \tag{4.15}$$

$$E_\xi = \sum_{\mathbf{x}' \in \mathcal{N}'_i(\mathbf{x})} \sum_{j=1}^{J} \sum_{n=1}^{N} \big(1 - \xi_{j,n}(\mathbf{x}) \cdot \xi_{j,n}(\mathbf{x}')\big), \tag{4.16}$$

$$E_\lambda = \sum_{\mathbf{x}' \in \mathcal{N}'_i(\mathbf{x})} \sum_{j=1}^{J} \sum_{n=1}^{N} \frac{1}{2} \big(\lambda_{j,n}(\mathbf{x}) - \lambda_{j,n}(\mathbf{x}')\big)^2, \tag{4.17}$$

where $\mathcal{N}'_i(\mathbf{x})$ denotes the set of participating spatial locations at levels i and $i-1$ for $\mathbf{x}$, and μ_β, μ_ξ, as well as μ_λ are respectively the user-defined weights for E_β, E_ξ, and E_λ. Note that the SRBF parameters in Equations 4.14–4.17 should depend on the level index i and spatial location $\mathbf{x}$, but we drop them for notational simplicity. With Equation 4.14, the smoothness energy terms will guide the model parameters of $\mathbf{x}$ to approach those of $\mathcal{N}'_i(\mathbf{x})$. Similar to $\mathcal{N}_{i-1}(\mathbf{x})$ in the upsampling stage, $\mathcal{N}'_i(\mathbf{x})$ can be defined as the 'valid' neighboring spatial locations of $\mathbf{x}$ at levels i and $i-1$ within a user-defined window, while a valid spatial location is referred to as the spatial location whose model parameters have ever been optimized. Since a change in the model parameters of $\mathbf{x}$ will influence those of $\mathcal{N}'_i(\mathbf{x})$, the above process is repeated until the parameters of each spatial location at level i converge or a user-defined maximum number of passes is reached.

If the parameterized multivariate SRBF representation is employed instead, we should also add another smoothness energy term of parameterization coefficients (Equation 4.19) to Equation 4.14 as

$$E_{addl} = \mu_\beta E_\beta + \mu_\xi E_\xi + \mu_\lambda E_\lambda + \mu_\theta E_\theta, \tag{4.18}$$

$$E_\theta = \sum_{\mathbf{x}' \in \mathcal{N}'_i(\mathbf{x})} \sum_{n=1}^{N'} \sum_{j=1}^{I_{\Theta_n}} \frac{1}{2} \big(\theta_{j,n}(\mathbf{x}) - \theta_{j,n}(\mathbf{x}')\big)^2, \tag{4.19}$$

where μ_θ is the user-defined weight for E_θ.

For multivariate Abel-Poisson and Gaussian SRBFs (Equations 4.2 and 4.3), since centers and bandwidths are highly coupled with each other, we additionally include a smoothness energy term of bandwidth-scaled centers (Equation 4.21) into Equation 4.14 (or Equation 4.18) as

$$E_{addl} = \mu_\beta E_\beta + \mu_\xi E_\xi + \mu_\lambda E_\lambda + \mu_{\xi,\lambda} E_{\xi,\lambda}, \tag{4.20}$$

$$E_{\xi,\lambda} = \sum_{\mathbf{x}' \in \mathcal{N}'_i(\mathbf{x})} \sum_{j=1}^{J} \sum_{n=1}^{N} \frac{1}{2} \Big\| \lambda_{j,n}(\mathbf{x})\xi_{j,n}(\mathbf{x}) - \lambda_{j,n}(\mathbf{x}')\xi_{j,n}(\mathbf{x}') \Big\|_2^2, \tag{4.21}$$

where $\mu_{\xi,\lambda}$ is the user-defined weight for $E_{\xi,\lambda}$.

Chapter 5

Clustered Tensor Approximation

Nowadays, computer graphics and vision researchers are facing the challenge of processing large-scale multi-dimensional visual data sets. As a compression and complexity-decreasing method, dimensionality reduction contributes tremendously to data-driven rendering. While the amount of visual data sets significantly increases to account for more photo-realistic image synthesis, achieving the objective of both compact compressed data and real-time rendering performance becomes more and more difficult. It is also important that the reconstruction process is able to take full advantage of graphics hardware. Nevertheless, most of previous approximation algorithms either are inefficient in decoding data on modern GPUs or can not provide low reconstruction errors under a high compression ratio.

Based on tensor approximation, we thus propose a novel dimensionality reduction algorithm, namely *clustered tensor approximation* (CTA), to exploit the coherence in a multi-dimensional visual data set, so that both storage space and rendering time are substantially reduced. CTA is also a general approximation algorithm that can be employed to compress, analyze, or represent any kinds of large-scale data sets whose structures are intrinsically multi-dimensional.

This chapter is a rewritten and extended version of Section 5.2 in our published paper [179]. It describes the algorithmic concept and implementation issues of the proposed CTA algorithm in more details, and additionally includes some mathematical proofs that demonstrate the correctness of CTA.

5.1 Preliminaries and Background

This section briefly reviews the background of tensor approximation[1]. For completeness, notation and basic definitions of tensor operators are also introduced. Although the out-of-core

[1]Throughout this dissertation, tensor approximation is particularly referred to as the multi-way analysis based on Tucker models [165, 180, 181]. Readers may notice that there is another popular multi-linear model named parallel factor analysis [57, 165] or canonical decomposition [18, 165] in chemometrics and psychometrics, but it is beyond the scope of this dissertation.

tensor decomposition algorithm [192] was applied in our experiments, in-core tensor notation is adopted in this dissertation for notational simplicity. For more details about various tensor operators, interested readers may additionally refer to [32, 33, 79, 165].

5.1.1 Basic Definitions

In the following context, scalars are written as italic roman lowercase letters $(a, b, \ldots)$; vectors as boldface roman lowercase letters $(\mathbf{a}, \mathbf{b}, \ldots)$; matrices as boldface roman capitals $(\mathbf{A}, \mathbf{B}, \ldots)$; tensors as boldface calligraphic capitals $(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \ldots)$. The entry in row i and column j of a matrix $\mathbf{U} \in \mathbb{R}^{I \times J}$ is denoted by $(\mathbf{U})_{ij}$. The i-th row of $\mathbf{U}$ is written as $(\mathbf{U})_{i*}$ and the j-th column of $\mathbf{U}$ as $(\mathbf{U})_{*j}$. The transpose of a matrix $\mathbf{U}$ is denoted by $\mathbf{U}^T$.

A tensor can be viewed as a multi-dimensional array or a high-order mapping over a set of vector spaces. Familiar examples are vectors (first order tensors) and matrices (second order tensors). An N-th order tensor is written as $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, where $I_1, I_2, \ldots, I_N \in \mathbb{N}$ are respectively the mode-1, mode-2, …, and mode-N ranks. Similar to a matrix, the entry of an N-th order tensor $\boldsymbol{\mathcal{A}}$ is denoted by $(\boldsymbol{\mathcal{A}})_{i_1 i_2 \cdots i_N}$, where $i_1 \in \{1, 2, \ldots, I_1\}$, $i_2 \in \{1, 2, \ldots, I_2\}$, …, and $i_N \in \{1, 2, \ldots, I_N\}$ are respectively the mode-1, mode-2, …, and mode-N indices.

The Frobenius norm of an N-th order tensor $\boldsymbol{\mathcal{A}}$ is defined as $\|\boldsymbol{\mathcal{A}}\|_F = \sqrt{\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{A}} \rangle}$, where

$$\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} (\boldsymbol{\mathcal{A}})_{i_1 i_2 \cdots i_N} (\boldsymbol{\mathcal{B}})_{i_1 i_2 \cdots i_N} \tag{5.1}$$

denotes the scalar product of two N-th order tensors $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. The mode-$n$ product of an N-th order tensor $\boldsymbol{\mathcal{A}}$ and a matrix $\mathbf{U} \in \mathbb{R}^{J_n \times I_n}$ is written as $\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{A}} \times_n \mathbf{U}$, where the entries of the resulting N-th order tensor $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times I_{n+2} \times \cdots \times I_N}$ are given by

$$(\boldsymbol{\mathcal{B}})_{i_1 i_2 \cdots i_{n-1} j_n i_{n+1} i_{n+2} \cdots i_N} = \sum_{i_n=1}^{I_n} (\boldsymbol{\mathcal{A}})_{i_1 i_2 \cdots i_{n-1} i_n i_{n+1} i_{n+2} \cdots i_N} (\mathbf{U})_{j_n i_n}, \tag{5.2}$$

for $j_n = 1, 2, \ldots, J_n$ and $i_k = 1, 2, \ldots, I_k$ for $k = 1, 2, \ldots, n-1, n+1, n+2, \ldots, N$. The symbol $\mathit{uf}_n(\boldsymbol{\mathcal{A}}) \in \mathbb{R}^{I_n \times (I_{n+1} I_{n+2} \cdots I_N I_1 I_2 \cdots I_{n-1})}$ denotes the mode-n unfolded matrix of an N-th order tensor $\boldsymbol{\mathcal{A}}$, which results from retaining the n-th mode of $\boldsymbol{\mathcal{A}}$ and flattening the others (refer to Figure 2.1 in [33]).

Moreover, we further define a series of mode-n products of a tensor and a set of matrices as follows:

Definition 5.1: *A series of mode-n products of an N-th order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a set of matrices $\left\{\mathbf{U}_n \in \mathbb{R}^{J_n \times I_n}\right\}_{n=1}^{N}$ is defined as*

$$\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{A}} \bigtimes_{n=1}^{N}{}_{n} \mathbf{U}_n = \boldsymbol{\mathcal{A}} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \cdots \times_N \mathbf{U}_N. \tag{5.3}$$

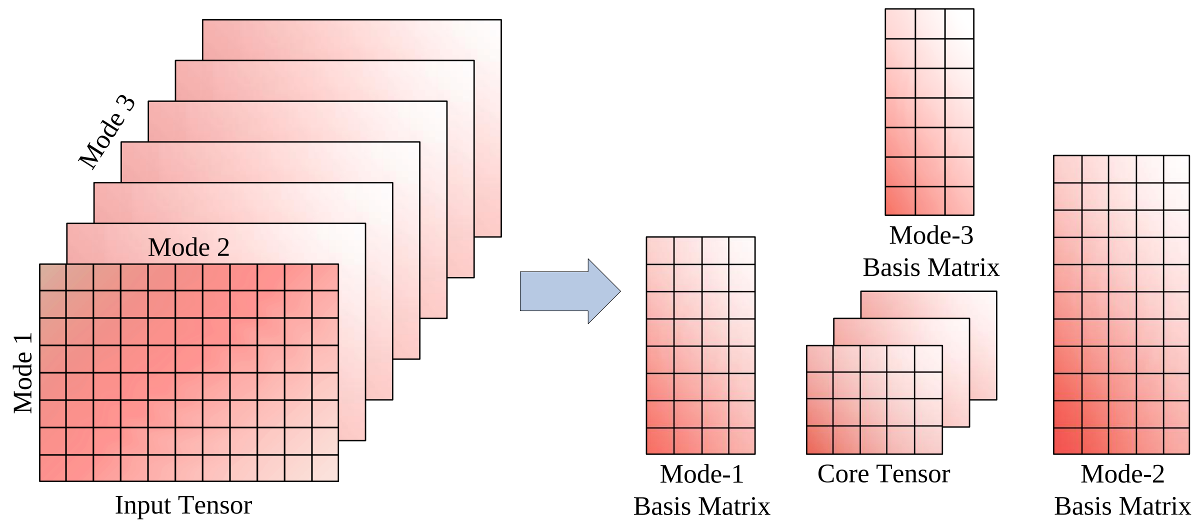


Figure 5.1: The decomposed core tensor and basis matrices of a third order tensor based on tensor approximation. The input tensor of size 8×11×7 is decomposed into the core tensor of size 4×5×3, the mode-1 basis matrix of size 8×4, the mode-2 basis matrix of size 11×5, and the mode-3 basis matrix of size 7×3.

We also introduce the following definition for a mode-n sub-tensor of a tensor:

Definition 5.2: *The i-th mode-n sub-tensor of an N-th order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is written as $\boldsymbol{\mathcal{A}}_{\langle n_i \rangle} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{n-1} \times 1 \times I_{n+1} \times I_{n+2} \times \cdots \times I_N}$, whose entries are given by*

$$\left(\boldsymbol{\mathcal{A}}_{\langle n_i \rangle}\right)_{i_1 i_2 \cdots i_{n-1} 1\, i_{n+1} i_{n+2} \cdots i_N} = (\boldsymbol{\mathcal{A}})_{i_1 i_2 \cdots i_{n-1} i\, i_{n+1} i_{n+2} \cdots i_N}, \tag{5.4}$$

where $i \in \{1, 2, \ldots, I_n\}$ and $i_k = 1, 2, \ldots, I_k$ for $k = 1, 2, \ldots, n-1, n+1, n+2, \ldots, N$.

5.1.2 Tensor Approximation

Given a set of reduced ranks $\left\{R_n \in \{1, 2, \ldots, I_n\}\right\}_{n=1}^{N}$, where R_n is the mode-n reduced rank, tensor approximation decomposes an N-th order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ as a series of mode-n products of an N-th order core tensor $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ and a set of N column-orthonormal matrices $\left\{\mathbf{U}_n \in \mathbb{R}^{I_n \times R_n}\right\}_{n=1}^{N}$, so that the following constrained least-squares optimization problem is resolved:

$$\min_{\left\{\boldsymbol{\mathcal{Z}}, \{\mathbf{U}_n\}_{n=1}^{N}\right\}} \frac{1}{2} \left\| \boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{Z}} \bigtimes_{n=1}^{N}{}_{n} \mathbf{U}_n \right\|_F^2, \quad \text{subject to} \quad \forall n, \; \mathbf{U}_n^T \mathbf{U}_n = \mathbf{I}_{R_n} \quad \text{(orthonormal constraints)}, \tag{5.5}$$

where $\mathbf{I}_{R_n} \in \mathbb{R}^{R_n \times R_n}$ represents the identity matrix of size $R_n \times R_n$, and $\mathbf{U}_n$ is also known as the mode-n basis matrix. The constraints in Equation 5.5 enforce that each basis matrix has orthonormal columns to remove the scale ambiguity. In this way, when a mode-n reduced rank is much smaller than the corresponding mode-n rank, $\boldsymbol{\mathcal{Z}}$ and $\left\{\mathbf{U}_n\right\}_{n=1}^{N}$ will generally provide

Algorithm 5.1: N-mode singular value decomposition.

```
Procedure: $N$-SVD($\mathcal{A}$, $\{R_n\}_{n=1}^{N}$)
Input: An $N$-th order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a set of reduced ranks $\{R_n\}_{n=1}^{N}$.
Output: The core tensor $\mathcal{Z}$ and basis matrices $\{\mathbf{U}_n\}_{n=1}^{N}$.
begin
    // Initialization
    for $n \leftarrow 1$ to $N$ do
        Initialize $\mathbf{U}_n$ as the identity matrix $\mathbf{I}_{I_n}$ or the matrix whose columns are the $R_n$
        dominant left singular vectors of $uf_n(\mathcal{A})$
    end
    repeat
        // Basis matrix update
        for $n \leftarrow 1$ to $N$ do
            $\mathcal{U}_n \leftarrow \mathcal{A} \bigtimes_{j=1, j \neq n}^{N}{}_j \, \mathbf{U}_j^T$
            Update $\mathbf{U}_n$ with the $R_n$ dominant left singular vectors of $uf_n(\mathcal{U}_n)$
        end
        $\mathcal{Z} \leftarrow \mathcal{A} \bigtimes_{n=1}^{N}{}_n \, \mathbf{U}_n^T$
    until $\|\mathcal{Z}\|_F$ converges
end
```

an optimal rank-$(R_1, R_2, \ldots, R_n)$ approximation to $\mathcal{A}$ with a significant reduction in data size.

N-mode singular value decomposition (N-SVD) [33] derives a locally optimal solution to Equation 5.5 using an iterative alternating least-squares algorithm that optimizes only one basis matrix at a time, while leaving other basis matrices unchanged. At the n-th iteration, the mode-n basis matrix $\mathbf{U}_n$ is extracted by retaining the structures of the n-th mode of $\mathcal{A}$, projecting $\mathcal{A}$ onto the basis matrices of other modes, and applying SVD to the mode-n unfolded matrix of the projected tensor. The above steps are repeated until the Frobenius norm of the core tensor converges. In Figure 5.1 and Algorithm 5.1, we respectively illustrate the decomposed core tensor as well as basis matrices of a third order tensor and present the pseudo-code of N-SVD.

5.2 Mathematical Formulation

In general, tensor approximation is more effective and flexible than conventional matrix factorization methods, but its reconstruction costs are frequently too high to allow real-time performance for data-driven rendering applications. For example, when the variations of data in a tensor are large, we can not adopt small reduced ranks for decomposition to decrease the

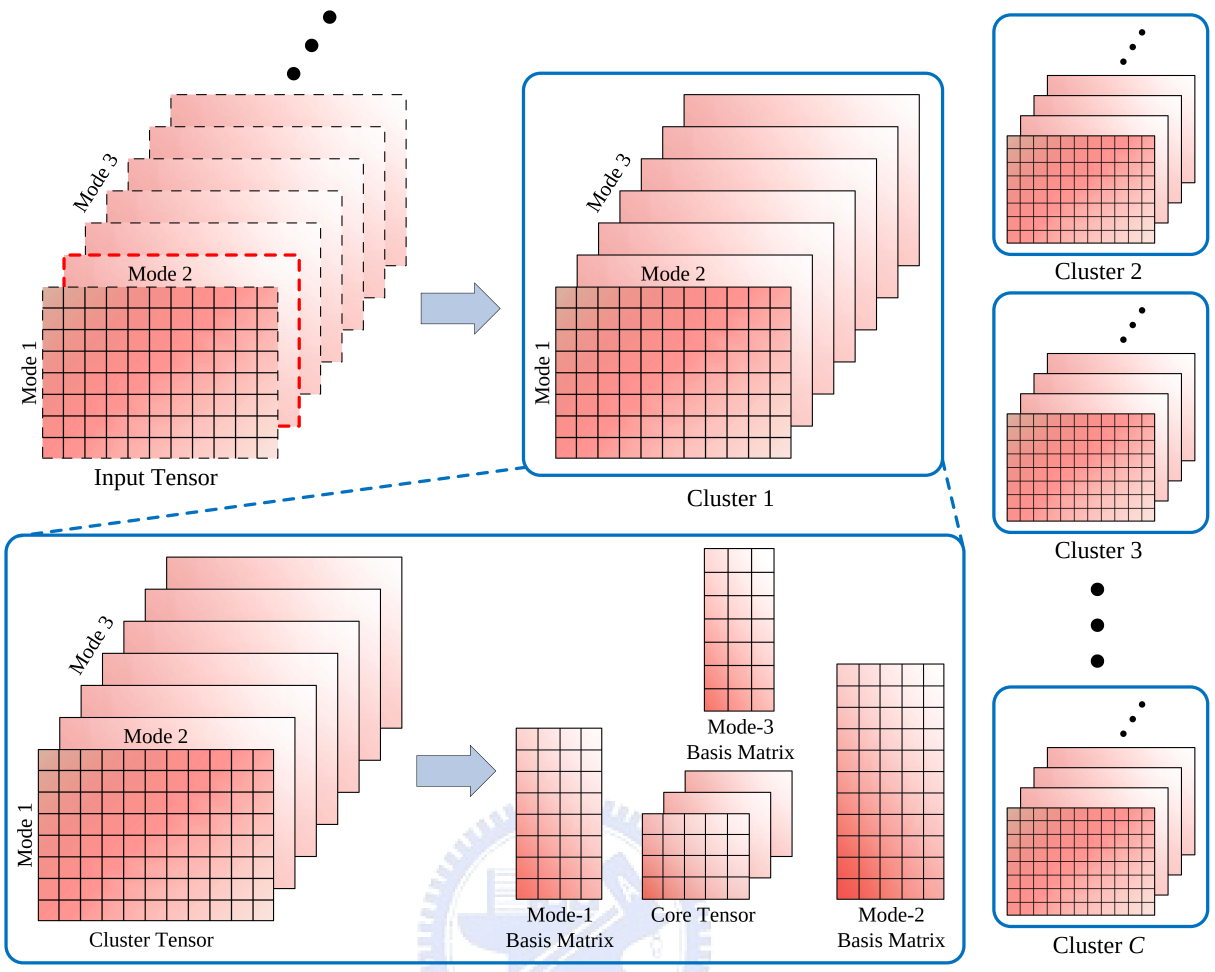


Figure 5.2: Decompose a third order tensor using CTA. Suppose that CTA is applied to a third order tensor (left top) along the third mode, and classifies mode-3 sub-tensors into total C disjoint clusters. An example of a mode-3 sub-tensor is the slice enclosed in the red rectangle. All the mode-3 sub-tensors within a cluster thus can be concatenated along the third mode to form another third order tensor. The core tensor as well as the mode-1, mode-2, and mode-3 basis matrices of each cluster are then obtained using traditional tensor approximation (left bottom).

storage space and at the same time obtain low approximation errors. Employing large reduced ranks for decomposition will certainly lead to more memory bandwidth overhead and slower rendering performance at run-time, since the reconstruction costs are roughly proportional to the values of reduced ranks. CTA thus aims at overcoming this drawback of tensor approximation by partitioning a tensor along a user-specified clustered mode, say m, into disjoint regions. The mode-m sub-tensors within a region thus form another tensor with lower variations which then can be decomposed with small reduced ranks using tensor approximation. In this way, CTA successfully combines clustering with tensor approximation to provide a powerful mathematical tool for data analysis. Figure 5.2 illustrates the main concept of CTA for a third order tensor.

Given an N-th order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, the clustered mode $m \in \{1, 2, \ldots, N\}$, and the number of disjoint clusters C, CTA can be formulated as the following constrained least-squares optimization problem:

$$\min_{\left\{\boldsymbol{\mathcal{Z}}_c, \{\mathbf{U}_{n,c}\}_{n=1}^{N}\right\}_{c=1}^{C}} \frac{1}{2}\left\|\boldsymbol{\mathcal{A}} - \sum_{c=1}^{C}\left(\boldsymbol{\mathcal{Z}}_c \bigtimes_{n=1}^{N}{}_n \mathbf{U}_{n,c}\right)\right\|_F^2,$$

$$\text{subject to} \begin{cases} \forall i,\ \sum_{c=1}^{C}\left\|\left(\mathbf{U}_{m,c}\right)_{i*}\right\|_0 = R_m & \text{(disjoint constraints)}, \\ \forall c,\ \forall i,\ \left\|\left(\mathbf{U}_{m,c}\right)_{i*}\right\|_0 \in \{0, R_m\} & \text{(membership constraints)}, \\ \forall c,\ \forall n,\ \mathbf{U}_{n,c}^T\mathbf{U}_{n,c} = \mathbf{I}_{R_n} & \text{(orthonormal constraints)}, \end{cases} \tag{5.6}$$

where $\boldsymbol{\mathcal{Z}}_c \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ and $\mathbf{U}_{n,c} \in \mathbb{R}^{I_n \times R_n}$ respectively denote the decomposed core tensor and the column-orthonormal mode-n basis matrix of cluster c, R_n specifies the mode-n reduced rank, and $\|\cdot\|_0$ represents the ℓ^0 norm of a vector. The disjoint and membership constraints in Equation 5.6 enforce that each mode-m sub-tensor is classified into only one cluster, and the entries of the i-th row of $\mathbf{U}_{m,c}$, for $i \in \{1, 2, \ldots, I_m\}$, must be all zeros if the i-th mode-m sub-tensor, namely $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$, is not classified into cluster c. Therefore, all the mode-m basis matrices are sparse and implicitly specify the cluster membership of each mode-m sub-tensor.

5.3 Algorithm

5.3.1 Overview

A simple and intuitive method for solving Equation 5.6, which is referred to as *static* CTA, is to classify mode-m sub-tensors into disjoint clusters according to a heuristic objective function, concatenate all the mode-m sub-tensors within a cluster along the m-th mode into a new tensor, and then decompose the resulting tensor of each cluster using N-SVD. Nevertheless, it is often difficult to define an objective function that is consistent with the approximation errors of tensor decomposition. The decomposed results using static CTA are only an approximate solution to Equation 5.6. We thus propose an *iterative* variant of CTA to guarantee at least a locally optimal solution by iteratively updating cluster membership based on approximation errors.

The proposed iterative CTA algorithm is an alternating least-squares approach that consists of two stages: the clustering stage (Section 5.3.2) and the update stage (Section 5.3.3). After computing the initial decomposed results of each cluster, each mode-m sub-tensor is reclassified into the cluster with the minimum approximation error in the clustering stage, while the decomposed results of each cluster are fixed. Since the number of member sub-tensors varies from cluster to cluster, straightforwardly employing the decomposed results from N-SVD would be difficult to compute the approximation error of a mode-m sub-tensor with respect to a cluster. We thus additionally compute a set of *dual* mode-m basis matrices $\left\{\mathbf{V}_{m,c}\right\}_{c=1}^{C}$ from the core tensor of each cluster to solve this issue. In the update stage, the core tensor and

Algorithm 5.2: Static and iterative clustered tensor approximation.

```
Procedure: StaticCTA($\boldsymbol{\mathcal{A}}$, $\{R_n\}_{n=1}^{N}$, $C$)
Input: An $N$-th order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, a set of reduced ranks $\{R_n\}_{n=1}^{N}$, and the
       number of clusters $C$ for the clustered mode $m$.
Output: The core tensor and basis matrices of each cluster $\left\{\boldsymbol{\mathcal{Z}}_c, \{\mathbf{U}_{n,c}\}_{n=1}^{N}\right\}_{c=1}^{C}$.
begin
    Partition $\{\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}\}_{i=1}^{I_m}$ into $C$ disjoint regions using traditional clustering methods
    for $c \leftarrow 1$ to $C$ do
        $\boldsymbol{\mathcal{A}}_c \leftarrow \boldsymbol{\mathcal{A}} \times_m \mathbf{M}_c^T$ (refer to Equations 5.9 and 5.10 for $\mathbf{M}_c$)
        $\left\{\boldsymbol{\mathcal{Z}}_c, \{\mathbf{U}_{n,c}\}_{n=1}^{N}\right\} \leftarrow N$-SVD($\boldsymbol{\mathcal{A}}_c$, $\{R_n\}_{n=1}^{N}$) (Algorithm 5.1)
        $\mathbf{U}_{m,c} \leftarrow \mathbf{M}_c \mathbf{U}_{m,c}$
    end
end

Procedure: IterativeCTA($\boldsymbol{\mathcal{A}}$, $\{R_n\}_{n=1}^{N}$, $C$)
Input: An $N$-th order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, a set of reduced ranks $\{R_n\}_{n=1}^{N}$, and the
       number of clusters $C$ for the clustered mode $m$.
Output: The core tensor and basis matrices of each cluster $\left\{\boldsymbol{\mathcal{Z}}_c, \{\mathbf{U}_{n,c}\}_{n=1}^{N}\right\}_{c=1}^{C}$.
begin
    $\left\{\boldsymbol{\mathcal{Z}}_c, \{\mathbf{U}_{n,c}\}_{n=1}^{N}\right\}_{c=1}^{C} \leftarrow$ StaticCTA($\boldsymbol{\mathcal{A}}$, $\{R_n\}_{n=1}^{N}$, $C$)  // Initialization
    repeat
        // Clustering stage
        for $c \leftarrow 1$ to $C$ do
            Compute $\mathbf{V}_{m,c}$ by Equation 5.7
        end
        for $i \leftarrow 1$ to $I_m$ do
            Re-classify $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ by solving Equation 5.8
        end
        // Update stage
        for $c \leftarrow 1$ to $C$ do
            $\boldsymbol{\mathcal{A}}_c \leftarrow \boldsymbol{\mathcal{A}} \times_m \mathbf{M}_c^T$
            $\left\{\boldsymbol{\mathcal{Z}}_c, \{\mathbf{U}_{n,c}\}_{n=1}^{N}\right\} \leftarrow N$-SVD($\boldsymbol{\mathcal{A}}_c$, $\{R_n\}_{n=1}^{N}$) (Algorithm 5.1)
            $\mathbf{U}_{m,c} \leftarrow \mathbf{M}_c \mathbf{U}_{m,c}$
        end
    until $\sum_{c=1}^{C} \left\|\boldsymbol{\mathcal{Z}}_c\right\|_F$ converges
end
```

the basis matrices of each cluster are then updated using N-SVD, while the cluster membership is fixed to satisfy the constraints in Equation 5.6. The above two stages are repeated until the sum of the Frobenius norm of each cluster core tensor converges, or a pre-defined maximum iteration count is reached. In Algorithm 5.2, the pseudo-code of the proposed static and iterative CTA algorithms is presented in details.

5.3.2 Clustering Stage

In this stage, we would like to re-classify each mode-m sub-tensor into the cluster with the minimum approximation error from the core tensor and basis matrices of each cluster. To facilitate the computation of approximation errors for the mode-m sub-tensors outside cluster c, the dual mode-m basis matrix $\mathbf{V}_{m,c} \in \mathbb{R}^{R_m \times (R_{m+1}R_{m+2}\cdots R_N R_1 R_2 \cdots R_{m-1})}$ of cluster c is additionally derived as

$$\mathbf{V}_{m,c} = \Big(uf_m\big(\boldsymbol{\mathcal{Z}}_c\big)\, uf_m\big(\boldsymbol{\mathcal{Z}}_c\big)^T\Big)^{-\frac{1}{2}} uf_m\big(\boldsymbol{\mathcal{Z}}_c\big). \tag{5.7}$$

Theoretically, if all the decomposed basis matrices of cluster c indeed converge, the rows of $uf_m\big(\boldsymbol{\mathcal{Z}}_c\big)$ should be orthogonal. Equation 5.7 is thus equivalent to the normalization of each row of $uf_m\big(\boldsymbol{\mathcal{Z}}_c\big)$. Nevertheless, in practice the N-SVD algorithm may not converge when a user-specified maximum iteration count is reached. it is recommended to compute the dual mode-m basis matrix as Equation 5.7 by using SVD.

With the help of dual mode-m basis matrices, we propose the following theorem to re-classify each mode-m sub-tensor:

Theorem 5.3: *Let $\mathbf{V}_{m,c} \in \mathbb{R}^{R_m \times (R_{m+1}R_{m+2}\cdots R_N R_1 R_2 \cdots R_{m-1})}$ be the dual mode-m basis matrix of cluster c as defined in Equation 5.7. The cluster c_i with the minimum approximation error for the i-th mode-m sub-tensor $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ is obtained by solving the following constrained integer optimization problem:*

$$\max_{c_i} \frac{1}{2} \left\| uf_m\Big(\boldsymbol{\mathcal{A}}_{\langle m_i \rangle} \bigtimes_{\substack{n=1 \\ n\neq m}}^{N}{}_n\, \mathbf{U}_{n,c_i}^T\Big) \mathbf{V}_{m,c_i}^T \right\|_F^2, \quad \text{subject to} \quad c_i \in \{1, 2, \ldots, C\}, \tag{5.8}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix.

Theorem 5.3 states that $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ should be classified into a cluster whose basis matrices adequately preserve its projected norm, so that the mode-m sub-tensors within a cluster are correlated with each other. Interested readers may refer to Section 5.5 for the mathematical proof of Theorem 5.3.

5.3.3 Update Stage

Given the cluster membership of each mode-m sub-tensor, the core tensor and the basis matrices of each cluster are updated using N-SVD in this stage. Since the partitioned clusters are independent of each other, their decomposed results can be separately computed. Nevertheless, to satisfy the orthonormal constraints on mode-m basis matrices in Equation 5.6, only the member sub-tensors of a cluster should be considered during tensor decomposition.

More formally, let M_c be the membership index set of cluster c defined as

$$M_c = \left\{ i \in \{1, 2, \ldots, I_m\} \mid \boldsymbol{\mathcal{A}}_{\langle m_i \rangle} \text{ is a member of cluster } c \right\}, \tag{5.9}$$

and $\mathbf{M}_c \in \mathbb{R}^{I_m \times |M_c|}$ denotes the membership matrix of cluster c, whose entries are

$$\forall i_1,\ \forall i_2,\ \left(\mathbf{M}_c\right)_{i_1 i_2} = \begin{cases} 1, & \text{if } i_1 = (M_c)_{i_2}, \\ 0, & \text{otherwise}, \end{cases} \tag{5.10}$$

where $|\cdot|$ denotes the cardinality of a set, and $(M_c)_{i_2}$ is the i_2-th element of the membership index set M_c. We thus can extract the member sub-tensors of cluster c into an N-th order tensor $\boldsymbol{\mathcal{A}}_c \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{m-1} \times |M_c| \times I_{m+1} \times I_{m+2} \times \cdots \times I_N}$ by the following equation:

$$\boldsymbol{\mathcal{A}}_c = \boldsymbol{\mathcal{A}} \times_m \mathbf{M}_c^T. \tag{5.11}$$

When applying tensor approximation to the *shrunken* tensor $\boldsymbol{\mathcal{A}}_c$, non-members of cluster c are excluded from the decomposition. $\boldsymbol{\mathcal{Z}}_c$ and $\left\{\mathbf{U}_{n,c}\right\}_{n=1}^{N}$ are then updated by the decomposed core tensor and basis matrices of $\boldsymbol{\mathcal{A}}_c$. Note that since $\boldsymbol{\mathcal{A}}_c$ contains only members of cluster c, $\mathbf{U}_{m,c}$ should be further updated by the multiplication $\mathbf{M}_c\mathbf{U}_{m,c}$ to satisfy the constraints in Equation 5.6.

5.4 Implementation Issues

5.4.1 Initial Guess

One significant implementation issue of static and iterative CTA is the initial guess of cluster membership of each mode-m sub-tensor. According to the experiments, we have found that applying some advanced clustering methods, such as enhanced LBG [133] and local (or clustered) principal component analysis [75, 161], to determine the initial cluster membership is often much better than using conventional K-means clustering. Although this scheme will push the problem back to the initial seeds for clustering methods, various techniques for generating and fixing the initial cluster seeds usually provide satisfactory results in our experience.

The best approach may vary with the given data sets, but here we present a general method as a guideline to determine the initial cluster seeds. For a real-world data set, each mode of the input tensor is associated with a parametric space that describes its physical conditions, for example, different illumination or view directions for a reflectance function. Based on the assumption that observations from nearby parameters in the mode-m parametric space are expected to be highly correlated, we can perform initial clustering in the parametric space instead. This is computationally efficient since the dimensionality of the mode-m parametric space is frequently lower than that of observations. Sophisticated or even exhaustive methods therefore can be employed to generate appropriate cluster seeds. In our experiments, this scheme generally reduces the final approximation errors of CTA by $3\% \sim 8\%$, when compared with directly performing K-means clustering on mode-m sub-tensors to determine the initial cluster seeds for CTA.

5.4.2 Global Basis Matrices

Sometimes a single *global* mode-n basis matrix of all clusters is preferred rather than an individual *local* mode-n basis matrix of each cluster. The preference for global basis matrices may be due to computational costs, storage space, or the special purpose of an application. To account for this issue, the global basis matrices are computed by decomposing an N-th order tensor $\boldsymbol{\mathcal{A}}$ before applying CTA.

Let G be the index set of global basis matrices $\{\mathbf{U}_n\}_{n\in G}$, where the subscript c of a local mode-n basis matrix is omitted to denote a global one. After extracting $\{\mathbf{U}_n\}_{n\in G}$ by applying tensor approximation to $\boldsymbol{\mathcal{A}}$, we project $\boldsymbol{\mathcal{A}}$ onto $\{\mathbf{U}_n\}_{n\in G}$ to obtain an N-th order reduced tensor $\boldsymbol{\mathcal{A}}_G$ as

$$\boldsymbol{\mathcal{A}}_G = \boldsymbol{\mathcal{A}} \bigtimes_{n\in G} {}_n \mathbf{U}_n^T. \tag{5.12}$$

CTA is then performed on $\boldsymbol{\mathcal{A}}_G$ to compute the core tensor and the local basis matrices of each cluster, while the global basis matrices are fixed.

Although an iterative algorithm can be employed to alternately update the global basis matrices and the decomposed results of CTA, it is computationally too expensive and only reduces approximation errors by a small amount. We therefore just performed the initial tensor decomposition on $\boldsymbol{\mathcal{A}}$ to derive the global basis matrices and did not update them after CTA for all experimental results in this dissertation.

5.5 Mathematical Proofs

Re-Classification in the Clustering Stage

In the clustering stage of CTA, the mode-m sub-tensors of $\boldsymbol{\mathcal{A}}$ are re-classified into C disjoint clusters, while the core tensor and basis matrices, other than the mode-m basis matrix, of each cluster are fixed. We thus introduced Theorem 5.3 in Section 5.3.1 to determine the cluster membership of each mode-m sub-tensor with strong intra-cluster correlations. In this section, the mathematical proof of Theorem 5.3 is presented in details to justify the correctness of the proposed CTA algorithm.

Proof of Theorem 5.3: Since the cluster membership of a mode-m sub-tensor does not depend on that of others, Equation 5.6, without the orthonormal constraints on $\mathbf{U}_{m,1}, \mathbf{U}_{m,2}, \ldots, \mathbf{U}_{m,C}$, can be separated into I_m distinct constrained least-squares optimization sub-problems as

$$\min_{\left\{(\mathbf{U}_{m,c})_{i*}\right\}_{c=1}^{C}} \frac{1}{2}\left\| \boldsymbol{\mathcal{A}}_{\langle m_i\rangle} - \sum_{c=1}^{C}\left(\boldsymbol{\mathcal{Z}}_c \times_m \left(\mathbf{U}_{m,c}\right)_{i*} \bigtimes_{\substack{n=1\\ n\neq m}}^{N}{}_{n} \mathbf{U}_{n,c}\right)\right\|_F^2,$$
$$\text{subject to} \begin{cases} \sum_{c=1}^{C} \left\|\left(\mathbf{U}_{m,c}\right)_{i*}\right\|_0 = R_m & \text{(disjoint constraint)}, \\ \forall c,\ \left\|\left(\mathbf{U}_{m,c}\right)_{i*}\right\|_0 \in \{0, R_m\} & \text{(membership constraints)}, \end{cases} \tag{5.13}$$

for $i = 1, 2, \ldots, I_m$. After solving all the optimization sub-problems, we should finally transform each derived mode-m basis matrix by a matrix of size $R_m \times R_m$ to satisfy the orthonormal constraints in Equation 5.6. Since this transformation actually resembles a rotation in the cluster sub-space associated with the m-th mode, one can always find such a transformation matrix without changing the final approximation errors. Therefore, the remaining issue of this proof is to show that the solution to Equation 5.13 will implicitly provide a solution to Equation 5.8.

Without losing generality, suppose that $\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}$ is re-classified into cluster c_i. Since the constraints in Equation 5.13 can be enforced by setting $\left(\mathbf{U}_{m,c}\right)_{i*}$ to zeros for all $c \neq c_i$, the objective function in Equation 5.13 becomes

$$\begin{aligned} &\frac{1}{2}\left\| \boldsymbol{\mathcal{A}}_{\langle m_i\rangle} - \boldsymbol{\mathcal{Z}}_{c_i} \times_m \left(\mathbf{U}_{m,c_i}\right)_{i*} \bigtimes_{\substack{n=1\\ n\neq m}}^{N}{}_{n} \mathbf{U}_{n,c_i}\right\|_F^2 \\ &= \frac{1}{2}\left\| uf_m\!\left(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\right) - \left(\mathbf{U}_{m,c_i}\right)_{i*} uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_i} \bigtimes_{\substack{n=1\\ n\neq m}}^{N}{}_{n} \mathbf{U}_{n,c_i}\right)\right\|_F^2, \end{aligned} \tag{5.14}$$

where the right side comes from the definitions of the mode-n unfolded matrix and the mode-n product. Furthermore, the definition of Frobenius norm allows us to rewrite Equation 5.14 into a standard least-squares objective function as follows:

$$\frac{1}{2}\left\| uf_m\big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\big) - \big(\mathbf{U}_{m,c_i}\big)_{i*} uf_m\Big(\boldsymbol{\mathcal{Z}}_{c_i} \bigtimes_{\substack{n=1\\n\neq m}}^{N}{}_n \mathbf{U}_{n,c_i}\Big)\right\|_2^2. \tag{5.15}$$

Therefore, the optimal solution of $\big(\mathbf{U}_{m,c_i}\big)_{i*}$ to Equation 5.15 is the least-squares estimation of the solution to the following linear equation:

$$uf_m\big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\big) = \big(\mathbf{U}_{m,c_i}\big)_{i*} uf_m\Big(\boldsymbol{\mathcal{Z}}_{c_i} \bigtimes_{\substack{n=1\\n\neq m}}^{N}{}_n \mathbf{U}_{n,c_i}\Big) = \big(\mathbf{U}_{m,c_i}\big)_{i*} uf_m\big(\boldsymbol{\mathcal{Z}}_{c_i}\big)\Big(\bigotimes_{\substack{n=1\\n\neq m}}^{N} \mathbf{U}_{n,c_i}^T\Big), \tag{5.16}$$

where

$$\bigotimes_{\substack{n=1\\n\neq m}}^{N} \mathbf{U}_{n,c_i}^T = \mathbf{U}_{1,c_i}^T \otimes \mathbf{U}_{2,c_i}^T \otimes \cdots \otimes \mathbf{U}_{m-1,c_i}^T \otimes \mathbf{U}_{m+1,c_i}^T \otimes \mathbf{U}_{m+2,c_i}^T \otimes \cdots \otimes \mathbf{U}_{N,c_i}^T \tag{5.17}$$

denotes a series of Kronecker products, and the symbol $\otimes$ represents the Kronecker product operator. Equation 5.16 comes from the relation between the mode-n product and the Kronecker product [33]. Since each basis matrix has orthonormal columns, the least-squares solution to Equation 5.15 is

$$\begin{aligned}
\big(\mathbf{U}_{m,c_i}\big)_{i*} &= uf_m\big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\big)\bigg(uf_m\big(\boldsymbol{\mathcal{Z}}_{c_i}\big)\Big(\bigotimes_{\substack{n=1\\n\neq m}}^{N} \mathbf{U}_{n,c_i}^T\Big)\bigg)^{+} \\
&= uf_m\big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\big)\Big(\bigotimes_{\substack{n=1\\n\neq m}}^{N} \mathbf{U}_{n,c_i}^T\Big)^{+} uf_m\big(\boldsymbol{\mathcal{Z}}_{c_i}\big)^{+} \\
&= uf_m\big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\big)\Big(\bigotimes_{\substack{n=1\\n\neq m}}^{N} \mathbf{U}_{n,c_i}\Big) uf_m\big(\boldsymbol{\mathcal{Z}}_{c_i}\big)^{+} \\
&= uf_m\Big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle} \bigtimes_{\substack{n=1\\n\neq m}}^{N}{}_n \mathbf{U}_{n,c_i}^T\Big) uf_m\big(\boldsymbol{\mathcal{Z}}_{c_i}\big)^{+},
\end{aligned} \tag{5.18}$$

where the superscript '$+$' specifies the Moore-Penrose pseudo-inverse of a matrix.

After substituting Equation 5.18 for $\big(\mathbf{U}_{m,c_i}\big)_{i*}$ in Equation 5.15, we have

$$\frac{1}{2}\left\| uf_m\big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\big) - uf_m\Big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle} \bigtimes_{\substack{n=1\\n\neq m}}^{N}{}_n \mathbf{U}_{n,c_i}^T\Big)\big(\boldsymbol{\mathcal{Z}}_{c_i}\big)^{+} uf_m\Big(\boldsymbol{\mathcal{Z}}_{c_i} \bigtimes_{\substack{n=1\\n\neq m}}^{N}{}_n \mathbf{U}_{n,c_i}\Big)\right\|_2^2$$

$$
\begin{aligned}
&= \frac{1}{2}\left\| uf_m\big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\big) - uf_m\Big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle} \mathop{\times_n}_{\substack{n=1\\ n\neq m}}^{N} \mathbf{U}_{n,c_i}^T\Big)\big(\boldsymbol{\mathcal{Z}}_{c_i}\big)^{+} uf_m\big(\boldsymbol{\mathcal{Z}}_{c_i}\big)\Big(\bigotimes_{\substack{n=1\\ n\neq m}}^{N} \mathbf{U}_{n,c_i}^T\Big)\right\|_2^2 \\
&= \frac{1}{2}\left\| uf_m\big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\big) - uf_m\Big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle} \mathop{\times_n}_{\substack{n=1\\ n\neq m}}^{N} \mathbf{U}_{n,c_i}^T\Big)\mathbf{V}_{m,c_i}^T\mathbf{V}_{m,c_i}\Big(\bigotimes_{\substack{n=1\\ n\neq m}}^{N} \mathbf{U}_{n,c_i}\Big)^T\right\|_2^2 \\
&= \frac{1}{2}\left\| uf_m\big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\big)\right\|_2^2 - uf_m\big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\big)\Big(\bigotimes_{\substack{n=1\\ n\neq m}}^{N} \mathbf{U}_{n,c_i}\Big)\mathbf{V}_{m,c_i}^T\mathbf{V}_{m,c_i}\, uf_m\Big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle} \mathop{\times_n}_{\substack{n=1\\ n\neq m}}^{N} \mathbf{U}_{n,c_i}^T\Big)^T \\
&\quad + \frac{1}{2}\left\| uf_m\Big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle} \mathop{\times_n}_{\substack{n=1\\ n\neq m}}^{N} \mathbf{U}_{n,c_i}^T\Big)\mathbf{V}_{m,c_i}^T\mathbf{V}_{m,c_i}\Big(\bigotimes_{\substack{n=1\\ n\neq m}}^{N} \mathbf{U}_{n,c_i}\Big)^T\right\|_2^2 \qquad (5.19)\\
&= \frac{1}{2}\left\|\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\right\|_F^2 - uf_m\Big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle} \mathop{\times_n}_{\substack{n=1\\ n\neq m}}^{N} \mathbf{U}_{n,c_i}^T\Big)\mathbf{V}_{m,c_i}^T\left(uf_m\Big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle} \mathop{\times_n}_{\substack{n=1\\ n\neq m}}^{N} \mathbf{U}_{n,c_i}^T\Big)\mathbf{V}_{m,c_i}^T\right)^T \\
&\quad + \frac{1}{2}\left\| uf_m\Big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle} \mathop{\times_n}_{\substack{n=1\\ n\neq m}}^{N} \mathbf{U}_{n,c_i}^T\Big)\mathbf{V}_{m,c_i}^T\right\|_2^2 \qquad (5.20)\\
&= \frac{1}{2}\left\|\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\right\|_F^2 - \left\| uf_m\Big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle} \mathop{\times_n}_{\substack{n=1\\ n\neq m}}^{N} \mathbf{U}_{n,c_i}^T\Big)\mathbf{V}_{m,c_i}^T\right\|_2^2 + \frac{1}{2}\left\| uf_m\Big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle} \mathop{\times_n}_{\substack{n=1\\ n\neq m}}^{N} \mathbf{U}_{n,c_i}^T\Big)\mathbf{V}_{m,c_i}^T\right\|_2^2 \\
&= \frac{1}{2}\left\|\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\right\|_F^2 - \frac{1}{2}\left\| uf_m\Big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle} \mathop{\times_n}_{\substack{n=1\\ n\neq m}}^{N} \mathbf{U}_{n,c_i}^T\Big)\mathbf{V}_{m,c_i}^T\right\|_F^2 . \qquad (5.21)
\end{aligned}
$$

From Equation 5.19 to Equation 5.20, we use the facts that the Kronecker product of two column-orthonormal matrices is another column-orthonormal matrix, and the ℓ^2 norm of a row vector $\mathbf{v}^T$ is unchanged under the transformation $\mathbf{v}^T\mathbf{U}$ for a row-orthonormal matrix $\mathbf{U}$, namely $\left\|\mathbf{v}^T\mathbf{U}\right\|_2 = \left\|\mathbf{v}^T\right\|_2$. Since the first term in Equation 5.21 is a constant, the minimization of Equation 5.21 is equivalent to the maximization of the second term in Equation 5.21. Theorem 5.3 thus is proved by identifying that the cluster membership is implicitly specified in the solution to Equation 5.13. □

Chapter 6

K-Clustered Tensor Approximation

Recently, there has been a growing interest in modeling real-world observations as sparse linear combinations of atoms (or basis functions) in an *over-complete* dictionary. Although the underlying physical process of a natural phenomenon may be a complex function or mixture of heterogeneous elements, it is frequently desirable to represent observations in a *sparse* form that allows efficient data analysis. However, this intuitive concept is far from easy to achieve in practice. Even with a fixed dictionary, searching for an optimal solution in which each signal exactly depends on a given number of atoms was proved to be NP-hard [31]. Therefore, many practical algorithms instead consider sub-optimal solutions, such as matching pursuit [110, 148], basis pursuit [19], and Bayesian models [83, 102].

In addition to pursuit algorithms and Bayesian models, previous studies have also reported a close connection between sparse representation and vector quantization [83, 177]. *K-singular value decomposition* (K-SVD) [3, 4] thus generalized K-means clustering to seek sparse representations by alternating between the pursuit process and dictionary learning. Nevertheless, rather than using Bayesian inference, it applied *singular value decomposition* (SVD) to simultaneously update dictionary atoms and non-zero basis coefficients in the dictionary learning stage, so that convergence rate could be improved. Interested readers may refer to [3] for more details about K-SVD and a comprehensive review on recent progress of sparse representation.

In this chapter, we introduce a novel sparse multi-linear model, namely *K-clustered tensor approximation* (K-CTA), which combines the advantages of *clustered tensor approximation* (CTA) and K-SVD to bridge the gap between sparse representation and tensor approximation. Several important theorems of K-CTA are discussed and proved to demonstrate that K-CTA in fact resembles the behaviors of K-SVD in the tensor space and is a natural extension of CTA. As a result, CTA is further generalized to provide a sparse multi-dimensional data analysis tool in which the sparsity is under user control.

6.1 Mathematical Formulation

One major drawback of CTA for decomposing an N-th order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ along the clustered mode m is that it enforces *hard* clustering in which each mode-m sub-tensor is classified into just one cluster. The subsequent tensor approximation within each cluster thus can only exploit intra-cluster coherence. In addition, the performance of CTA heavily depends on the initial guess of cluster membership, but estimating an appropriate initial guess is a non-trivial problem at all. Even if the globally optimal solution to hard clustering could be easily found, the decomposed core tensors and basis matrices of different clusters may still have strong correlations.

As illustrated in Figure 6.1, our solution to this issue is to relax the hard clustering constraint into a *soft* one. Each mode-m sub-tensor now can be classified into more than one cluster and approximated by mixing the decomposed results of these clusters. To reduce run-time reconstruction costs, this soft constraint should also be a sparse one in which each mode-m sub-tensor belongs to just a few, say K_m, clusters. This not only permits K-CTA to exploit the inter-cluster coherence that can not be analyzed by CTA, but also alleviates the influence of an inappropriate initial guess by breaking the hard cluster boundaries. K-CTA thus can be regarded as a sparse extension of CTA and a multi-linear generalization of K-SVD [3, 4].

To allow soft and sparse clustering for K-CTA, Equation 5.6 is reformulated into the following constrained least-squares optimization problem:

$$\min_{\left\{\boldsymbol{\mathcal{Z}}_c, \{\mathbf{U}_{n,c}\}_{n=1}^{N}\right\}_{c=1}^{C}} \frac{1}{2}\left\|\boldsymbol{\mathcal{A}} - \sum_{c=1}^{C}\left(\boldsymbol{\mathcal{Z}}_c \bigtimes_{n=1}^{N}{}_{n} \mathbf{U}_{n,c}\right)\right\|_F^2,$$

$$\text{subject to} \begin{cases} \forall i,\ \sum_{c=1}^{C}\left\|\left(\mathbf{U}_{m,c}\right)_{i*}\right\|_0 = K_m R_m & \text{(sparse constraints)}, \\ \forall c,\ \forall i,\ \left\|\left(\mathbf{U}_{m,c}\right)_{i*}\right\|_0 \in \{0, R_m\} & \text{(membership constraints)}, \\ \forall c,\ \forall n,\ \mathbf{U}_{n,c}^T \mathbf{U}_{n,c} = \mathbf{I}_{R_n} & \text{(orthonormal constraints)}, \end{cases} \tag{6.1}$$

where K_m specifies the number of mixture clusters for a mode-m sub-tensor. For K-CTA, the mode-m basis matrix $\mathbf{U}_{m,c}$ is also called the mixing matrix of cluster c whose i-th row contains the mixing coefficients of the i-th mode-m sub-tensor $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ with respect to cluster c. The only difference between Equation 5.6 and Equation 6.1 is that the sparse and membership constraints of K-CTA enforce that each mode-m sub-tensor belongs to exact K_m clusters instead of only one. As a result, the decomposed mode-m basis matrices are still sparse for K-CTA, and their sparsity is totally controllable by the value of K_m. Note that Equation 6.1 will be exactly equivalent to Equation 5.6 when $K_m = 1$. In this case, K-CTA should derive the same results as CTA to permit it as a natural generalization of CTA.

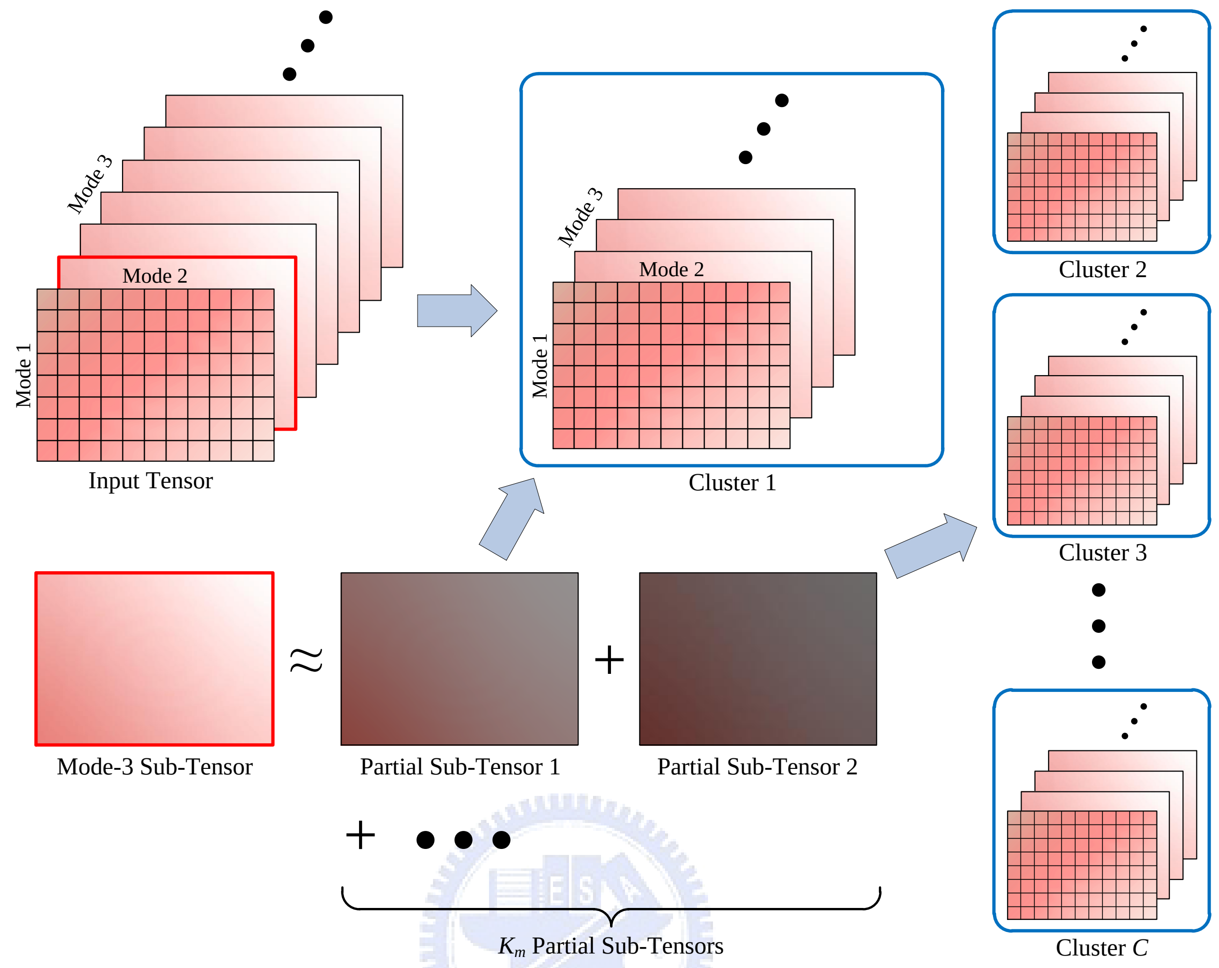


Figure 6.1: Decompose a third order tensor using K-CTA. Suppose that we intend to classify the mode-3 sub-tensors of a third order tensor (left top) into total C clusters. A mode-3 sub-tensor (enclosed in the red rectangle) is approximated as the sum of K_m partial sub-tensors, each of which is classified into one different cluster. All the partial sub-tensors within a cluster thus can be concatenated along the third mode and decomposed using traditional tensor approximation.

6.2 Algorithm

6.2.1 Overview

Although the mathematical formulation of K-CTA is almost identical to that of CTA, the practical algorithm for solving Equation 6.1 is substantially different from static and iterative CTA. For a small total number of clusters, while an exhaustive approach for finding the optimal solution to a hard clustering problem may be efficient in practice, a brute-force method for soft clustering is frequently time-consuming and only feasible when the number of mixture clusters is rather small. It is therefore difficult to extend CTA into K-CTA with just minor modifications, but the fundamental alternating framework of CTA is still applicable.

Similar to CTA, the proposed iterative K-CTA algorithm also consists of two stages: the clustering stage (Section 6.2.2) and the update stage (Section 6.2.3). After initializing the core tensor and basis matrices of each cluster, all variables in Equation 6.1 are fixed in the clustering stage, except for the mode-m basis matrix of each cluster. For each mode-m sub-tensor, a greedy approach is then applied to sequentially search for the best K_m mixture clusters that minimize its approximation error. The mixing coefficients, namely the rows of the mode-m basis matrix of each cluster, are also updated by the proposed optimal projection method. Since the mode-m basis matrices derived by the optimal projection method do not have orthonormal columns, we should additionally update them, together with the core tensor of each cluster, to satisfy the orthonormal constraints in Equation 6.1. In the update stage, the core tensor and basis matrices of each cluster are then iteratively updated using N-SVD, one cluster at a time, while those of other clusters are unchanged. Note that we also fix the cluster membership in the update stage to simplify the proposed algorithm. Finally, the above two stages are iteratively executed until the sum of the Frobenius norm of each cluster core tensor converges, or a user-specified maximum iteration count is reached. The whole process of iterative K-CTA is summarized in Algorithm 6.1. Here, we omit the description of static K-CTA, since it just follows the process of iterative K-CTA without repeating the clustering stage and the update stage.

6.2.2 Clustering Stage

Given the core tensor and basis matrices of each cluster, we would like to find the best K_m mixture clusters for each mode-m sub-tensor and derive the corresponding mixing coefficients in this stage, so that the sparse and membership constraints in Equation 6.1 are satisfied. This issue can be considered as a multi-linear counterpart of the pursuit problem for sparse representation [3, 4, 19, 110, 148], which was proved to be NP-hard [31]. Indeed, it is difficult to solve the cluster membership and the non-zero mixing coefficients of a mode-m sub-tensor at the same time, since the decomposed results of different clusters are frequently correlated. Even a single change in the membership of a mode-m sub-tensor, either adding the sub-tensor to or removing it from a cluster, will affect its mixing coefficients for other clusters. Nevertheless, greedy approaches often provide a satisfactory approximate solution to the pursuit problem in both theory and practice [31, 176]. We thus sequentially update the cluster membership and sparse mixing coefficients of all the mode-m sub-tensors, one cluster at a time.

Greedy Search

First mixture cluster: To allow K-CTA as a natural generalization of CTA, the first mixture cluster of a mode-m sub-tensor is obtained by using the same method in the clustering stage of CTA. From Theorem 5.3, we can determine the first mixture cluster c_{i_1} of the i-th mode-m sub-tensor $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ by solving the following constrained integer optimization problem:

$$\max_{c_{i_1}} \frac{1}{2} \left\| uf_m\!\left(\boldsymbol{\mathcal{A}}_{\langle m_i \rangle} \bigtimes_{\substack{n=1 \\ n \neq m}}^{N}{}_n \, \mathbf{U}_{n,c_{i_1}}^T \right) \mathbf{V}_{m,c_{i_1}}^T \right\|_F^2, \quad \text{subject to} \;\; c_{i_1} \in \{1, 2, \ldots, C\}\,. \qquad (6.2)$$

Algorithm 6.1: Iterative K-clustered tensor approximation.

```
Procedure: IterativeK-CTA($\boldsymbol{\mathcal{A}}$, $\{R_n\}_{n=1}^{N}$, $C$, $K_m$)
Input: An $N$-th order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, a set of reduced ranks $\{R_n\}_{n=1}^{N}$, the
       number of clusters $C$ for the clustered mode $m$, and the number of mixture
       clusters $K_m$.
Output: The core tensor and basis matrices of each cluster $\left\{\boldsymbol{\mathcal{Z}}_c, \{\mathbf{U}_{n,c}\}_{n=1}^{N}\right\}_{c=1}^{C}$.

begin
    // Initialization
    $\left\{\boldsymbol{\mathcal{Z}}_c, \{\mathbf{U}_{n,c}\}_{n=1}^{N}\right\}_{c=1}^{C} \leftarrow$ StaticCTA($\boldsymbol{\mathcal{A}}$, $\{R_n\}_{n=1}^{N}$, $C$) (Algorithm 5.2)
    repeat
        // Clustering stage
        for $c \leftarrow 1$ to $C$ do
            Compute $\mathbf{V}_{m,c}$ by Equation 5.7
            Initialize each entry of $\mathbf{U}_{m,c}$ to zero
        end
        for $i \leftarrow 1$ to $I_m$ do  // Greedy search
            Obtain the first mixture cluster $c_{i_1}$ of $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ by solving Equation 6.2
            Update $\left(\mathbf{U}_{m,c_{i_1}}\right)_{i*}$ as Equation 6.3
            for $k \leftarrow 2$ to $K_m$ do
                Obtain the $k$-th mixture cluster $c_{i_k}$ of $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ by solving Equation 6.5
                Update $\left\{\left(\mathbf{U}_{m,c_{i_j}}\right)_{i*}\right\}_{j=1}^{k}$ as Equation 6.6  // Optimal projection
            end
        end
        for $c \leftarrow 1$ to $C$ do  // Post-processing
            Decompose $\mathbf{U}_{m,c}$ to obtain $\mathbf{U}'_{m,c}$ and $\mathbf{W}_c$ (Equation 6.10)
            $\mathbf{U}_{m,c} \leftarrow \mathbf{U}'_{m,c}$
            $\boldsymbol{\mathcal{Z}}_c \leftarrow \boldsymbol{\mathcal{Z}}_c \times_m \mathbf{W}_c$
        end

        // Update stage
        for $c \leftarrow 1$ to $C$ do
            Compute $\boldsymbol{\mathcal{R}}_c$ as Equation 6.12
            $\boldsymbol{\mathcal{R}}'_c \leftarrow \boldsymbol{\mathcal{R}}_c \times_m \mathrm{M}_c^T$ (refer to Equations 5.9 and 5.10 for $\mathrm{M}_c$)
            $\left\{\boldsymbol{\mathcal{Z}}_c, \{\mathbf{U}_{n,c}\}_{n=1}^{N}\right\} \leftarrow N$-SVD($\boldsymbol{\mathcal{R}}'_c$, $\{R_n\}_{n=1}^{N}$) (Algorithm 5.1)
            $\mathbf{U}_{m,c} \leftarrow \mathbf{M}_c \mathbf{U}_{m,c}$
        end
    until $\sum_{c=1}^{C} \left\|\boldsymbol{\mathcal{Z}}_c\right\|_F$ converges
end
```

Then, the corresponding mixing coefficients for the first mixture cluster of a mode-m sub-tensor are given by the following corollary:

Corollary 6.1: *The mixing coefficients for the first mixture cluster c_{i_1} of the i-th mode-m sub-tensor $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ are computed as*

$$\left(\mathbf{U}_{m,c_{i_1}}\right)_{i*} = uf_m\!\left(\boldsymbol{\mathcal{A}}_{\langle m_i \rangle} \bigtimes_{\substack{n=1\\n\neq m}}^{N}{}_n \mathbf{U}^T_{n,c_{i_1}}\right) uf_m\!\left(\boldsymbol{\mathcal{Z}}_c\right)^+, \tag{6.3}$$

where the superscript '$+$' specifies the Moore-Penrose pseudo-inverse of a matrix.

Interested readers may refer to Section 6.4.1 for the mathematical proof of Corollary 6.1. From Theorem 5.3, we identify that correlated mode-m sub-tensors will likely be classified into the same cluster. Corollary 6.1 further indicates that the mixing coefficients for the first mixture cluster of the i-th mode-m sub-tensor is the projection coefficients of $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ onto the sub-spaces of cluster c_{i_1}. Equation 6.2 also implies that if the total number of mixture clusters is set to one, K-CTA will be identical to CTA in the clustering stage.

Remaining mixture clusters: After resolving the first mixture cluster of a mode-m sub-tensor, its remaining mixture clusters are then iteratively derived, one cluster at each iteration, from the results of previous iteration. We thus propose the following theorem to settle the k-th mixture cluster of a mode-m sub-tensor:

Theorem 6.2: *The k-th mixture cluster c_{i_k} of the i-th mode-m sub-tensor $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ is resolved from the mixing coefficients for the previously selected $k-1$ mixture clusters, $c_{i_1}, c_{i_2}, \ldots, c_{i_{k-1}}$, by minimizing the approximation error of the residual sub-tensor*

$$\boldsymbol{\mathcal{R}}^{(k)}_{\langle m_i \rangle} = \boldsymbol{\mathcal{A}}_{\langle m_i \rangle} - \sum_{j=1}^{k-1} \left(\boldsymbol{\mathcal{Z}}_{c_{i_j}} \times_m \left(\mathbf{U}_{m,c_{i_j}}\right)_{i*} \bigtimes_{\substack{n=1\\n\neq m}}^{N}{}_n \mathbf{U}_{n,c_{i_j}}\right), \tag{6.4}$$

which is equivalent to solving the following constrained integer optimization problem:

$$\max_{c_{i_k}} \frac{1}{2} \left\| uf_m\!\left(\boldsymbol{\mathcal{A}}_{\langle m_i \rangle} \bigtimes_{\substack{n=1\\n\neq m}}^{N}{}_n \mathbf{U}^T_{n,c_{i_k}}\right) \mathbf{V}^T_{m,c_{i_k}} - \sum_{j=1}^{k-1} uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_{i_j}} \times_m \left(\mathbf{U}_{m,c_{i_j}}\right)_{i*} \bigtimes_{\substack{n=1\\n\neq m}}^{N}{}_n \mathbf{U}^T_{n,c_{i_k}} \mathbf{U}_{n,c_{i_j}}\right) \mathbf{V}^T_{m,c_{i_k}} \right\|_F^2, \tag{6.5}$$

$$\text{subject to} \quad c_{i_k} \in \{1, 2, \ldots, C\}, \; c_{i_k} \notin \{c_{i_1}, c_{i_2}, \ldots, c_{i_{k-1}}\}.$$

The mathematical proof of Theorem 6.2 can be found in Section 6.4.1. For each mode-m sub-tensor, the first term of the objective function in Equation 6.5 is actually the same as the objective function in Equation 6.2, and the second term instead penalizes a cluster whose basis matrices are correlated to those of previously selected mixture clusters. Therefore, Theorem 6.2 implies that the k-th mixture cluster of a mode-m sub-tensor is determined by maximizing intra-cluster correlations and minimizing inter-cluster correlations at the same time. This interesting result is similar to the optimized orthogonal matching pursuit approach [148], where the k-th atom is resolved by simultaneously minimizing its linear dependence with previously selected atoms and maximizing the projected norm of the residual.

Moreover, it is obvious that Equation 6.5 can be computed in the reduced tensor space[1] to significantly decrease computational costs. The first term of the objective function in Equation 6.5 is the projection coefficients of $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ onto the basis matrices and the dual mode-m basis matrix of cluster c_{i_k}. It should be already computed when resolving the first mixture cluster of $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ and remains unchanged during the whole clustering stage. The second term instead can be interpreted as transforming the projected $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ in the sub-spaces of cluster c_{i_j}, for $j = 1, 2, \ldots, k-1$, to the sub-spaces of cluster c_{i_k}, followed by the multiplication with $\mathbf{V}_{m,c_{i_k}}^T$ to obtain projection coefficients. As a result, we can avoid computing the residual sub-tensor in the original tensor space (Equation 6.4), which needs to first reconstruct the corresponding mode-m sub-tensor from the results of previous iteration.

Optimal Projection

Since the sub-spaces of different clusters may be correlated, each time when assigning a new mixture cluster to a mode-m sub-tensor, its mixing coefficients for previously selected mixture clusters should be updated to account for the change in the cluster membership. This guarantees an optimal projection of the mode-m sub-tensor onto the sub-spaces of all selected mixture clusters. We therefore introduce the following theorem to update the mixing coefficients of a mode-m sub-tensor:

Theorem 6.3: *The mixing coefficients for the k selected mixture clusters $c_{i_1}, c_{i_2}, \ldots, c_{i_k}$ of the i-th mode-m sub-tensor $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ are given by*

$$\mathbf{u}_{m_i}^{(k)} = \mathbf{Z}_{m_i}^{(k)+} \mathbf{a}_{m_i}^{(k)}, \tag{6.6}$$

where

$$\mathbf{u}_{m_i}^{(k)} = \left[\left(\mathbf{U}_{m,c_{i_1}}\right)_{i*} \quad \cdots \quad \left(\mathbf{U}_{m,c_{i_k}}\right)_{i*} \right]^T, \tag{6.7}$$

[1]In this dissertation, the *reduced tensor space* is referred to as the union of all decomposed cluster sub-spaces, whose dimensionality is frequently much less than the original tensor space.

$$\mathbf{Z}^{(k)}_{m_i} = \begin{bmatrix} uf_m\Big(\boldsymbol{\mathcal{Z}}_{c_{i_1}} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_n \mathbf{U}^T_{n,c_{i_1}}\mathbf{U}_{n,c_{i_1}}\Big) uf_m\big(\boldsymbol{\mathcal{Z}}_{c_{i_1}}\big)^T & \cdots & uf_m\Big(\boldsymbol{\mathcal{Z}}_{c_{i_1}} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_n \mathbf{U}^T_{n,c_{i_k}}\mathbf{U}_{n,c_{i_1}}\Big) uf_m\big(\boldsymbol{\mathcal{Z}}_{c_{i_k}}\big)^T \\ \vdots & \ddots & \vdots \\ uf_m\Big(\boldsymbol{\mathcal{Z}}_{c_{i_k}} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_n \mathbf{U}^T_{n,c_{i_1}}\mathbf{U}_{n,c_{i_k}}\Big) uf_m\big(\boldsymbol{\mathcal{Z}}_{c_{i_1}}\big)^T & \cdots & uf_m\Big(\boldsymbol{\mathcal{Z}}_{c_{i_k}} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_n \mathbf{U}^T_{n,c_{i_k}}\mathbf{U}_{n,c_{i_k}}\Big) uf_m\big(\boldsymbol{\mathcal{Z}}_{c_{i_k}}\big)^T \end{bmatrix}, \tag{6.8}$$

$$\mathbf{a}^{(k)}_{m_i} = \begin{bmatrix} uf_m\Big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_n \mathbf{U}^T_{n,c_{i_1}}\Big) uf_m\big(\boldsymbol{\mathcal{Z}}_{c_{i_1}}\big)^T & \cdots & uf_m\Big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_n \mathbf{U}^T_{n,c_{i_k}}\Big) uf_m\big(\boldsymbol{\mathcal{Z}}_{c_{i_k}}\big)^T \end{bmatrix}^T . \tag{6.9}$$

Interested readers may refer to Section 6.4.2 for the mathematical proof of Theorem 6.3. Intriguingly, Equation 6.6 resembles the least-squares solution to the projection coefficients of an observation onto a set of basis vectors, where $\mathbf{Z}^{(k)}_{m_i}$ can be regarded as the Gram matrix that accounts for the correlations between all available cluster sub-spaces. Note that the proposed K-CTA algorithm is indeed efficient since K_m is usually a small positive integer. Moreover, all of the operations during the clustering stage are performed in the reduced tensor space, and most of them only need to be computed once at the beginning of this stage.

Post-Processing

To satisfy the orthonormal constraints on basis matrices in Equation 6.1, we additionally decompose the mode-m basis matrix of each cluster using principal component analysis to obtain

$$\mathbf{U}_{m,c} = \mathbf{U}'_{m,c}\mathbf{W}_c, \tag{6.10}$$

where $\mathbf{U}'_{m,c} \in \mathbb{R}^{I_m \times R_m}$ is a basis matrix whose columns are orthonormal principal components of $\mathbf{U}_{m,c}$, and each column of $\mathbf{W}_c \in \mathbb{R}^{R_m \times R_m}$ contains the projection coefficients of each column of $\mathbf{U}_{m,c}$, respectively. After that, we replace the mode-m basis matrix of cluster c with $\mathbf{U}'_{m,c}$ and re-compute the core tensor of cluster c by the mode-m product $\boldsymbol{\mathcal{Z}}_c \times_m \mathbf{W}_c$, so that the value of the objective function in Equation 6.1 is unchanged.

6.2.3 Update Stage

In this stage, the core tensor and basis matrices of each cluster are updated from the results of the clustering stage. While this problem for CTA can be easily solved by simultaneously applying tensor approximation to the member sub-tensors of each cluster, sub-space learning for all clusters at the same time would be difficult for K-CTA, since each mode-m sub-tensor now belongs to K_m different clusters that may be correlated with each other. We thus alternately decompose one cluster at a time by tensor approximation, while fixing the results of other clusters. To simplify the proposed algorithm, the cluster membership of each mode-m sub-tensor is not altered in this stage, leaving it to be updated only in the clustering stage.

At the c-th iteration, the core tensor $\boldsymbol{\mathcal{Z}}_c$ and basis matrices $\{\mathbf{U}_{n,c}\}_{n=1}^{N}$ of cluster c are allowed to change, while those of other clusters are fixed. Equation 6.1 thus can be rewritten as the following constrained least-squares optimization problem:

$$\min_{\{\boldsymbol{\mathcal{Z}}_c, \{\mathbf{U}_{n,c}\}_{n=1}^{N}\}} \frac{1}{2}\left\|\boldsymbol{\mathcal{R}}_c - \boldsymbol{\mathcal{Z}}_c \bigtimes_{n=1}^{N}{}_n \mathbf{U}_{n,c}\right\|_F^2,$$

$$\text{subject to} \begin{cases} \forall i,\ \sum_{c=1}^{C} \left\|\left(\mathbf{U}_{m,c}\right)_{i*}\right\|_0 = K_m R_m & \text{(sparse constraints)}, \\ \forall i,\ \left\|\left(\mathbf{U}_{m,c}\right)_{i*}\right\|_0 \in \{0, R_m\} & \text{(membership constraints)}, \\ \forall n,\ \mathbf{U}_{n,c}^T \mathbf{U}_{n,c} = \mathbf{I}_{R_n} & \text{(orthonormal constraints)}, \end{cases} \tag{6.11}$$

where $\boldsymbol{\mathcal{R}}_c$ is the residual tensor at the c-th iteration defined as

$$\boldsymbol{\mathcal{R}}_c = \boldsymbol{\mathcal{A}} - \sum_{\substack{j=1 \\ j\neq c}}^{C} \left(\boldsymbol{\mathcal{Z}}_j \bigtimes_{n=1}^{N}{}_n \mathbf{U}_{n,j}\right). \tag{6.12}$$

By comparing the objective function in Equation 6.11 to Equations 4.1 and 4.2 in [33], we know that Equation 6.11 can be solved with the aid of tensor decomposition. To enforce the sparse constraint on $\mathbf{U}_{m,c}$, the key idea is to only include member sub-tensors of cluster c in the tensor decomposition process and just update the non-zero entries of $\mathbf{U}_{m,c}$, which is based on the same concept in the update stage of CTA (Section 5.3.3) and the codebook update stage of K-SVD [3, 4]. As a result, the membership of cluster c is fixed, and the zero entries of $\mathbf{U}_{m,c}$ remain zeros. However, the non-zero entries of $\mathbf{U}_{m,c}$ are allowed to change with the decomposed results of cluster c at the same time.

6.3 Implementation Issues

In this section, we discuss some practical issues of K-CTA, such as the initial guess of the core tensor and basis matrices of each cluster (Section 6.3.1) and the degeneracy and convergence problems of K-CTA (Section 6.3.2). We omit a detailed description of the extraction of global basis matrices, since they can be easily derived based on the same technique described in Section 5.4.2.

6.3.1 Initial Guess

Similar to CTA, one practical issue of K-CTA is the initial guess of the core tensor and basis matrices of each cluster. A simple and heuristic solution is to employ the decomposed results of hard clustering. We can execute static CTA to obtain the initial core tensor and basis matrices of each cluster for further data analysis using K-CTA. Various general soft clustering techniques for determining the initial membership of each mode-m sub-tensor, such as mixture models [6, 37, 174] and sparse representation [3, 4, 177], also provide desirable solutions.

Interestingly, while finding a favorable initial guess is a significant issue for CTA and other iterative algorithms, K-CTA is less sensitive to the quality of an initial solution. As the number of mixture clusters increases, the importance of an appropriate initial guess for K-CTA decreases. This result is not surprising since the impact of inappropriate initial cluster membership can be compensated by additional mixture clusters. The compensation also can be regarded as an optimal interpolation scheme from other cluster sub-spaces in the least-squares sense, which particularly allows smooth transitions across different physical conditions in practical applications. In Section 9.2, we will further demonstrate the influence of this characteristic of K-CTA in the experimental results of bi-scale radiance transfer.

6.3.2 Degeneracy and Convergence

When the total number of clusters is large, we have observed that sometimes all the entries in a column of a mode-m basis matrix of a cluster, say $\mathbf{U}_{m,c}$, may become zeros at the end of the clustering stage, which implies that no mode-m sub-tensors are classified into cluster c. Although this degeneracy problem does not occur frequently in practice, an empty cluster actually consumes memory space without any contributions to final approximation errors[2]. In the worst case, K-CTA may even derive a poor solution when there are too many empty clusters, as if the total number of clusters were set to a lower value. Our solution to this issue is to split the cluster with the largest total sum of approximation errors in half by sorting the approximation error of each mode-m sub-tensor within this cluster. If there are more than one empty cluster, we can sequentially split non-empty clusters using the above method until each empty cluster has been assigned at least one mode-m sub-tensor.

Another important question is whether the proposed iterative K-CTA algorithm always converges to a local optimum. Apparently, the answer is no. Similar to K-SVD [3, 4], the convergence of iterative K-CTA is not guaranteed, since only the approximate mixing coefficients (and also the cluster membership) of each mode-m sub-tensor are derived in the clustering stage. Therefore, the total sum of approximation errors is not guaranteed to decrease when compared to the decomposed results of previous iteration. Fortunately, although iterative K-CTA theoretically does not ensure convergence, we have found that it practically converges within just a few iterations in the experiments. We currently have no idea about how to efficiently update mixing coefficients and cluster sub-spaces, so that approximation errors are always reduced. However,

[2]When encountering the degeneracy problem, we can save sparse mode-m basis matrices in the compressed column storage format [5] without memory space overhead. Nevertheless, it is better to employ the compressed row storage format [5], so that the mixing coefficients of a mode-m sub-tensor are grouped together to reduce the cache miss rates of CPUs (or GPUs) for efficient run-time reconstruction.

a simple and intuitive technique can be applied to prevent bad results due to the divergence problem. At the end of the update stage, if the total sum of approximation errors increases, we instead restore the decomposed results of previous iteration. As a result, the approximation errors of an input tensor will never increase, and the convergence criterion of K-CTA in Algorithm 6.1 can be always reached.

6.4 Mathematical Proofs

In the following two sections, we present the mathematical proofs of the proposed greedy search (Section 6.4.1) and optimal projection (Section 6.4.2) methods in the clustering stage of K-CTA, including Corollary 6.1 for the first mixture cluster as well as Theorems 6.2 and 6.3 for the remaining mixture clusters.

6.4.1 Greedy Search

In the clustering stage of K-CTA, each mode-m sub-tensor of $\boldsymbol{\mathcal{A}}$ is sequentially classified into K_m mixture clusters using a greedy approach, while the core tensor and basis matrices, except for the mode-m basis matrix, of each cluster are fixed. Since the cluster membership and mixing coefficients of each mode-m sub-tensor are independent, Equation 6.1, without the orthonormal constraints on $\mathbf{U}_{m,1}, \mathbf{U}_{m,2}, \ldots, \mathbf{U}_{m,C}$, can be separated into I_m distinct constrained least-squares optimization sub-problems as

$$\min_{\left\{(\mathbf{U}_{m,c})_{i*}\right\}_{c=1}^{C}} \frac{1}{2}\left\| \boldsymbol{\mathcal{A}}_{\langle m_i \rangle} - \sum_{c=1}^{C} \left(\boldsymbol{\mathcal{Z}}_c \times_m \left(\mathbf{U}_{m,c}\right)_{i*} \bigtimes_{\substack{n=1 \\ n\neq m}}^{N}{}_{n} \mathbf{U}_{n,c} \right) \right\|_F^2,$$
$$\text{subject to } \begin{cases} \sum_{c=1}^{C} \left\| \left(\mathbf{U}_{m,c}\right)_{i*} \right\|_0 = K_m R_m & \text{(sparse constraint)}, \\ \forall c, \; \left\| \left(\mathbf{U}_{m,c}\right)_{i*} \right\|_0 \in \{0, R_m\} & \text{(membership constraints)}, \end{cases} \tag{6.13}$$

for $i = 1, 2, \ldots, I_m$. Note that the orthonormal constraints on $\mathbf{U}_{m,1}, \mathbf{U}_{m,2}, \ldots, \mathbf{U}_{m,C}$ in Equation 6.1 can be enforced from the optimized results of Equation 6.13 using a post-processing approach as described in Section 6.2.2.

First Mixture Cluster

For the first mixture cluster and corresponding mixing coefficients of a mode-m sub-tensor, we propose to obtain them from Theorem 5.3 and Corollary 6.1. By following the same approach as in the proof of Theorem 5.3, it is quite easy to verify the correctness of Corollary 6.1.

Proof of Corollary 6.1: The first mixture cluster c_{i_1} of the i-th mode-m sub-tensor $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ is selected as if $K_m = 1$. Equation 6.13 thus can be further simplified into Equation 5.13. From

Equation 5.18, the least-squares solution to Equation 5.13 is the projection coefficients of $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ onto the sub-spaces of cluster c_{i_1} so that the approximation error of $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ is minimized. Equation 6.3 is thus proved. $\square$

Remaining Mixture Clusters

After selecting the first mixture cluster of a mode-m sub-tensor, its remaining mixture clusters are then sequentially derived from its previously determined mixture clusters and mixing coefficients by applying Theorem 6.2. The mathematical proof of Theorem 6.2 is described in the following paragraph.

Proof of Theorem 6.2: For the k-th mixture cluster c_{i_k} of the i-th mode-m sub-tensor $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ other than c_{i_1}, it is determined as if $K_m = k$. When previously selected mixture clusters $c_{i_1}, c_{i_2}, \ldots, c_{i_{k-1}}$ and the corresponding mixing coefficients are fixed, the objective function in Equation 6.13 becomes

$$\frac{1}{2}\left\|\boldsymbol{\mathcal{R}}^{(k)}_{\langle m_i \rangle} - \boldsymbol{\mathcal{Z}}_{c_{i_k}} \times_m \left(\mathbf{U}_{m,c_{i_k}}\right)_{i*} \bigtimes_{\substack{n=1 \\ n\neq m}}^{N}{}_n \mathbf{U}_{n,c_{i_k}}\right\|_F^2, \tag{6.14}$$

where $\boldsymbol{\mathcal{R}}^{(k)}_{\langle m_i \rangle}$ is defined as Equation 6.4. By following the same approach as in the proofs of Theorem 5.3 and Corollary 6.1, the minimization of Equation 6.14 is equivalent to the maximization of

$$\frac{1}{2}\left\| uf_m\!\left(\boldsymbol{\mathcal{R}}^{(k)}_{\langle m_i \rangle} \bigtimes_{\substack{n=1 \\ n\neq m}}^{N}{}_n \mathbf{U}^T_{n,c_{i_k}}\right)\mathbf{V}^T_{m,c_{i_k}}\right\|_F^2. \tag{6.15}$$

Substituting Equation 6.4 for $\boldsymbol{\mathcal{R}}^{(k)}_{\langle m_i \rangle}$ in Equation 6.15 then yields the objective function in Equation 6.5. Theorem 6.2 thus is proved by identifying that the cluster membership is implicitly specified in the solution to Equation 6.13. $\square$

6.4.2 Optimal Projection

In the clustering stage of K-CTA, each time when adding a new mixture cluster to a mode-m sub-tensor, its mixing coefficients for previously determined mixture clusters should be updated by applying the proposed optimal projection algorithm (Theorem 6.3). This actually guarantees an optimal solution of mixing coefficients with respect to all selected mixture clusters, since the sub-spaces of different clusters may have strong correlations. For completeness, we present the detailed mathematical proof of Theorem 6.3 as follows.

Proof of Theorem 6.3: Suppose that total k mixture clusters $c_{i_1}, c_{i_2}, \ldots, c_{i_k}$ of the i-th mode-m sub-tensor $\boldsymbol{\mathcal{A}}_{\langle m_i \rangle}$ have been selected as if $K_m = k$. Similar to Equations 5.14 and 5.15, since the constraints in Equation 6.13 can be satisfied by setting $\left(\mathbf{U}_{m,c}\right)_{i*}$ to zeros for all $c \notin \{c_{i_1}, c_{i_2}, \ldots, c_{i_k}\}$, Equation 6.13 can be simplified into a standard unconstrained least-squares problem whose objective function is

$$\frac{1}{2}\left\| uf_m\!\left(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\right) - \sum_{j=1}^{k}\left(\left(\mathbf{U}_{m,c_{i_j}}\right)_{i*} uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_{i_j}} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_{n}\, \mathbf{U}_{n,c_{i_j}}\right)\right)\right\|_2^2 . \tag{6.16}$$

By taking the first-order partial derivatives of Equation 6.16 with respect to each entry of $\left(\mathbf{U}_{m,c_{i_1}}\right)_{i*}$ and setting the resulting derivatives to zeros, we have the following linear equation:

$$\begin{aligned}\sum_{j=1}^{k}\left(uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_{i_1}} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_{n}\, \mathbf{U}_{n,c_{i_1}}\right) uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_{i_j}} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_{n}\, \mathbf{U}_{n,c_{i_j}}\right)^{T} \left(\mathbf{U}_{m,c_{i_j}}\right)_{i*}^{T}\right)& \\ = uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_{i_1}} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_{n}\, \mathbf{U}_{n,c_{i_1}}\right) uf_m\!\left(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\right)^{T}&.\end{aligned} \tag{6.17}$$

The right side of Equation 6.17 can be rewritten as

$$\begin{aligned} uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_{i_1}} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_{n}\, \mathbf{U}_{n,c_{i_1}}\right) uf_m\!\left(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\right)^{T} &= uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_{i_1}}\right)\left(\bigotimes_{\substack{n=1\\ n\neq m}}^{N} \mathbf{U}_{n,c_{i_1}}^{T}\right) uf_m\!\left(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\right)^{T} \\ &= uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_{i_1}}\right)\left(uf_m\!\left(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\right)\left(\bigotimes_{\substack{n=1\\ n\neq m}}^{N} \mathbf{U}_{n,c_{i_1}}\right)\right)^{T} \\ &= uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_{i_1}}\right) uf_m\!\left(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_{n}\, \mathbf{U}_{n,c_{i_1}}^{T}\right)^{T}. \end{aligned} \tag{6.18}$$

Similarly, the left side of Equation 6.17 is equal to

$$\begin{aligned} &\sum_{j=1}^{k}\left(uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_{i_1}} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_{n}\, \mathbf{U}_{n,c_{i_1}}\right) uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_{i_j}} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_{n}\, \mathbf{U}_{n,c_{i_j}}\right)^{T} \left(\mathbf{U}_{m,c_{i_j}}\right)_{i*}^{T}\right) \\ &\quad= \sum_{j=1}^{k}\left(uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_{i_1}} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_{n}\, \mathbf{U}_{n,c_{i_1}}\right)\left(\bigotimes_{\substack{n=1\\ n\neq m}}^{N} \mathbf{U}_{n,c_{i_j}}\right) uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_{i_j}}\right)^{T} \left(\mathbf{U}_{m,c_{i_j}}\right)_{i*}^{T}\right) \\ &\quad= \sum_{j=1}^{k}\left(uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_{i_1}} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_{n}\, \mathbf{U}_{n,c_{i_1}} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_{n}\, \mathbf{U}_{n,c_{i_j}}^{T}\right) uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_{i_j}}\right)^{T} \left(\mathbf{U}_{m,c_{i_j}}\right)_{i*}^{T}\right) \\ &\quad= \sum_{j=1}^{k}\left(uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_{i_1}} \mathop{\times}\limits_{\substack{n=1\\ n\neq m}}^{N}{}_{n}\, \mathbf{U}_{n,c_{i_j}}^{T}\mathbf{U}_{n,c_{i_1}}\right) uf_m\!\left(\boldsymbol{\mathcal{Z}}_{c_{i_j}}\right)^{T} \left(\mathbf{U}_{m,c_{i_j}}\right)_{i*}^{T}\right). \end{aligned} \tag{6.19}$$

By following Equations 6.17–6.19 with respect to each entry of $\left(\mathbf{U}_{m,c_{i_j}}\right)_{i*}$ for all j, we have the following set of linear equations:

$$\begin{aligned}
&\sum_{j=1}^{k}\left(uf_m\Big(\boldsymbol{\mathcal{Z}}_{c_{i_1}}\bigtimes_{\substack{n=1\\n\neq m}}^{N}{}_n\,\mathbf{U}_{n,c_{i_j}}^T\mathbf{U}_{n,c_{i_1}}\Big)uf_m\big(\boldsymbol{\mathcal{Z}}_{c_{i_j}}\big)^T\big(\mathbf{U}_{m,c_{i_j}}\big)_{i*}^T\right)\\
&\qquad\qquad\qquad = uf_m\big(\boldsymbol{\mathcal{Z}}_{c_{i_1}}\big)\,uf_m\Big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\bigtimes_{\substack{n=1\\n\neq m}}^{N}{}_n\,\mathbf{U}_{n,c_{i_1}}^T\Big)^T,\\
&\qquad\qquad\qquad\vdots \\
&\sum_{j=1}^{k}\left(uf_m\Big(\boldsymbol{\mathcal{Z}}_{c_{i_k}}\bigtimes_{\substack{n=1\\n\neq m}}^{N}{}_n\,\mathbf{U}_{n,c_{i_j}}^T\mathbf{U}_{n,c_{i_k}}\Big)uf_m\big(\boldsymbol{\mathcal{Z}}_{c_{i_j}}\big)^T\big(\mathbf{U}_{m,c_{i_j}}\big)_{i*}^T\right)\\
&\qquad\qquad\qquad = uf_m\big(\boldsymbol{\mathcal{Z}}_{c_{i_k}}\big)\,uf_m\Big(\boldsymbol{\mathcal{A}}_{\langle m_i\rangle}\bigtimes_{\substack{n=1\\n\neq m}}^{N}{}_n\,\mathbf{U}_{n,c_{i_k}}^T\Big)^T,
\end{aligned}\tag{6.20}$$

which can be further written in a matrix form as $\mathbf{Z}_{m_i}^{(k)}\mathbf{u}_{m_i}^{(k)} = \mathbf{a}_{m_i}^{(k)}$, where $\mathbf{u}_{m_i}^{(k)}$, $\mathbf{Z}_{m_i}^{(k)}$, and $\mathbf{a}_{m_i}^{(k)}$ are respectively defined in Equations 6.7–6.9. As a result, we can conclude that Equation 6.6 gives the least-squares solution of $\big(\mathbf{U}_{m,c_{i_1}}\big)_{i*}, \big(\mathbf{U}_{m,c_{i_2}}\big)_{i*}, \ldots, \big(\mathbf{U}_{m,c_{i_k}}\big)_{i*}$ to Equation 6.16. Theorem 6.3 thus is proved. $\square$

Chapter 7

Application I: Illumination and Reflectance Functions

This chapter presents the applications of the proposed *spherical radial basis function* (SRBF) representations to important illumination and reflectance functions in computer graphics and vision, such as high dynamic range environment maps (Section 7.1) and bidirectional reflectance distribution functions (Section 7.2). In the third part of this chapter, we also describe an importance sampling scheme for direct illumination estimation in ray tracing algorithms, where the materials of object surfaces are modeled with the scattered univariate SRBF representation (Section 7.3).

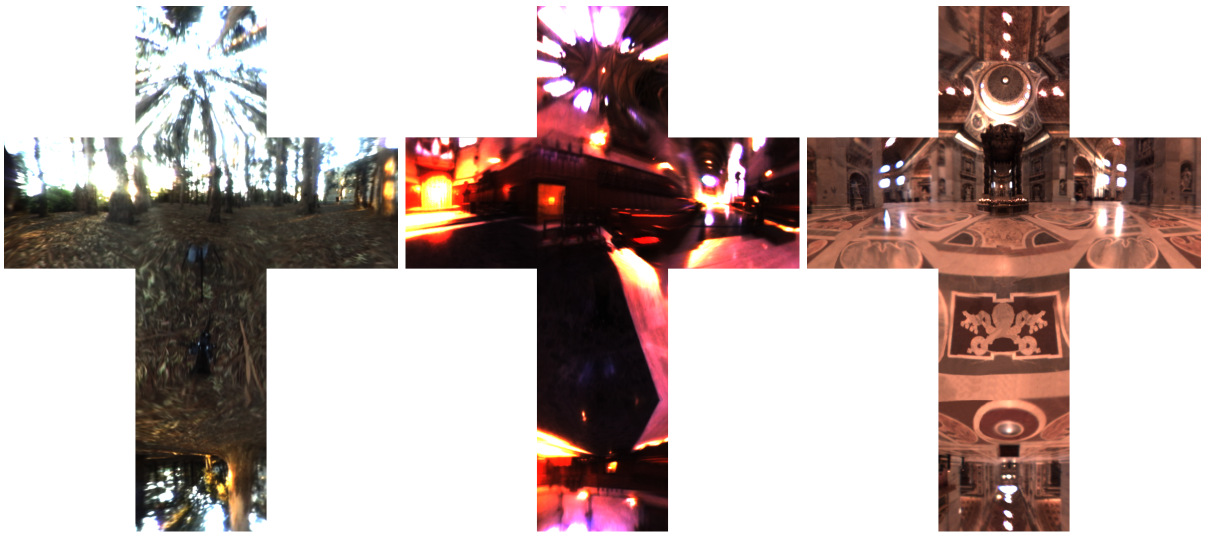
7.1 High Dynamic Range Environment Maps

7.1.1 Problem Formulation

The dynamic range of intensities between light and dark areas in a real-world scene is frequently much greater than traditional digital imaging techniques and display devices. Therefore, the *high dynamic range* (HDR) environment map [34, 36] particularly aims at capturing the incident radiance of object surfaces from all illumination directions in a natural scene with a wide range of intensity levels (Figure 7.1). When adopting a HDR environment map as the light source in a real-time rendering application, the visual fidelity of rendered images can be greatly enhanced to simulate the surface irradiance under realistic illumination.

It is obvious that a HDR environment map is a real-valued univariate spherical function $L(\omega_l) \in \mathbb{R}$ that depends on the illumination direction ω_l on the unit sphere $\mathbb{S}^2$. To efficiently shade objects using a HDR environment map, we can therefore model the lighting environment with the scattered univariate SRBF representation as described in Section 3.4. The entire process is formulated as an optimization problem, and handled by minimizing squared approximation errors.

Given a desired number of univariate SRBFs J for a HDR environment map $L(\omega_l)$, we



(a) *Eucalyptus Grove* (b) *Grace Cathedral* (c) *St. Peter's Basilica*

Figure 7.1: Examples of HDR environment maps from the light probe image gallery [35]. Each image shows the spectral radiance (red, green, and blue light) from different illumination directions.

intend to learn three sets of SRBF parameters: the basis coefficient set $\left\{\beta_j^{(L)} \in \mathbb{R}\right\}_{j=1}^{J}$, the center set $\left\{\xi_j^{(L)} \in \mathbb{S}^2\right\}_{j=1}^{J}$, and the bandwidth set $\left\{\lambda_j^{(L)} \in \mathbb{R}\right\}_{j=1}^{J}$, such that the following unconstrained least-squares optimization problem is solved:

$$\min_{\left\{\beta_j^{(L)},\xi_j^{(L)},\lambda_j^{(L)}\right\}_{j=1}^{J}} \frac{1}{2}\int_{\mathbb{S}^2}\Big(L(\omega_l)-\hat{L}(\omega_l)\Big)^2 d\omega_l, \tag{7.1}$$

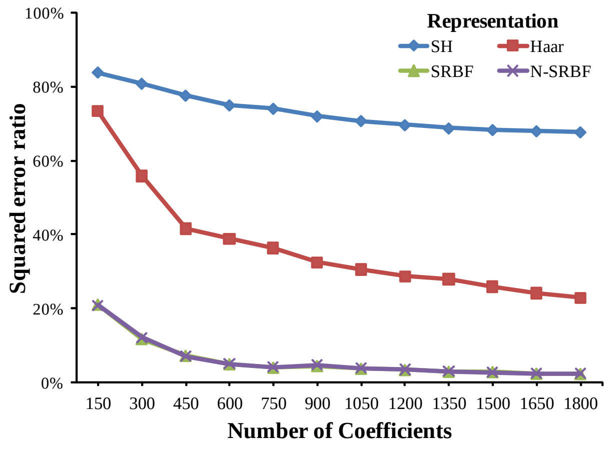
where

$$\hat{L}(\omega_l)=\sum_{j=1}^{J}\beta_j^{(L)}G\big(\omega_l\cdot\xi_j^{(L)}|\lambda_j^{(L)}\big). \tag{7.2}$$

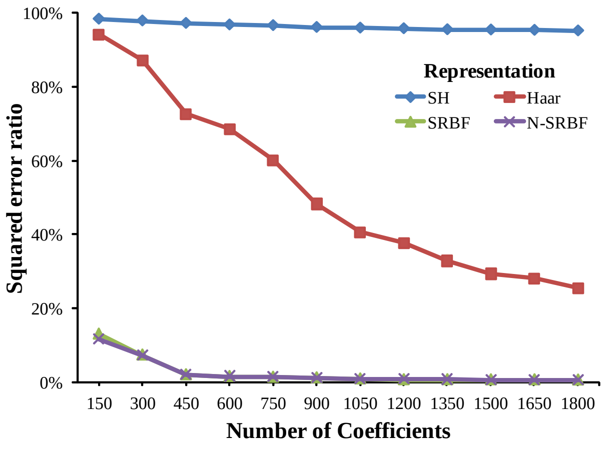
It is then straightforward to derive the SRBF parameters in Equation 7.1 by applying Algorithm 3.1.

7.1.2 Experimental Results

The experiments and simulation timings of HDR environment map approximation were conducted and measured on a workstation with an Intel Core 2 Extreme QX9650 CPU, an NVIDIA GeForce 9800 GX2 graphics card, and 8 gigabytes main memory under Microsoft Windows Vista operating system. Parts of the fitting process, including the initial guess of SRBF parameters and gradient computation, were accelerated on GPUs using NVIDIA *Compute Unified Device Architecture* (CUDA) [128]. The univariate Gaussian SRBFs were adopted to represent the HDR environment maps in our experiments. In general, we have found no significant dif-



(a) *Grace Cathedral*

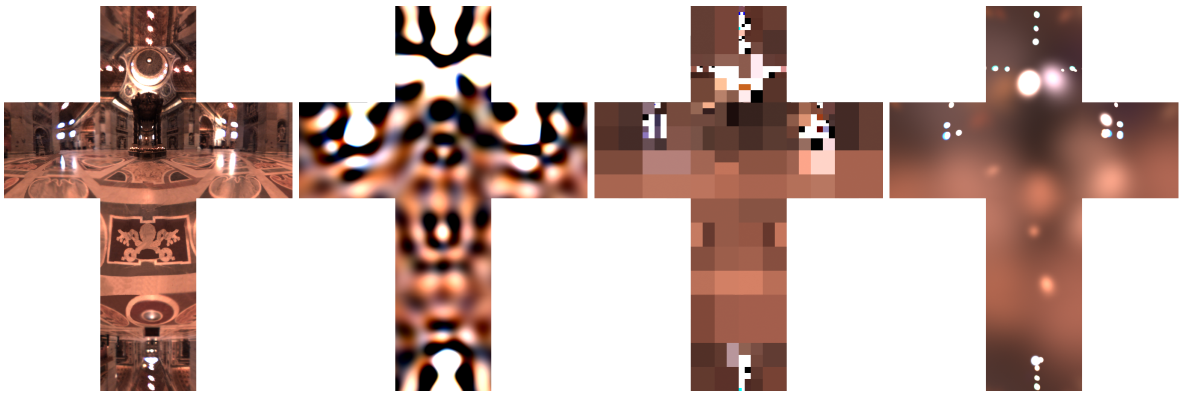


(b) *St. Peter's Basilica*

Figure 7.2: Plots of the squared error ratio versus the number of coefficients/parameters based on different functional representations for HDR environment maps, including *spherical harmonics* (SH); *area-weighted Haar wavelets* (Haar) [125]; *scattered univariate SRBFs* (SRBF); *scattered univariate SRBFs with non-negative basis coefficients* (N-SRBF).

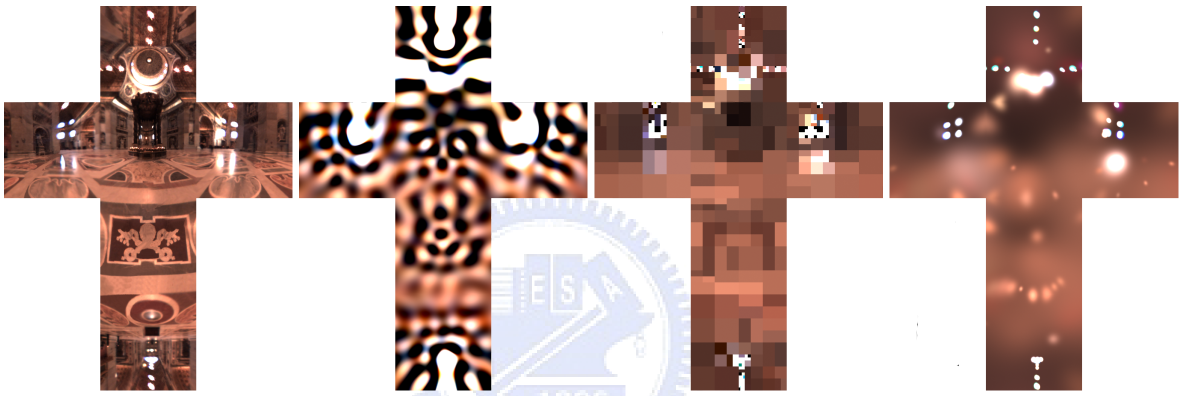
ference between various types of univariate SRBFs for approximating HDR environment maps, but univariate Gaussian SRBFs are more locally compact than univariate Abel-Poisson SRBFs and handle most common cases very well. The SRBF center and bandwidth sets were constrained to be the same for red, green, and blue channels of a HDR environment map, since separately fitting the data of each channel only slightly reduces approximation errors, but increases computational costs and storage space by a factor of 2 or more.

Figure 7.2 plots the squared error ratio versus the number of coefficients/parameters for different functional representations, including spherical harmonics, area-weighted Haar wavelets [125], and the proposed scattered univariate SRBF representation (with/without non-negative basis coefficient constraints). Figures 7.3 and 7.4 further demonstrate the reconstructed results of various HDR environment maps. Note that we compare the functional representations based on the same total number of coefficients/parameters, namely the same storage space. For spherical harmonics and area-weighted Haar wavelets, the number of coefficients is equivalent to the number of basis functions, whereas the number of parameters for the scattered univariate SRBF representation is six times the number of basis functions. Typically, fitting a $6\times256\times256$ HDR environment map with 300 univariate SRBFs (total 1800 parameters) takes about 25 minutes on average. Figures 7.2–7.4 obviously show that the proposed approach achieves the lowest squared approximation error among the three functional representations and captures most features of the HDR environment map with only a few univariate SRBFs. By contrast, the results of the area-weighted Haar wavelets are blocky and fail to distinguish some features, especially the high-energy lights at the basilica ceiling. For spherical harmonics, since high-energy signals dominate over low-energy ones, there are a lot of unusual ringing artifacts in the reconstructed results. Although a simple windowing technique can reduce the ringing effects, they can not be totally eliminated owing to the wide range of intensity levels in the HDR environment maps,



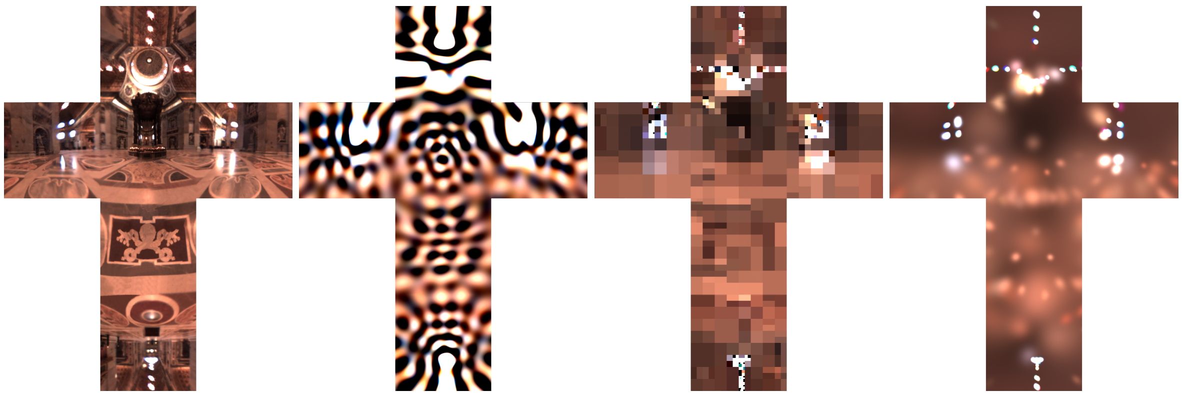
(96.77%) (68.49%) (1.63%)

(a) Number of coefficeints/parameters: 600



(95.72%) (37.59%) (0.87%)

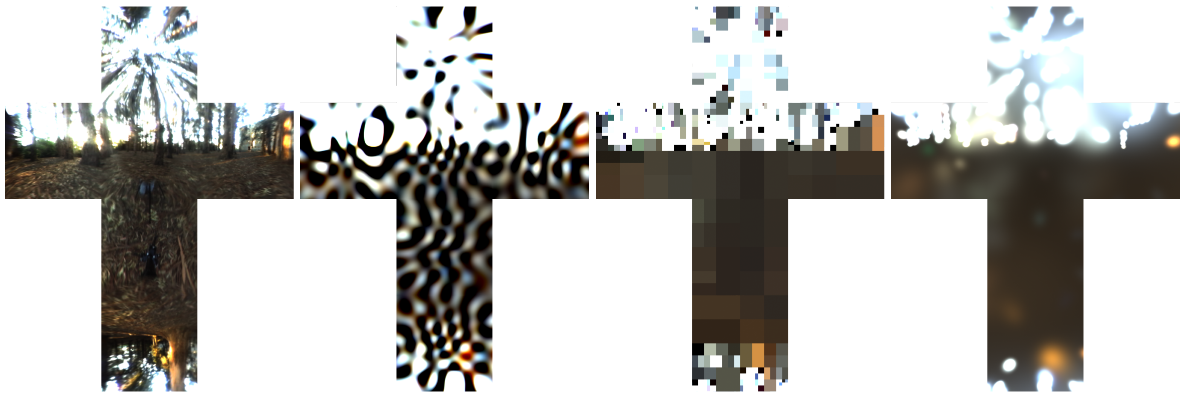
(b) Number of coefficeints/parameters: 1200



(95.1%) (25.48%) (0.71%)

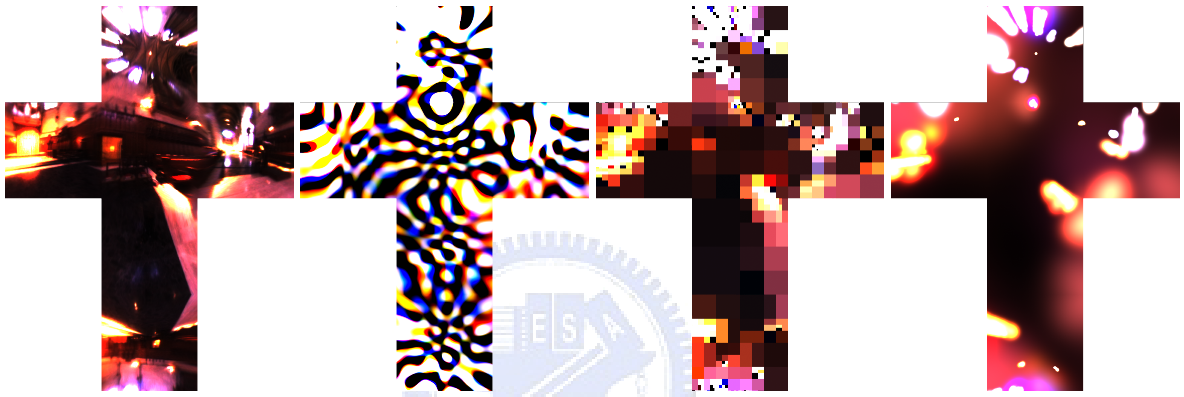
(c) Number of coefficeints/parameters: 1800

Figure 7.3: Reconstructed images of the HDR environment map *St. Peter's Basilica* based on different functional representations. From left to right in each row: raw data; spherical harmonics; area-weighted Haar wavelets [125]; scattered univariate SRBFs with non-negative basis coefficients. The squared error ratio of each reconstructed image is shown in parentheses.



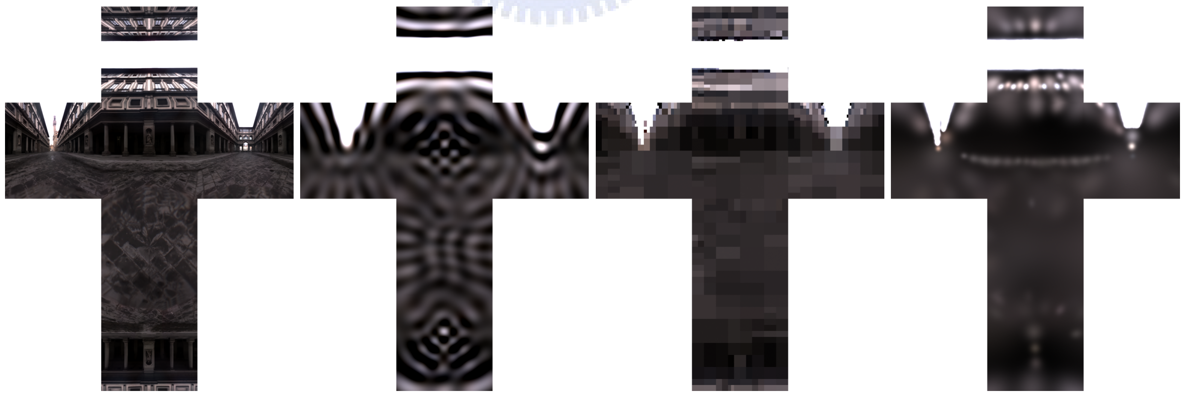
(41.88%) (25.25%) (5.63%)

(a) *Eucalyptus Grove*



(67.59%) (22.92%) (2.39%)

(b) *Grace Cathedral*



(6.28%) (3.68%) (1.26%)

(c) *The Uffizi Gallery*

Figure 7.4: Reconstructed images of the HDR environment maps *Eucalyptus Grove*, *Grace Cathedral*, and *The Uffizi Gallery* based on different functional representations. From left to right in each row: raw data; spherical harmonics; area-weighted Haar wavelets [125]; scattered univariate SRBFs with non-negative basis coefficients. The number of coefficeints/parameters for each functional representation is 1800, and the squared error ratio of each reconstructed image is shown in parentheses.

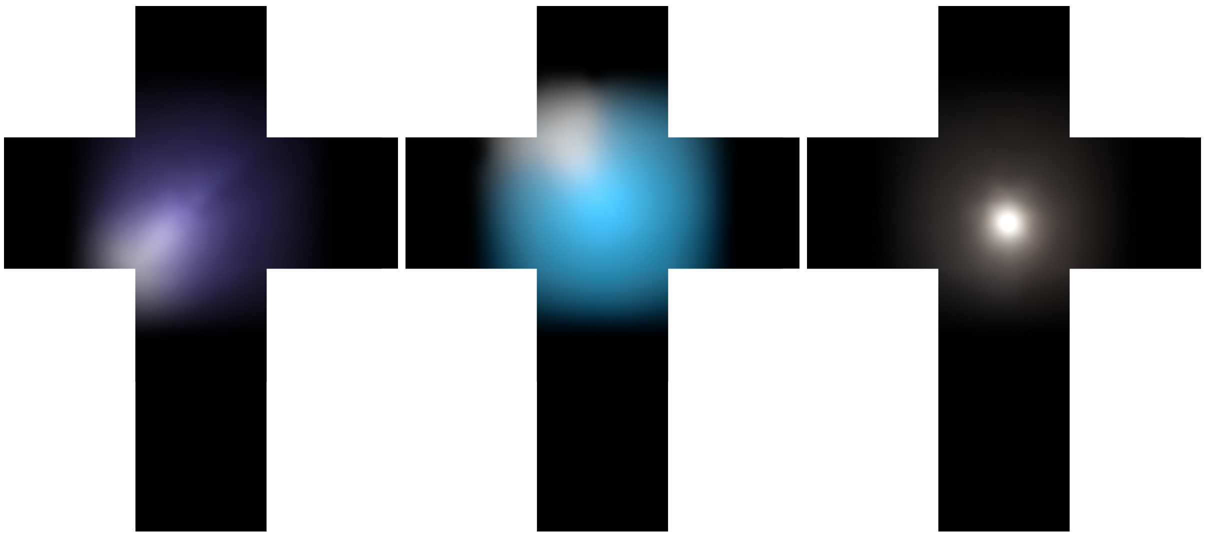

(a) *Blue Metallic Paint* (b) *Krylon Blue* (c) *Nickel*

Figure 7.5: Examples of BRDFs from the databases of Cornell University [140] and Mitsubishi Electric Research Laboratories [120]. Each image shows the spectral reflectance (red, green, and blue light) in a single view direction from different illumination directions.

which may exceed six orders of magnitude.

Moreover, it is recommended to constrain the basis coefficients as non-negative, which can be achieved by simply setting the lower bounds of basis coefficients to zeros for the L-BFGS-B optimization solver [16, 212]. According to our experiments, non-negative basis coefficients only result in a slight increase in approximation errors (Figure 7.2), but tend to provide much smoother transitions than coefficients without non-negative constraints when rotating the HDR environment map.

7.2 Bidirectional Reflectance Distribution Functions

The surface reflectance properties significantly affect the appearance of real-world objects. Computer graphics and vision researchers are particularly interested in representing the interaction between materials and light in an efficient way. One of the most popular models is the *bidirectional reflectance distribution function* (BRDF). BRDFs assume that light hits a surface point from an illumination direction ω_l, and leaves the surface from the same point in a view direction ω_v on the hemisphere to which ω_l belongs. As a result, a BRDF is a real-valued bivariate spherical function $\rho(\omega_l, \omega_v) \in \mathbb{R}$ that gives the reflectance of a surface point with respect to the illumination and view directions. In some literatures, BRDFs are instead defined over the entire unit sphere $\mathbb{S}^2$. This further combines the reflective and transmissive parts of light transport together into one material model for transparent surfaces. Figure 7.5 shows some examples of BRDFs.

In this dissertation, we consider BRDFs measured from real-world opaque object surfaces, and introduce two compact functional models for BRDFs based on univariate and multivariate SRBFs. The first model relies on discretizing the illumination (or view) domain of a BRDF, so that the reflectance distribution of each illumination (or view) direction can be separately described using the scattered univariate SRBF representation (Section 7.2.1). In addition, the second model instead approximates a BRDF in its intrinsic domain to derive a continuous functional model based on the parameterized multivariate SRBF representation (Section 7.2.2).

7.2.1 Univariate SRBFs

Problem Formulation

Recall that the proposed scattered univariate SRBF representation can only handle a univariate spherical function, but the BRDF is intrinsically a bi-variate spherical function of illumination and view directions. This implies that the scattered univariate SRBF representation can not be directly applied to model a BRDF. Fortunately, a measured BRDF is one set of discrete reflectance observations in the most common case. The key idea is that if we collect all the BRDF data of a single view direction, the resulting data set will be the observations of a univariate spherical function, which describes the reflectance values in this view direction from different illumination directions.

More formally, let $\Omega_v = \{\omega_{v,i} \in \mathbb{S}^2\}_{i=1}^{I_{\Omega_v}}$ denote the set of sampled view directions for a measured BRDF $\rho(\omega_l, \omega_v)$, where I_{Ω_v} is the total number of sampled view directions. Then, all the BRDF data of the i-th view direction $\omega_{v,i}$ form a set of observations for the univariate spherical function $\rho_i(\omega_l) = \rho(\omega_l, \omega_{v,i})$, which can be roughly interpreted as the function of exitant radiance received in $\omega_{v,i}$ from different illumination directions. Similar to the HDR environment maps, each resulting univariate spherical function is separately approximated with a linear combination of J univariate SRBFs as

$$\forall i,\ \rho_i(\omega_l) \approx \hat{\rho}_i(\omega_l) = \sum_{j=1}^{J} \beta_j^{(\rho_i)} G\big(\omega_l \cdot \xi_j^{(\rho_i)} | \lambda_j^{(\rho_i)}\big), \tag{7.3}$$

where $\{\beta_j^{(\rho_i)} \in \mathbb{R}\}_{j=1}^{J}$, $\{\xi_j^{(\rho_i)} \in \mathbb{S}^2\}_{j=1}^{J}$, and $\{\lambda_j^{(\rho_i)} \in \mathbb{R}\}_{j=1}^{J}$ are respectively the basis coefficient set, the center set, and the bandwidth set of i-th illumination-dependent reflectance function $\rho_i(\omega_l)$. The SRBF parameters in Equation 7.3 thus can be obtained by solving the following set of total I_{Ω_v} distinct unconstrained least-squares optimization problems:

$$\forall i, \quad \min_{\left\{\beta_j^{(\rho_i)}, \xi_j^{(\rho_i)}, \lambda_j^{(\rho_i)}\right\}_{j=1}^{J}} \frac{1}{2} \int_{\mathbb{S}^2} \Big(\rho_i(\omega_l) - \hat{\rho}_i(\omega_l)\Big)^2 d\omega_l. \tag{7.4}$$

Note that one may instead discretize the illumination domain of a BRDF and then respectively optimize the resulting view-dependent spherical function of each sampled illumination direction. The choice of discretizing which domain strongly depends on the target application.

We will further discuss this subject in Section 7.3 and present an application of the scattered univariate SRBF representation for BRDFs based on discretized view domain in ray tracing algorithms.

Run-Time Rendering

Based on the scattered univariate SRBF representation, the shading color of a pixel is obtained from the SRBF parameters of discretized reflectance functions $\rho_1(\omega_l), \rho_1(\omega_2), \ldots, \rho_{I_{\Omega_v}}(\omega_l)$ at run-time. All of the SRBF parameters are stored in one or more two-dimensional textures if necessary. To achieve smooth view transitions, we apply linear interpolation on the reconstructed results of the K nearest neighbors of a view direction. The rendering process thus consists of the following steps:

1. Search for the K nearest neighbors of current novel view direction from Ω_v.
2. For each neighboring direction, sample the texture(s) for all the corresponding SRBF parameters and then reconstruct the reflectance value of current novel illumination direction as Equation 7.3.
3. The final shading color is then given by linearly interpolating the reconstructed reflectance value of each neighboring direction.

To efficiently perform Step 1 on GPUs, we construct a two-dimensional index texture in the parabolic parameterization [62] (or a cubic index texture) to record the indices of the K nearest neighbors for the associated direction of each texel in this texture. In current implementation, we only consider a single texel in the index texture, whose associated direction is nearest to current novel view direction. Note that this approach requires a high-resolution index texture (typically 256×256 in the parabolic parameterization or $6\times128\times128$ for the cubic format) to avoid visible artifacts and will increase the amount of data for rendering by a substantial factor. For computing the interpolation weights, the coordinates of each element in Ω_v also need to be stored in a texture or transferred to buffers on GPUs.

7.2.2 Multivariate SRBFs

Problem Formulation

Although the scattered univariate SRBF representation can accurately approximate BRDFs, the run-time rendering process still needs carefully designed interpolation techniques for smooth transitions across novel illumination/view directions. It is also not compact and frequently provides limited compression ratios when counting the additional auxiliary data for smooth interpolation. Fortunately, an important observation is that BRDFs are intrinsically bi-variate spherical functions. This implies that it would be better to represent BRDFs in their intrinsic domain using multivariate SRBFs.

Table 7.1: Statistics and timing measurements of different SRBF representations for BRDFs, including the *scattered univariate SRBF* (SU-SRBF) representation and the *parameterized multivariate SRBF* (PM-SRBF) representation. All compressed data were stored as half-precision (16-bit) floating point numbers [65].

Material	Blue Metallic Paint		Krylon Blue		Nickel	
Illumination/view directions	1536		1536		1536	
Raw data (MB)	27		27		27	
Representation	SU-SRBF	PM-SRBF	SU-SRBF	PM-SRBF	SU-SRBF	PM-SRBF
SRBFs: J	5	6	4	8	5	6
Parameterized variates: N'	-	3	-	4	-	3
Compressed data (KB)	90	0.15	72	0.25	90	0.15
Squared error ratio	0.36%	0.34%	1.01%	1.02%	0.51%	0.44%
Compression time (min.)	9.54	10.81	10.7	19.36	8.06	10.49

Based on the parameterized multivariate SRBF representation (Section 4.3), we can approximate the BRDF $\rho(\omega_l, \omega_v)$ with total J multivariate SRBFs by Equation 4.10 with the parameterization functions defined in Equation 4.11. It is then straightforward to apply Algorithm 4.3 to derive the SRBF parameters by solving the following unconstrained least-squares optimization problem:

$$\min_{\Upsilon^{(\rho)}} \frac{1}{2} \int_{\mathbb{S}^2} \int_{\mathbb{S}^2} \left(\rho(\omega_l, \omega_v) - \hat{\rho}'\left(\Psi^{(\rho)}\right) \right)^2 d\omega_l d\omega_v, \tag{7.5}$$

where $\Upsilon^{(\rho)} = \left\{ \{\beta_j^{(\rho)}, \Xi_j^{(\rho)}, \Lambda_j^{(\rho)}\}_{j=1}^{J}, \{\Theta_n^{(\rho)}\}_{n=1}^{N'} \right\}$ for $\Xi_j^{(\rho)} = \{\xi_{j,n}^{(\rho)}\}_{n=1}^{N'}$ and $\Lambda_j^{(\rho)} = \{\lambda_{j,n}^{(\rho)}\}_{n=1}^{N'}$.

Run-Time Rendering

Unlike univariate SRBFs, the rendering process of approximated BRDFs based on multivariate SRBFs is quite simple and intuitive. Since the proposed parameterized multivariate SRBF representation is a continuous functional model, no additional interpolation techniques for smooth transitions across different illumination (or view) directions are required. This implies that there is no need to generate auxiliary data for efficient run-time rendering. As a result, the rendering process corresponds to directly evaluating Equations 4.10 and 4.11 on GPUs. Note that the derived multivariate SRBF parameters are frequently compact enough to be stored in GPU buffers rather than textures, which will slightly reduce data access time.

7.2.3 Experimental Results

The experiments and simulation timings of BRDF approximation were conducted and measured on a workstation with an Intel Core 2 Extreme QX9650 CPU, an NVIDIA GeForce 9800 GX2 graphics card, and 8 gigabytes main memory under Microsoft Windows Vista operating sys-

Table 7.2: Comparisons of the rendering performance of different SRBF representations for BRDFs, including the *scattered univariate SRBF* (SU-SRBF) representation and the *parameterized multivariate SRBF* (PM-SRBF) representation. The screen resolution and the number of directional light sources were respectively set to 640×480 and 3. In the row *Auxiliary data*, we list the amount of additional data for efficient run-time rendering. All floating point numbers were stored in half precision [65].

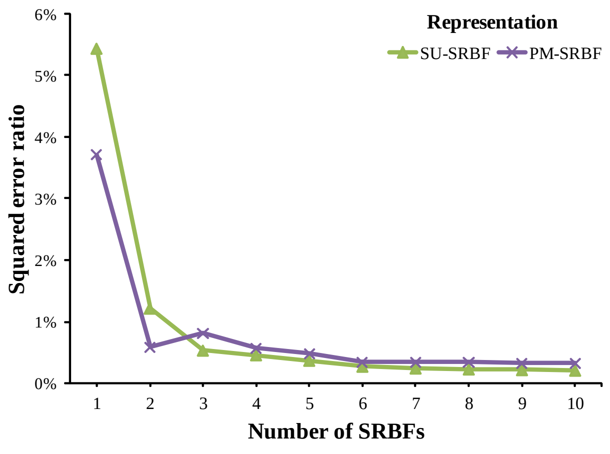
Model	Buddha		Bunny		Venus	
Material	Krylon Blue		Blue Metallic Paint		Nickel	
Vertices	25k		36k		25k	
Representation	SU-SRBF	PM-SRBF	SU-SRBF	PM-SRBF	SU-SRBF	PM-SRBF
Auxiliary data (KB)	1548	0	1548	0	1548	0
Total data (KB)	1620	0.25	1638	0.15	1638	0.15
Frames per second	129.35	270.54	112.23	548.79	142.41	672.36

tem. We also employed NVIDIA CUDA [128] to accelerate parts of the optimization process and Microsoft Direct3D 10 [15] for run-time rendering on GPUs. In our experiments, the univariate and multivariate Gaussian SRBFs were adopted to represent BRDFs with non-negative constraints on basis coefficients. The SRBF center and bandwidth sets were also constrained to be the same for red, green, and blue channels of a BRDF. To generate high-quality images, we employed per-pixel lighting and performed the rendering process mostly in the pixel shader. The measured BRDFs were provided in courtesy of Cornell University [140] and *Mitsubishi Electric Research Laboratories* (MERL) [113, 114, 120].

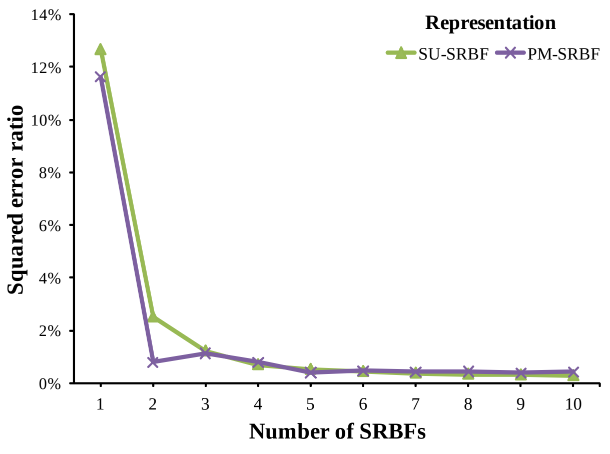
Table 7.1 compares the statistics and timing measurements of different SRBF representations for BRDFs, including the scattered univariate SRBF representation (Section 7.2.1) and the parameterized multivariate SRBF representation (Section 7.2.2). For parameterized multivariate SRBFs, the first three parameterization coefficient sets were initialized as the half-way, illumination, and view parameterizations before optimization, while the remainders were set to random real numbers. Figure 7.6 also plots the squared error ratio versus the number of SRBFs for different SRBF representations. It particularly shows that with the same number of SRBFs, the approximation errors of the parameterized multivariate SRBF representation are close to those of the scattered univariate SRBF representation, but the amount of compressed data for the parameterized multivariate SRBF representation is much more compact from Table 7.1.

In Figures 7.7–7.10, we further demonstrate the reconstructed and rendered images of various measured BRDFs based on the proposed SRBF representations. These figures show that the proposed approaches can provide high-quality and accurate approximation for real-world BRDFs with different reflectance properties. Table 7.2 also compares the rendering performance of the proposed SRBF representations for BRDFs at run-time. It clearly reveals that both SRBF representations can easily achieve real-time rendering rates for a medium-size model.

In general, the multivariate SRBF representation for BRDFs is superior to the univariate



(a) *Blue Metallic Paint*



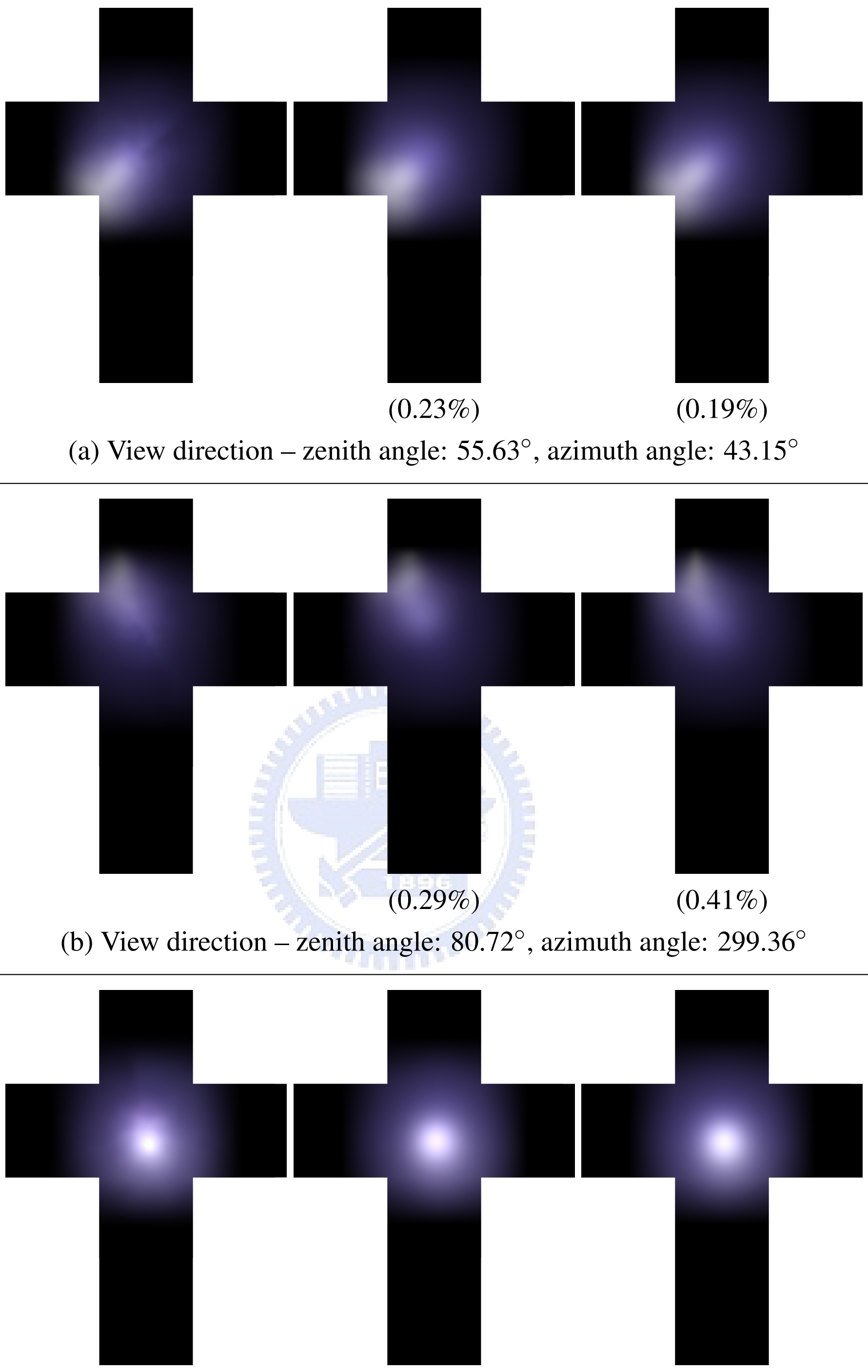
(b) *Nickel*

Figure 7.6: Plots of the squared error ratio versus the number of basis functions based on different SRBF representations for BRDFs, including *scattered univariate SRBFs* (SU-SRBF) and *parameterized multivariate SRBFs* (PM-SRBF).

model in terms of storage space and rendering performance. Univariate SRBFs usually require less number of basis functions to achieve approximation errors comparable to multivariate SRBFs, but they have to respectively tabulate the parameters of different view directions. This certainly results in a limited compression ratio for the univariate representation. Meanwhile, multivariate SRBFs can additionally exploit the coherence in different view directions to provide a more compact representation for BRDFs. As for run-time rendering, univariate SRBFs need auxiliary texture data to efficiently interpolate the reconstructed reflectance values of nearby view directions for smooth transitions. While the auxiliary data consume GPU memory bandwidth, the interpolation process further increases the total computational costs. By contrast, although the reconstruction costs of univariate SRBFs are theoretically lower, the approximated BRDFs based on multivariate SRBFs are much more compact so that GPU memory bandwidth can be substantially saved. Interpolation is also inherent in the multivariate SRBF model without additional memory or computational overhead. From the discussions above, it might seem that the univariate SRBF representation for BRDFs is rather useless. However, the univariate model is suitable for some graphics applications, such as importance sampling in ray tracing algorithms. We will present more details about this topic in Section 7.3.

7.3 Importance Sampling of Direct Illumination

In computer graphics, one of the most representative global illumination algorithms might be ray tracing. Pioneered by Whitted [202], ray tracing simulates global light transport by shooting rays from the viewpoint, finding the nearest intersection points with the scene, and deterministically generating new rays for further tracing from each intersection point. This process is then repeated for each new ray in a recursive way until no intersection points are found. As a result,

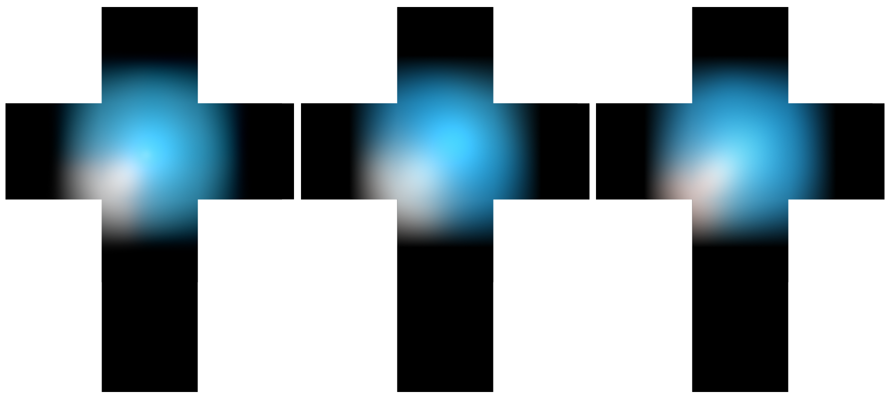


(a) View direction – zenith angle: 55.63°, azimuth angle: 43.15°

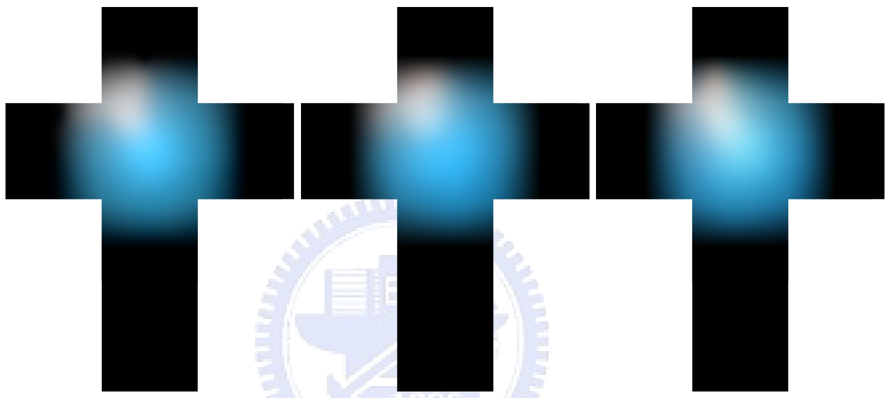
(b) View direction – zenith angle: 80.72°, azimuth angle: 299.36°

(c) View direction – zenith angle: 17.68°, azimuth angle: 101.31°

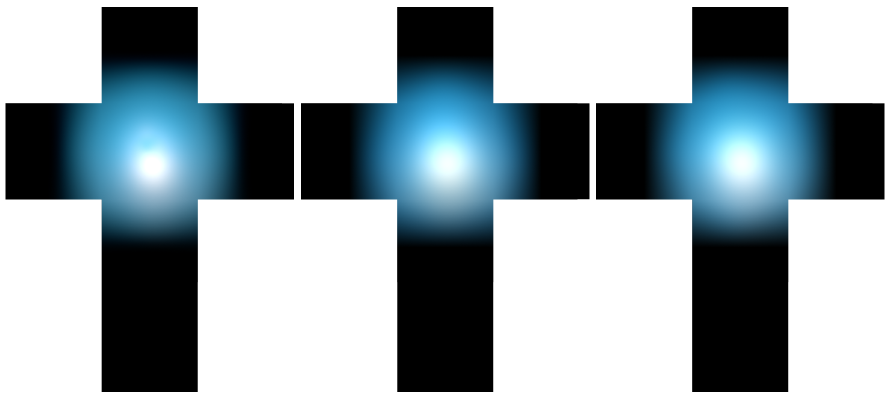
Figure 7.7: Reconstructed images of the BRDF *Blue Metallic Paint* based on different SRBF representations. From left to right in each row: raw data; scattered univariate SRBFs; parameterized multivariate SRBFs. The squared error ratio of each reconstructed image is shown in parentheses. For the configurations of each SRBF representation, please refer to Table 7.1.



(a) View direction – zenith angle: 55.63°, azimuth angle: 43.15°

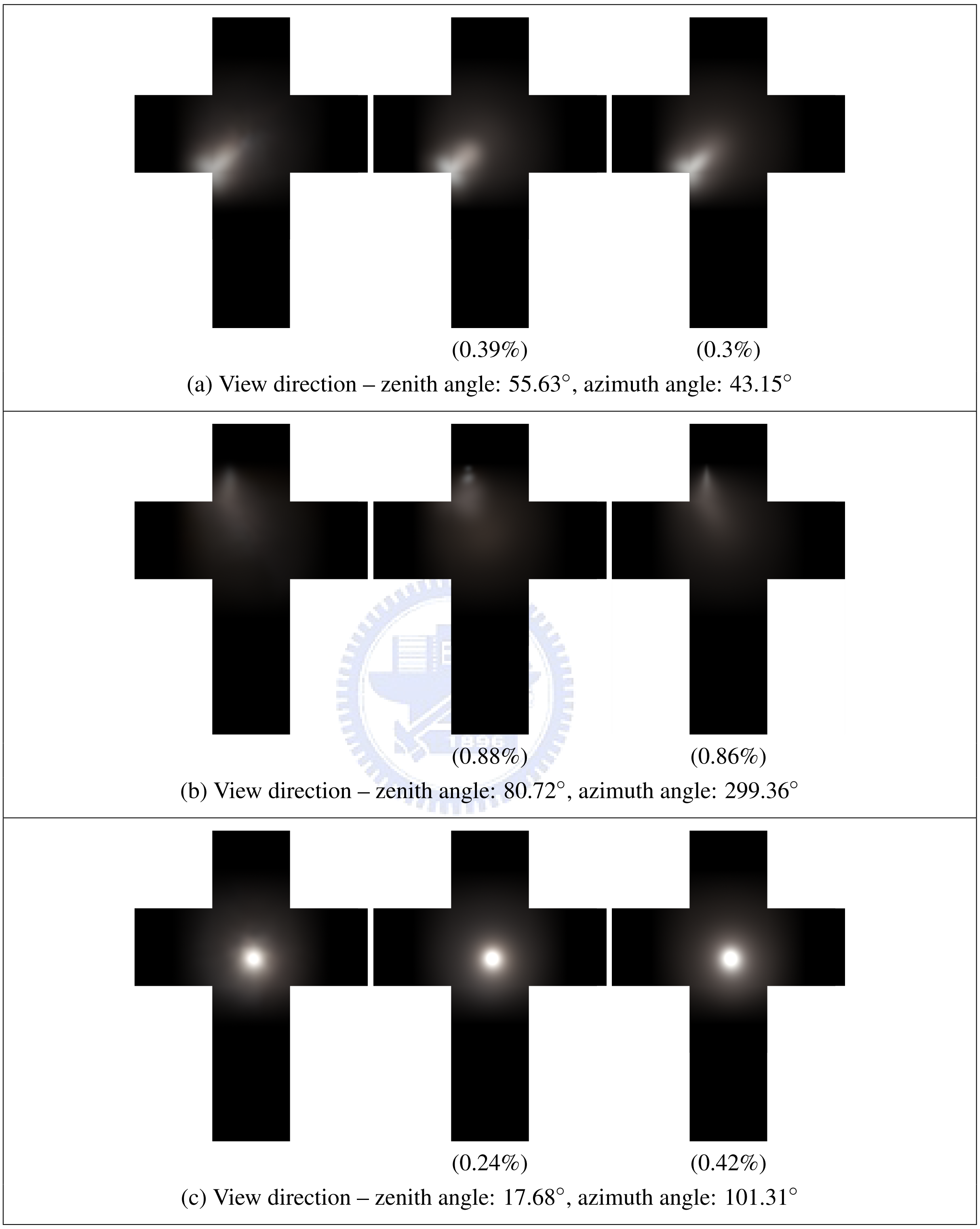


(b) View direction – zenith angle: 80.72°, azimuth angle: 299.36°



(c) View direction – zenith angle: 17.68°, azimuth angle: 101.31°

Figure 7.8: Reconstructed images of the BRDF *Krylon Blue* based on different SRBF representations. From left to right in each row: raw data; scattered univariate SRBFs; parameterized multivariate SRBFs. The squared error ratio of each reconstructed image is shown in parentheses. For the configurations of each SRBF representation, please refer to Table 7.1.

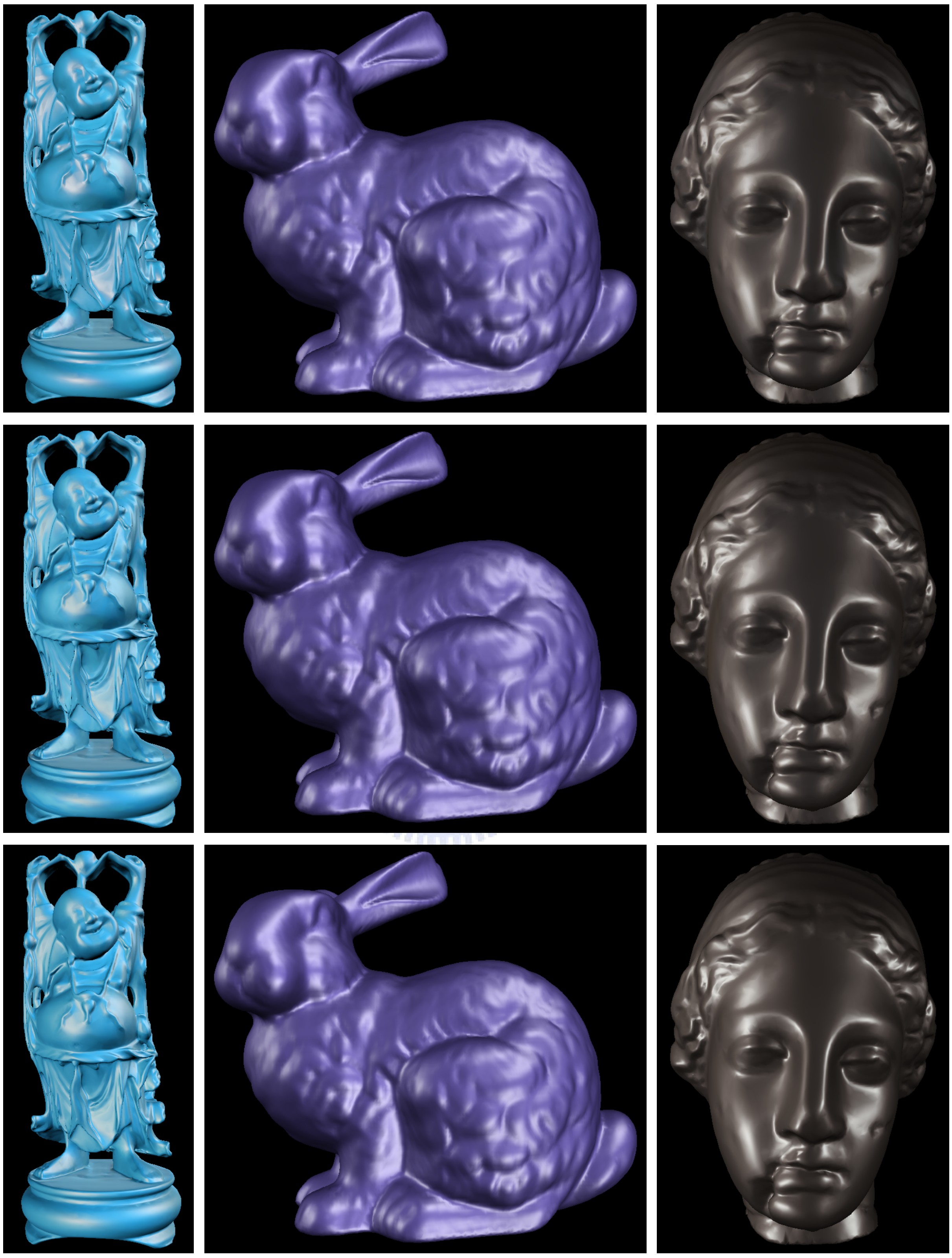


(a) View direction – zenith angle: 55.63°, azimuth angle: 43.15°

(b) View direction – zenith angle: 80.72°, azimuth angle: 299.36°

(c) View direction – zenith angle: 17.68°, azimuth angle: 101.31°

Figure 7.9: Reconstructed images of the BRDF *Nickel* based on different SRBF representations. From left to right in each row: raw data; scattered univariate SRBFs; parameterized multivariate SRBFs. The squared error ratio of each reconstructed image is shown in parentheses. For the configurations of each SRBF representation, please refer to Table 7.1.



(a) *Buddha* with *Krylon Blue* (b) *Bunny* with *Blue Metallic Paint* (c) *Venus* with *Nickel*

Figure 7.10: Rendered images based on different SRBF representations for BRDFs. From top to bottom: raw data: scattered univariate SRBFs; parameterized multivariate SRBFs. For the configurations of SRBF representations and run-time rendering, please refer to Tables 7.1 and 7.2.

many global illumination effects, such as reflection, refraction, and shadows, can be faithfully captured as inherent parts of the simulation. Interested readers may refer to the book by Pharr and Humphreys [137] for a comprehensive review on ray tracing algorithms.

Beyond the deterministic approach, ray tracing was later extended into a stochastic process by Cook [24] and Kajiya [73] to prevent aliasing artifacts in synthesized images. The key concept of the stochastic framework is to employ Monte Carlo sampling to estimate the integral in the rendering equation [73]. Nevertheless, a brute-force Monte Carlo method based on random samples is frequently time-consuming and tends to produce noisy results.

To solve this problem, importance sampling is a widely-adopted variance reduction technique for increasing the efficiency of Monte Carlo methods. It generates samples according to their contributions to the final estimation, namely their 'importance'. If important samples are considered more frequently, the variance of estimation can be greatly reduced even with fewer number of samples. Based on this concept, we propose an importance sampling method of direct illumination for stochastic ray tracing algorithms. Our approach relies on producing new rays for intersection points from the distributions of their associated BRDFs. The proposed univariate SRBF representation for BRDFs (Section 7.2.1) especially facilitates identifying important sampling rays in an efficient way.

Portions of the proposed importance sampling algorithm were also published in our paper [178]. Nevertheless, we focus more on its mathematical part and describe the fundamental concept from a different point of view in this dissertation.

7.3.1 Problem Formulation

For distant light sources and direct illumination, the rendering equation [73] over the unit sphere $\mathbb{S}^2$ can be written as

$$L_{o,\mathbf{p}}(\omega_v) = \int_{\mathbb{S}^2} L_{in}(\omega_l)\rho_{\mathbf{p}}(\omega_l, \omega_v)V_{\mathbf{p}}(\omega_l)\,|\omega_l \cdot \mathbf{n}_{\mathbf{p}}|\,d\omega_l, \tag{7.6}$$

where $L_{o,\mathbf{p}}(\omega_v) \in \mathbb{R}$ denotes the exitant radiance from a surface point $\mathbf{p} \in \mathbb{R}^3$ in view direction $\omega_v \in \mathbb{S}^2$, $L_{in}(\omega_l) \in \mathbb{R}$ is the incident radiance from a distant light source in illumination direction $\omega_l \in \mathbb{S}^2$. Moreover, $\rho_{\mathbf{p}}(\omega_l, \omega_v) \in \mathbb{R}$, $V_{\mathbf{p}}(\omega_l) \in \{x \in \mathbb{R} \mid 0 \le x \le 1\}$, and $\mathbf{n}_{\mathbf{p}} \in \mathbb{S}^2$ respectively represent the BRDF, the visibility function from the light source in direction ω_l, and the surface normal at point $\mathbf{p}$.

To estimate the integral in Equation 7.6 using important sampling, we can generate a set of N illumination directions $\Omega_l = \{\omega_{l,n}\}_{n=1}^{N}$ from a conditional probability distribution function $\Pr(\omega_l|\mathbf{p}, \omega_v)$ on $\mathbb{S}^2$, and apply the following Monte Carlo estimator:

$$L_{o,\mathbf{p}}(\omega_v) \approx \hat{L}_{o,\mathbf{p}}(\omega_v) = \frac{1}{N}\sum_{n=1}^{N} \frac{L_{in}(\omega_{l,n})\rho_{\mathbf{p}}(\omega_{l,n}, \omega_v)V_{\mathbf{p}}(\omega_{l,n})\,|\omega_l \cdot \mathbf{n}_{\mathbf{p}}|}{\Pr(\omega_{l,n}|\mathbf{p}, \omega_v)}. \tag{7.7}$$

There are many choices of the distribution function $\Pr(\omega_l|\mathbf{p}, \omega_v)$. For example, one may generate sampling directions from the distribution of the BRDF [88], the incident radiance [2], or the product of them [22].

In this dissertation, we employ BRDF importance sampling and construct $\Pr(\omega_l|\mathbf{p}, \omega_v)$ from the distribution of $\rho_{\mathbf{p}}(\omega_l, \omega_v)$ for a given view direction. As described in Section 7.2.1, suppose that $\rho_{\mathbf{p}}(\omega_l, \omega_v)$ is view discretized into a set of illumination-dependent reflectance functions $\{\rho_{\mathbf{p},i}(\omega_l) \in \mathbb{R}\}_{i=1}^{I_{\Omega_v}}$ with respect to a set of view directions $\Omega_v = \{\omega_{v,i} \in \mathbb{S}^2\}_{i=1}^{I_{\Omega_v}}$. Each discretized reflectance function is then approximated with scattered univariate SRBFs by

$$\forall i,\ \rho_{\mathbf{p},i}(\omega_l) \approx \hat{\rho}_{\mathbf{p},i}(\omega_l) = \sum_{j=1}^{J} \beta_j^{(\rho_{\mathbf{p}},i)} G\big(\omega_l \cdot \xi_j^{(\rho_{\mathbf{p}},i)} | \lambda_j^{(\rho_{\mathbf{p}},i)}\big), \tag{7.8}$$

where $\big\{\beta_j^{(\rho_{\mathbf{p}},i)} \in \mathbb{R}\big\}_{j=1}^{J}$, $\big\{\xi_j^{(\rho_{\mathbf{p}},i)} \in \mathbb{S}^2\big\}_{j=1}^{J}$, and $\big\{\lambda_j^{(\rho_{\mathbf{p}},i)} \in \mathbb{R}\big\}_{j=1}^{J}$ respectively denote the basis coefficient set, the center set, and the bandwidth set of $\rho_{\mathbf{p},i}(\omega_l)$. Here, we choose to discretize the view domain of $\rho_{\mathbf{p}}(\omega_l, \omega_v)$ just because rays are shot from the viewpoint. If the ray tracing process begins from light sources, one should discretize the illumination domain of $\rho_{\mathbf{p}}(\omega_l, \omega_v)$ instead. Note that to employ the approximated results for importance sampling, we need to additionally constrain that all the basis coefficients are non-negative, namely $\forall i,\ \forall j,\ \beta_j^{(\rho_{\mathbf{p}},i)} \geq 0$. Each approximated illumination-dependent reflectance function is then transformed into a conditional probability distribution function as

$$\forall i,\ \Pr(\omega_l|\mathbf{p}, \omega_{v,i}) = \sum_{j=1}^{J} \beta_j'^{(\rho_{\mathbf{p}},i)} G'\big(\omega_l \cdot \xi_j^{(\rho_{\mathbf{p}},i)} | \lambda_j^{(\rho_{\mathbf{p}},i)}\big), \tag{7.9}$$

where $\Pr(\omega_l|\mathbf{p}, \omega_{v,i})$ denotes the density distribution of $\hat{\rho}_{\mathbf{p},i}(\omega_l)$, and

$$\forall i,\ \forall j,\ \beta_j'^{(\rho_{\mathbf{p}},i)} = \frac{\beta_j^{(\rho_{\mathbf{p}},i)}}{\sum_{j=1}^{J} \beta_j^{(\rho_{\mathbf{p}},i)}}, \tag{7.10}$$

$$\forall i,\ \forall j,\ G'\big(\omega_l \cdot \xi_j^{(\rho_{\mathbf{p}},i)} | \lambda_j^{(\rho_{\mathbf{p}},i)}\big) = \frac{G\big(\omega_l \cdot \xi_j^{(\rho_{\mathbf{p}},i)} | \lambda_j^{(\rho_{\mathbf{p}},i)}\big)}{\displaystyle\int_{S^2} G\big(\omega_l \cdot \xi_j^{(\rho_{\mathbf{p}},i)} | \lambda_j^{(\rho_{\mathbf{p}},i)}\big) d\omega_l}. \tag{7.11}$$

From Equation 7.9, it is easy to verify that $\Pr(\omega_l|\mathbf{p}, \omega_{v,i})$ is in fact a mixture model for the distribution of $\rho_{\mathbf{p},i}(\omega_l)$. To sample an illumination direction from $\Pr(\omega_l|\mathbf{p}, \omega_{v,i})$, we can first select the j-th univariate SRBF for sampling with probability $\beta_j'^{(\rho_{\mathbf{p}},i)}$, and then generate an illumination direction on $\mathbb{S}^2$ from the j-th normalized univariate SRBF $G'\big(\omega_l \cdot \xi_j^{(\rho_{\mathbf{p}},i)} | \lambda_j^{(\rho_{\mathbf{p}},i)}\big)$.

Since a normalized univariate SRBF is circularly symmetric around its center, sampling a direction on $\mathbb{S}^2$ from $G'\big(\omega_l \cdot \xi_j^{(\rho_{\mathbf{p}},i)} | \lambda_j^{(\rho_{\mathbf{p}},i)}\big)$ can be solved by a two-stage process [178, 200]. First, $G'\big(\omega_l \cdot \xi_j^{(\rho_{\mathbf{p}},i)} | \lambda_j^{(\rho_{\mathbf{p}},i)}\big)$ is transformed into a marginal density distribution in the interval

$[0, \pi]$ by

$$\begin{aligned} \Pr\left(\phi_1 | \lambda_j^{(\rho_{\mathbf{p}},i)}\right) &= \int_0^{2\pi} G'\left(\omega_l \cdot \xi_j^{(\rho_{\mathbf{p}},i)} | \lambda_j^{(\rho_{\mathbf{p}},i)}\right) \sin\phi_1 d\phi_2 \\ &= \int_0^{2\pi} G'\left(\cos\phi_1 | \lambda_j^{(\rho_{\mathbf{p}},i)}\right) \sin\phi_1 d\phi_2, \end{aligned} \tag{7.12}$$

where ϕ_1 and ϕ_2 respectively denote the zenith and azimuth angles of the sampling direction ω_l in the spherical coordinate system. The zenith angle thus can be generated from $\Pr\left(\phi_1 | \lambda_j^{(\rho_{\mathbf{p}},i)}\right)$ using the traditional inversion method or Metropolis-Hastings algorithm. Second, the azimuth angle is then determined by uniformly generating a random real number in the interval $[-\pi, \pi]$. Note that the sampled zenith and azimuth angles are not absolute coordinates, but with respect to the SRBF center $\xi_j^{(\rho_{\mathbf{p}},i)}$. The final sampling direction thus is given by increasing (or decreasing) the zenith angle of $\xi_j^{(\rho_{\mathbf{p}},i)}$ by ϕ_1, and then rotating the resulting direction about $\xi_j^{(\rho_{\mathbf{p}},i)}$ by angle ϕ_2.

7.3.2 Importance Sampling Algorithm

Based on the scattered univariate SRBF representation for BRDFs (Equations 7.8 and 7.9), the proposed importance sampling algorithm for stochastic ray tracing thus consists of the following steps:

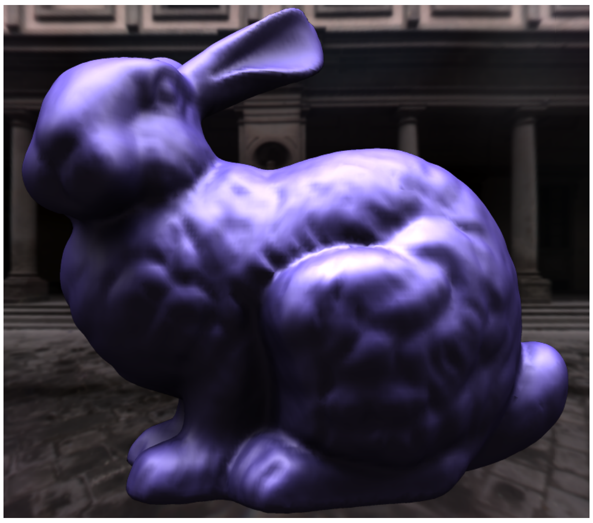
1. From the viewpoint, shoot a ray that passes a sample point on the image plane, and find the nearest intersection point $\mathbf{p}$ of this ray with the scene.
2. At point $\mathbf{p}$, search for the K nearest neighbors of current novel view direction ω_v from Ω_v.
3. Linearly interpolate the density distribution (Equation 7.9) of each neighboring direction to obtain $\Pr(\omega_l | \mathbf{p}, \omega_v)$.
4. Sample total N illumination directions from $\Pr(\omega_l | \mathbf{p}, \omega_v)$.
5. The final shading color is then given by evaluating Equation 7.7.

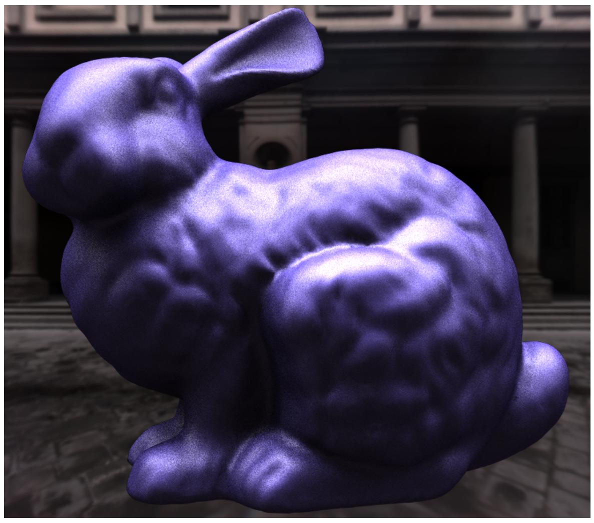
7.3.3 Experimental Results

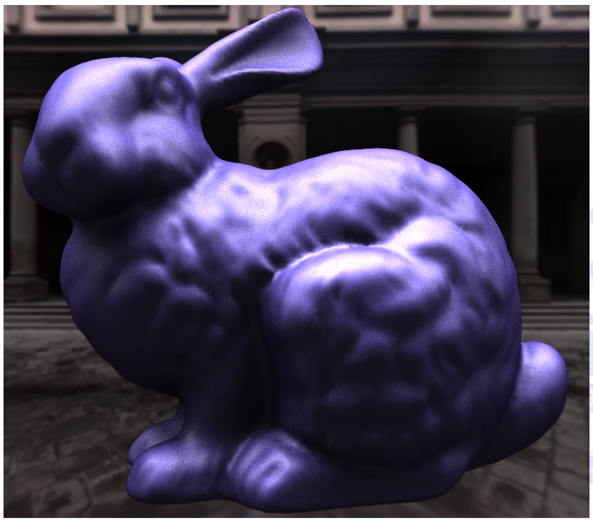
The experiments and simulation timings of importance sampling were conducted and measured on a workstation with an Intel Core 2 Extreme QX9650 CPU, an NVIDIA GeForce 9800 GX2 graphics card, and 8 gigabytes main memory under Microsoft Windows Vista operating system. We modified the ray tracing program from the book by Pharr and Humphreys [137] to support the proposed importance sampling algorithm on CPUs. Besides BRDF sampling, multiple importance sampling [186, 187] is additionally employed to combine the sampled results of BRDFs and the lighting environment. In our experiments, the configurations of the scattered univariate SRBF representation for BRDFs were the same as those in Table 7.1 (Section 7.2.3). For each intersection point, the number of nearest neighboring view directions for linear interpolation was set to 4.

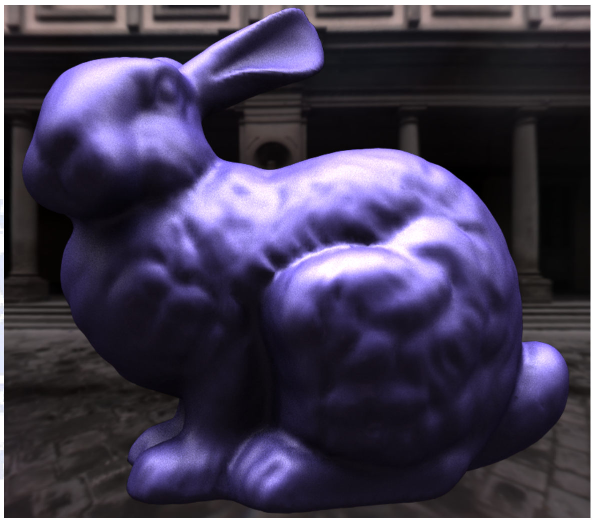
Figures 7.11–7.13 compare the rendered images with different numbers of samples per pixel based on the proposed importance sampling algorithm, with the configurations of the scattered univariate SRBF representation for each BRDF listed in Table 7.1. In all rendered images, the screen resolution was set to 640×480. The reference images were generated on GPUs from raw BRDF data by uniformly sampling a large number of rays over the unit sphere at each intersection point. As for the results of importance sampling, half of the samples were utilized for BRDF sampling, with the other half for incidence radiance sampling. Typically, a configuration of 800 samples per pixel can provide a visually pleasing result, with a computation time of less than 30 minutes on average. Since the current implementation of stochastic ray tracing neither utilizes the multi-core capability of CPUs nor allows operations to be accelerated on GPUs, there is still much room for improving the overall performance. Indeed, the bottleneck of our system mostly lies in the fundamental ray tracing process, not in the proposed importance sampling algorithm.

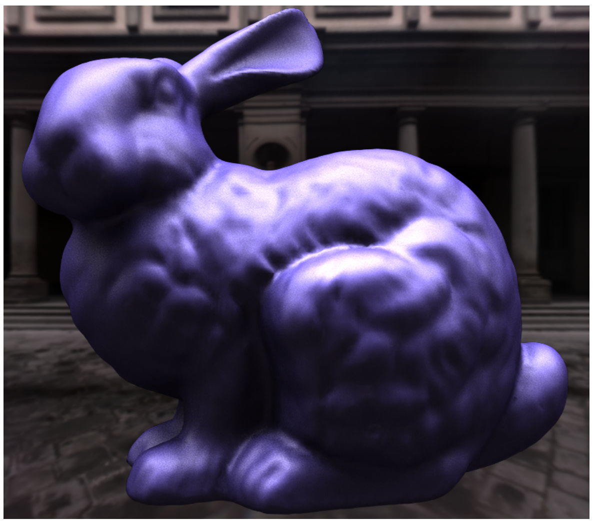
Note that one can also generate samples from the product of a BRDF and a lighting environment [22]. It is rather intuitive to extend the proposed importance sampling algorithm to support this concept. By exploiting the rotational invariance of univariate SRBFs, the product of a BRDF and a lighting environment can be efficiently computed, and then analyzed using the proposed approach to determine sampling directions. This approach typically requires much fewer samples per pixel to achieve similar image quality. Its overall computation time is also expected to be faster than previous importance sampling methods. A detailed description of the product sampling based on the scattered univariate SRBF representation is beyond the scope of this dissertation. Interested readers may refer to [71, 178].


(a) Reference image (∼400000 samples per pixel)


(b) 200 samples per pixel


(c) 400 samples per pixel


(d) 600 samples per pixel


(e) 800 samples per pixel

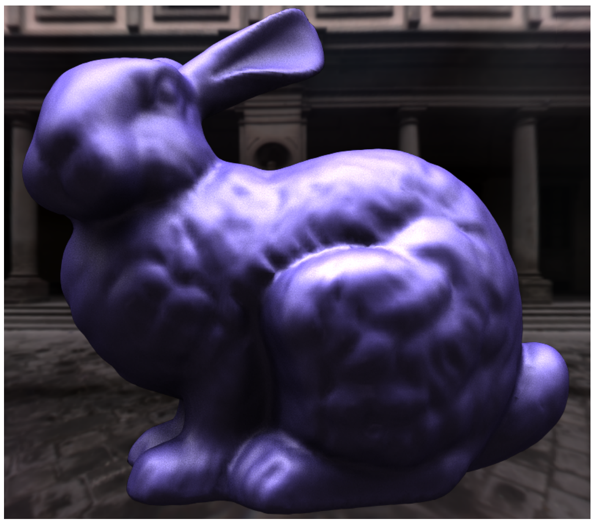
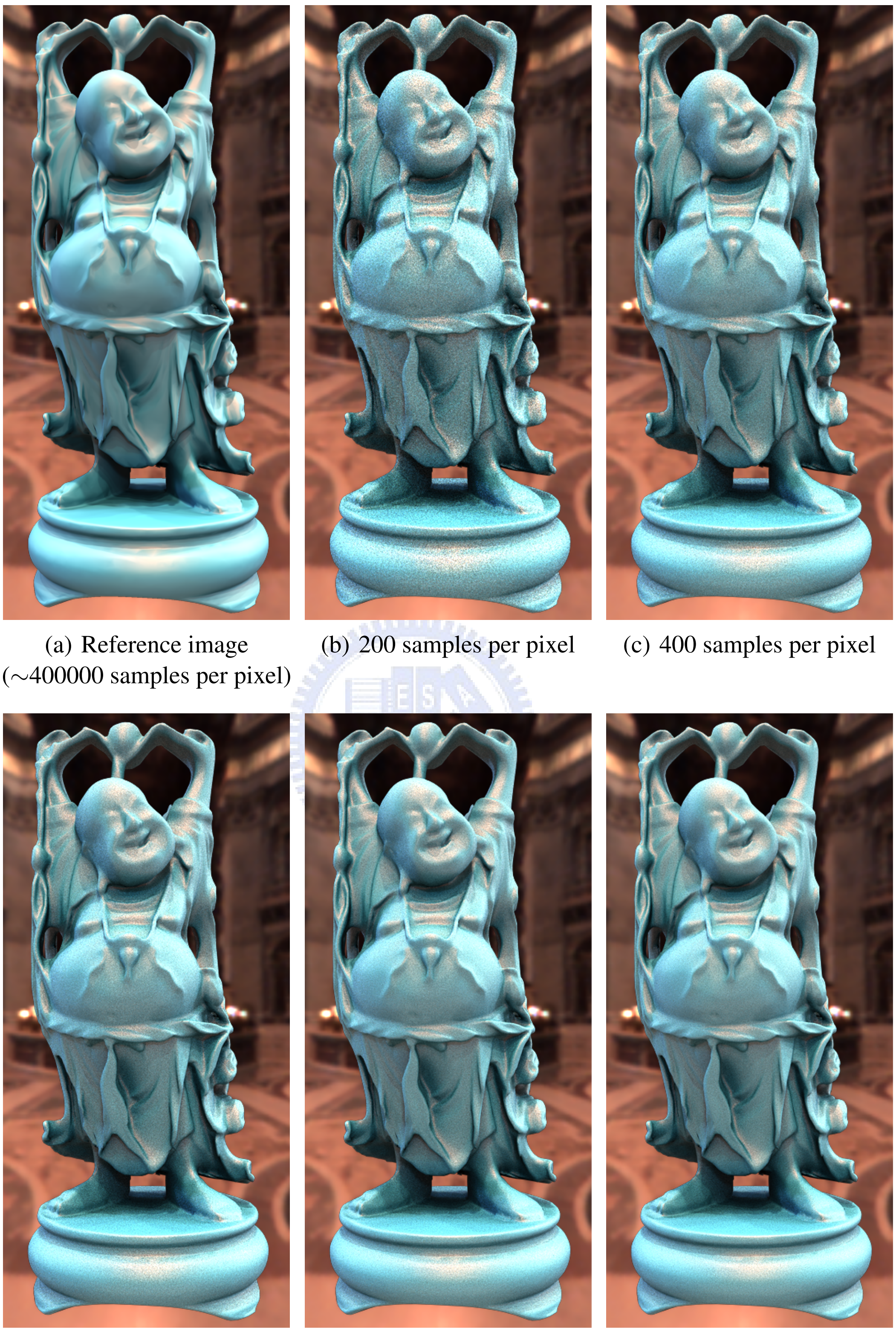

(f) 1000 samples per pixel

Figure 7.11: Comparisons of rendered images with different numbers of samples per pixel based on the proposed importance sampling algorithm for the model *Bunny* with the BRDF *Blue Metallic Paint*.



(a) Reference image (∼400000 samples per pixel)

(b) 200 samples per pixel

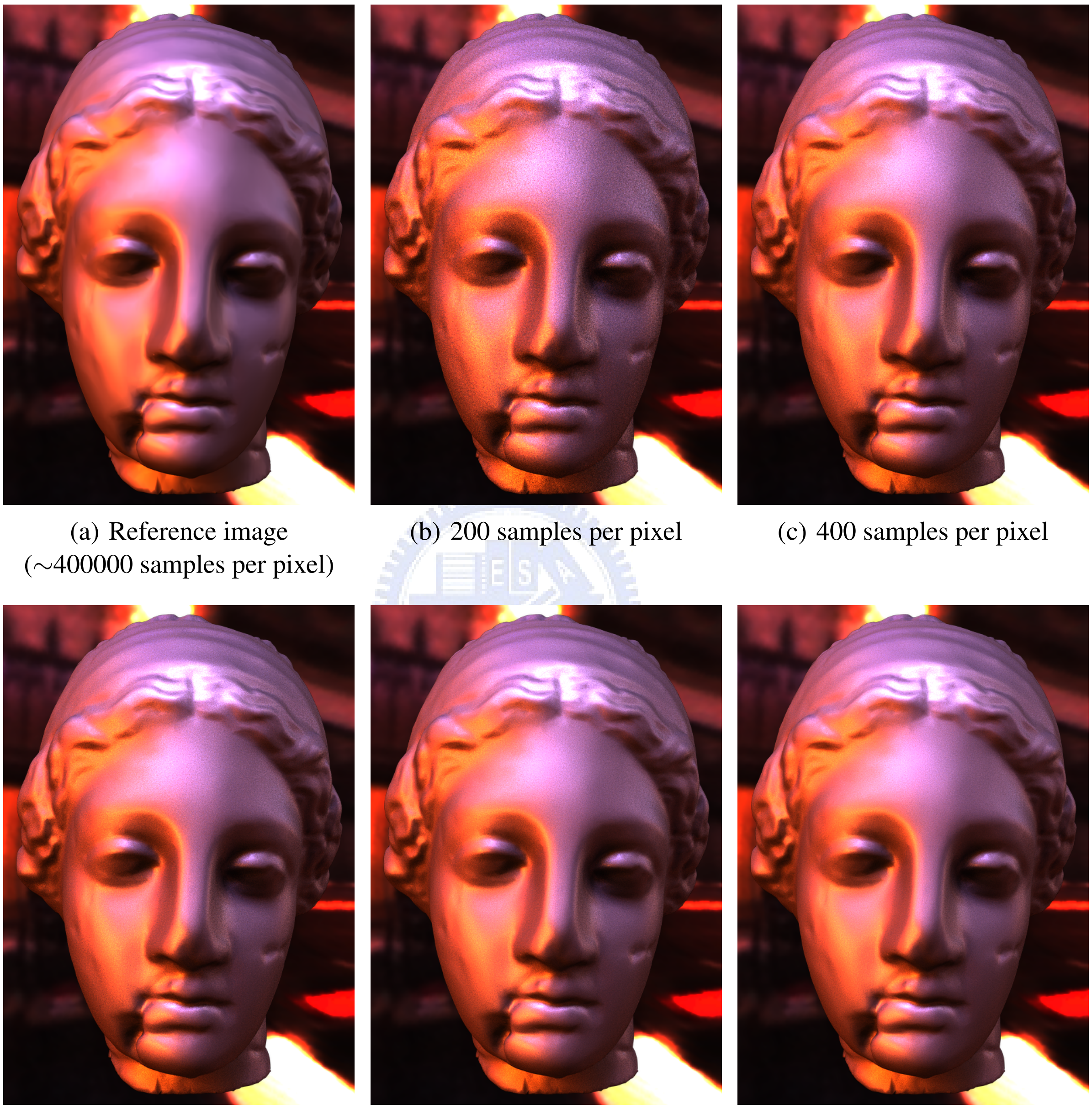
(c) 400 samples per pixel

(d) 600 samples per pixel

(e) 800 samples per pixel

(f) 1000 samples per pixel

Figure 7.12: Comparisons of rendered images with different numbers of samples per pixel based on the proposed importance sampling algorithm for the model *Buddha* with the BRDF *Krylon Blue*.



(a) Reference image (∼400000 samples per pixel)

(b) 200 samples per pixel

(c) 400 samples per pixel

(d) 600 samples per pixel

(e) 800 samples per pixel

(f) 1000 samples per pixel

Figure 7.13: Comparisons of rendered images with different numbers of samples per pixel based on the proposed importance sampling algorithm for the model *Venus* with the BRDF *Nickel*.

Chapter 8

Application II: Spatially-Varying Appearance Models

Modeling spatially-varying surface appearance for real-time graphics and vision applications is a more challenging problem than approximating illumination and reflectance functions. For photo-realistic image synthesis, the large amount of appearance data not only consumes considerable storage space but also frequently becomes the performance bottleneck of run-time rendering process. In the first section of this chapter, we present two categories of powerful compression methods for *bidirectional texture functions* (BTFs) [28], including tensor approximation algorithms (Section 8.1.1) and multivariate *spherical radial basis functions* (SRBFs) (Section 8.1.2). Experimental results further reveal that real-time rendering performance can be easily achieved with the proposed methods.

Moreover, previous spatially-varying appearance materials, such as BTFs, only permit realistic illumination response of object surfaces at the meso-scale, but the complex surface micro-geometry is not explicitly modeled. In the second section of this chapter, we thus introduce a novel spatially-varying appearance model, namely *view-dependent occlusion texture function* (VOTF), which records meso-scale visibility information from different view directions. In this way, the proposed VOTFs can be easily combined with existing appearance models to visualize silhouettes at the meso-scale.

8.1 Bidirectional Texture Functions

Bidirectional texture functions [28] generalize *bidirectional reflectance distribution functions* (BRDFs) to contain textural patterns of real-world object surfaces. They capture spatially-varying surface appearance and reflectance that change with respect to the illumination and view directions. A BTF is frequently a six-dimensional function of three variables: ω_l, ω_v, and $\mathbf{t} = \left[x, y\right]^T$, where ω_l and ω_v are respectively the illumination and view directions on the unit sphere $\mathbb{S}^2$, and $\mathbf{t}$ denotes the two-dimensional spatial coordinates, x and y, of a texel. Figure 8.1

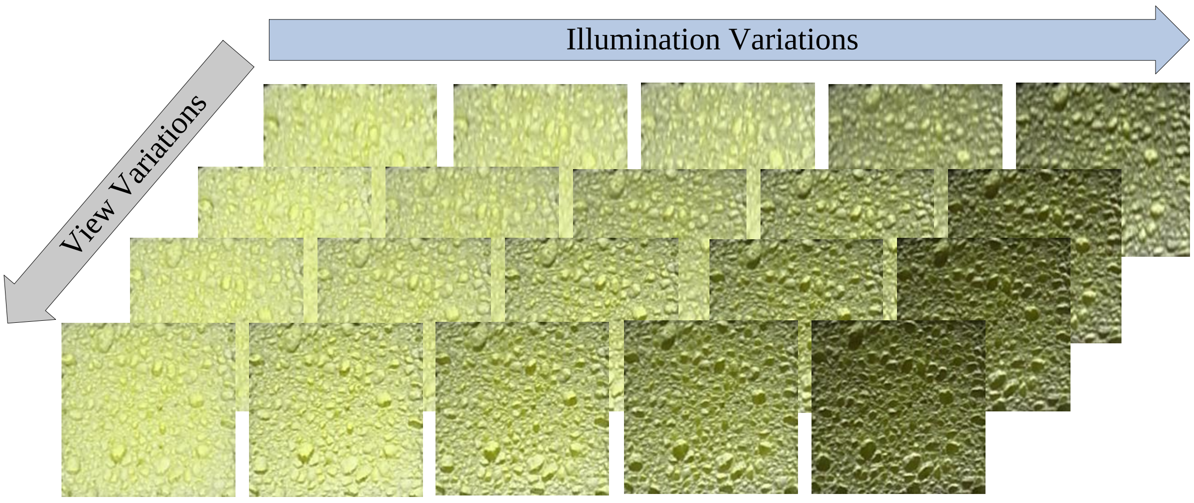


Figure 8.1: An example of a BTF from the volumetric surface texture database [84]. Some images of the material *Sponge* are shown in this figure. While images along the blue arrow were captured from the same view direction but under different illumination conditions, those along the gray arrow were acquired under the same illumination condition but from different view directions.

shows an example of measured BTFs from the volumetric surface texture database [84].

Although the illumination effects of real-world object surfaces can be faithfully captured with BTFs, we usually have to tabulate several gigabytes of raw data for a single BTF. This is certainly impractical for photo-realistic image synthesis in real-time rendering applications. In the following sub-sections, we therefore present two categories of powerful compression methods to reduce the amount of BTF data and accelerate run-time rendering performance. The first category (Section 8.1.1) relies on organizing a BTF as a multi-dimensional array so that tensor approximation algorithms, including clustered tensor approximation (Chapter 5) and K-clustered tensor approximation (Chapter 6), can be directly applied to compress it. The second category (Section 8.1.2) is instead based on the parameterized multivariate SRBF representation introduced in Chapter 4. This parametric representation is especially suitable to high-speed rendering on GPUs for performance-intensive applications. Finally, detailed comparisons of these two categories of compression methods and the choice between them in a particular application are discussed in Section 8.1.3.

8.1.1 Tensor Approximation Algorithms

Problem Formulation

For tensor approximation algorithms, we organize a BTF $B(\omega_l, \omega_v, \mathbf{t})$ as a fourth order tensor $\boldsymbol{\mathcal{A}}^{(B)} \in \mathbb{R}^{I_{\omega_l}^{(B)} \times I_{\omega_v}^{(B)} \times I_x^{(B)} \times I_y^{(B)}}$ to retain its intrinsic structures, where $I_{\omega_l}^{(B)}$ and $I_{\omega_v}^{(B)}$ denote the numbers of sampled illumination and view directions, and $I_x^{(B)}$ as well as $I_y^{(B)}$ specify the spatially horizontal and vertical resolutions. Note that one may organize a BTF as a sixth order tensor by using the spherical coordinate system for illumination and view directions. This will not collapse the intrinsic structures of directions on $\mathbb{S}^2$, and could exploit the coherence in different zenith/azimuth angles. Nevertheless, we do not adopt this approach since the sampled illumination and view directions of a BTF are usually very sparse. There may not be much room for reduction in the zenith and azimuth modes.

Given the reduced ranks of each mode, namely $R^{(B)}_{\omega_l}$, $R^{(B)}_{\omega_v}$, $R^{(B)}_{x}$, and $R^{(B)}_{y}$, we then directly apply *N-mode singular value decomposition* (N-SVD) [33] to decompose $\boldsymbol{\mathcal{A}}^{(B)}$ as

$$\boldsymbol{\mathcal{A}}^{(B)} \approx \hat{\boldsymbol{\mathcal{A}}}^{(B)} = \boldsymbol{\mathcal{Z}}^{(B)} \times_{\omega_l} \mathbf{U}^{(B)}_{\omega_l} \times_{\omega_v} \mathbf{U}^{(B)}_{\omega_v} \times_{x} \mathbf{U}^{(B)}_{x} \times_{y} \mathbf{U}^{(B)}_{y}, \tag{8.1}$$

where $\boldsymbol{\mathcal{Z}}^{(B)} \in \mathbb{R}^{R^{(B)}_{\omega_l} \times R^{(B)}_{\omega_v} \times R^{(B)}_{x} \times R^{(B)}_{y}}$ is the decomposed core tensor, and $\mathbf{U}^{(B)}_{\omega_l} \in \mathbb{R}^{I^{(B)}_{\omega_l} \times R^{(B)}_{\omega_l}}$, $\mathbf{U}^{(B)}_{\omega_v} \in \mathbb{R}^{I^{(B)}_{\omega_v} \times R^{(B)}_{\omega_v}}$, $\mathbf{U}^{(B)}_{x} \in \mathbb{R}^{I^{(B)}_{x} \times R^{(B)}_{x}}$, as well as $\mathbf{U}^{(B)}_{y} \in \mathbb{R}^{I^{(B)}_{y} \times R^{(B)}_{y}}$ are respectively the mode-ω_l, mode-ω_v, mode-x, and mode-y basis matrices. As described in Section 5.1.2, this decomposition can be obtained by employing Algorithm 5.1.

For *clustered tensor approximation* (CTA) and *K-clustered tensor approximation* (K-CTA), we set total $C^{(B)}$ clusters for the view mode and decompose $\boldsymbol{\mathcal{A}}^{(B)}$ as

$$\boldsymbol{\mathcal{A}}^{(B)} \approx \hat{\boldsymbol{\mathcal{A}}}^{(B)} = \sum_{c=1}^{C^{(B)}} \left(\boldsymbol{\mathcal{Z}}^{(B)}_{c} \times_{\omega_l} \mathbf{U}^{(B)}_{\omega_l,c} \times_{\omega_v} \mathbf{U}^{(B)}_{\omega_v,c} \times_{x} \mathbf{U}^{(B)}_{x,c} \times_{y} \mathbf{U}^{(B)}_{y,c} \right), \tag{8.2}$$

where $\boldsymbol{\mathcal{Z}}^{(B)}_{c} \in \mathbb{R}^{R^{(B)}_{\omega_l} \times R^{(B)}_{\omega_v} \times R^{(B)}_{x} \times R^{(B)}_{y}}$ denotes the core tensor of cluster c, and $\mathbf{U}^{(B)}_{\omega_l,c} \in \mathbb{R}^{I^{(B)}_{\omega_l} \times R^{(B)}_{\omega_l}}$, $\mathbf{U}^{(B)}_{\omega_v,c} \in \mathbb{R}^{I^{(B)}_{\omega_v} \times R^{(B)}_{\omega_v}}$, $\mathbf{U}^{(B)}_{x,c} \in \mathbb{R}^{I^{(B)}_{x} \times R^{(B)}_{x}}$, as well as $\mathbf{U}^{(B)}_{y,c} \in \mathbb{R}^{I^{(B)}_{y} \times R^{(B)}_{y}}$ respectively represent the mode-ω_l, mode-ω_v, mode-x, and mode-y basis matrices of cluster c. Note that the mode-ω_v basis matrices should satisfy the disjoint and membership constraints in Equation 5.6 for CTA, or the sparse and membership constraints in Equation 6.1 for K-CTA with total $K^{(B)}_{\omega_v}$ mixture clusters for a mode-ω_v sub-tensor. Then, Algorithms 5.2 and 6.1 can be respectively applied to derive the decomposition in Equation 8.2 for CTA and K-CTA.

The configurations of CTA and K-CTA for BTF compression are determined for the following reasons:

- Efficient texture filtering techniques on GPUs can be directly utilized by not clustering the x and y modes.

- For a complex BTF, a high mode-ω_v reduced rank is usually required to model large view-dependent variations in the appearance data, which substantially increases run-time rendering costs on GPUs. Clustering along the view mode allows us to reduce the costs in an adjustable manner.

- In our experience, the mode-ω_v reduced rank often dominates the perceptual quality of reconstructed BTFs and may need to be much higher than the mode-ω_l reduced rank.

According to the reconstructed results in previous articles [185, 192], we identify that if the mode-ω_v reduced rank is too low to capture the view variations in a BTF, the reconstructed images will become over-blurred or have strong ringing effects. Human eyes seem to be more sensitive to these artifacts than unrealistic illumination effects. Therefore, clustering along the view mode will reduce the performance penalty when we need a high mode-ω_v reduced rank.

Rendering Issues

Meso-structure synthesis: Synthesis over object surfaces is an important issue of spatially-varying materials due to the huge storage space required for high-resolution meso-structures. By extending the *appearance-space texture synthesis* (ASTS) [97], the compressed BTFs based on tensor approximation algorithms can be efficiently synthesized over object surfaces. Unlike ASTS that focuses on low-frequency and diffuse materials, the proposed approach permits the synthesis of all-frequency and view-dependent meso-structures over arbitrary surfaces. The key observation is that the flexible ASTS framework allows synthesizing various types of input data if the neighborhood similarity could be well estimated.

For a compressed BTF based on N-SVD, we first reconstruct the x and y modes of the decomposed core tensor as

$$\hat{\boldsymbol{\mathcal{Z}}}^{(B)} = \boldsymbol{\mathcal{Z}}^{(B)} \times_x \mathbf{U}_x^{(B)} \times_y \mathbf{U}_y^{(B)}, \tag{8.3}$$

where $\hat{\boldsymbol{\mathcal{Z}}}^{(B)} \in \mathbb{R}^{R_{\omega_l}^{(B)} \times R_{\omega_v}^{(B)} \times I_x^{(B)} \times I_y^{(B)}}$ denotes the reconstructed core tensor based on N-SVD. After unfolding the illumination and view modes of $\hat{\boldsymbol{\mathcal{Z}}}^{(B)}$ into one mode, $\hat{\boldsymbol{\mathcal{Z}}}^{(B)}$ becomes a third order tensor $\hat{\boldsymbol{\mathcal{Z}}}'^{(B)} \in \mathbb{R}^{(R_{\omega_l}^{(B)} R_{\omega_v}^{(B)}) \times I_x^{(B)} \times I_y^{(B)}}$ that can be utilized as the exemplar texture for meso-structure synthesis. After that, we follow ASTS to transform the exemplar texture into an appearance-space exemplar and finally synthesize a texture S of BTF spatial coordinates in the object surface atlas domain.

For a compressed BTF based on CTA or K-CTA, we can follow the above approach in almost the same way, but the pre-reconstruction in Equation 8.3 should be modified to let ASTS accurately estimate the neighborhood similarity. For CTA, we respectively reconstruct the x and y modes of each cluster core tensor as

$$\forall c,\ \hat{\boldsymbol{\mathcal{Z}}}_c^{(B)} = \boldsymbol{\mathcal{Z}}_c^{(B)} \times_x \mathbf{U}_{x,c}^{(B)} \times_y \mathbf{U}_{y,c}^{(B)}, \tag{8.4}$$

where $\hat{\boldsymbol{\mathcal{Z}}}_c^{(B)} \in \mathbb{R}^{R_{\omega_l}^{(B)} \times R_{\omega_v}^{(B)} \times I_x^{(B)} \times I_y^{(B)}}$ represents the reconstructed core tensor of cluster c based on CTA. Then, each reconstructed cluster core tensor is unfolded as above and then concatenated along the unfolded mode to form a third order tensor as the exemplar texture for ASTS. Therefore, ASTS actually compares the neighborhood similarity within each cluster. Since CTA partitions the view mode of $\boldsymbol{\mathcal{A}}^{(B)}$ into disjoint clusters, there is no need to consider mode-ω_v sub-tensors across the cluster boundaries. Nevertheless, recall that the decomposed clusters of K-CTA are overlapped. Thus for K-CTA, we also need to reconstruct the view mode, in addition to the x and y modes, of each cluster core tensor as

$$\forall c,\ \hat{\boldsymbol{\mathcal{Z}}}_c^{(B)} = \boldsymbol{\mathcal{Z}}_c^{(B)} \times_{\omega_v} \mathbf{U}_{\omega_v,c}^{(B)} \times_x \mathbf{U}_{x,c}^{(B)} \times_y \mathbf{U}_{y,c}^{(B)}, \tag{8.5}$$

where $\hat{\boldsymbol{\mathcal{Z}}}_c^{(B)} \in \mathbb{R}^{R_{\omega_l}^{(B)} \times I_{\omega_v}^{(B)} \times I_x^{(B)} \times I_y^{(B)}}$ represents the reconstructed core tensor of cluster c based on K-CTA.

Resampling: Due to the sparse sampling rates of the illumination and view modes, it is necessary to resample the compressed BTF data for efficient run-time rendering under novel illumination and view conditions. Although N-SVD allows us to generate observations from a novel illumination direction by resampling just the columns of the mode-ω_l basis matrix, resampling the view mode is complicated since CTA and K-CTA partition the view mode into different regions. For CTA, we employ traditional linear interpolation from nearby view directions to solve this issue, whereas the proposed K-CTA algorithm implicitly suggests a better resampling scheme.

For a compressed BTF based on K-CTA, we can resample the view mode of the raw BTF first and employ the greedy search and optimal projection in the clustering stage of K-CTA to compute the mixing coefficients of each mode-ω_v sub-tensor of the resampled BTF. Thus, the overall effect is equivalent to resampling the columns of the mode-ω_v basis matrices, so that the BTF reconstruction and interpolation are combined together without reducing run-time performance. Although one can continue to update the cluster sub-spaces of the compressed BTF and follow the process of K-CTA on the resampled BTF until convergence, we do not recommend applying this time-consuming iterative scheme. Moreover, to accelerate the resampling of the raw BTF using sophisticated methods, such as cubic or thin-plate spline interpolation in the spherical domain, we perform N-SVD on the BTF without approximation errors to obtain a full-rank mode-ω_v basis matrix of size $I_{\omega_v}^{(B)} \times I_{\omega_v}^{(B)}$, and then resample columns of this matrix instead.

Run-time rendering: The rendering process of compressed BTFs based on tensor approximation algorithms is rather straightforward. To increase performance and employ texture filtering on GPUs, we reconstruct the x and y modes of each (cluster) core tensor, create their mipmaps, and then concatenate the results into one or more two-dimensional texture arrays before rendering[1]. The mode-ω_l and mode-ω_v basis matrices are instead stored in two-dimensional textures in the parabolic parameterization [62]. The run-time process on GPUs thus consists of the following steps:

[1]If texture arrays are not supported in the graphics application programming interface, we may tile the mipmap of each (cluster) core tensor into one or more two-dimensional textures. However, one should be careful not to include texels out of the tile boundaries in the texture filtering. This can be achieved by clamping texture coordinates into an appropriate range before sampling.

Table 8.1: Statistics and timing measurements of different tensor approximation algorithms for BTFs. All decomposed data were stored as half-precision (16-bit) floating point numbers [65]. For the material *RoughHole*, we did not reduce its spatial x and y modes, hence no mode-x and mode-y basis matrices were extracted.

Material	Fiber			RoughHole			Sponge			Wool		
$I^{(B)}_{\omega_l} \times I^{(B)}_{\omega_v} \times I^{(B)}_{x} \times I^{(B)}_{y}$	100×100×96×96			100×100×96×96			120×90×128×128			81×81×128×128		
Raw data (GB)	1.03			1.03			1.98			1.2		
Algorithm	N-SVD	CTA	K-CTA	N-SVD	CTA	K-CTA	N-SVD	CTA	K-CTA	N-SVD	CTA	K-CTA
$R^{(B)}_{\omega_l} \times R^{(B)}_{\omega_v}$	16×80	16×4	16×4	28×44	28×4	28×4	16×20	16×4	16×4	20×28	20×4	20×4
$R^{(B)}_{x} \times R^{(B)}_{y}$	64×64	64×64	64×64	96×96	96×96	96×96	80×80	80×80	80×80	64×64	64×64	64×64
Clusters: $C^{(B)}$	1	20	20	1	11	11	1	5	5	1	7	7
Mixture clusters: $K^{(B)}_{\omega_v}$	1	1	3	1	1	3	1	1	3	1	1	3
Compressed data (MB)	10.05	10.48	10.48	21.68	21.67	21.67	3.96	4.11	4.11	4.42	4.6	4.6
Squared error ratio (%)	1.84	2.26	1.86	3.34	5.35	3.51	0.37	0.44	0.38	0.85	1.06	0.89
Compression time (min.)	11.23	21.87	42.37	3.15	12.48	15.25	20.58	26.03	28.73	13.72	12.53	18.75

1. In the pixel shader, sample the synthesized texture S for the BTF spatial coordinates $\mathbf{t_p}$ of current pixel $\mathbf{p}$.

2. Sample the texture(s) of core tensor(s) for all the data that correspond to $\mathbf{t_p}$, and the textures of mode-ω_l and mode-ω_v basis matrices for all the components of current novel illumination and view directions.

3. The shading color of pixel $\mathbf{p}$ is then given by reconstructing the illumination and view modes according to the tensor approximation algorithm.

For CTA and K-CTA, since mode-ω_v basis matrices are sparse, the reconstruction in Step 3 can be computed from only their non-zero entries. We achieve this by packing only the non-zero entries of mode-ω_v basis matrices in the corresponding texture, and constructing another two-dimensional texture to identify the cluster membership of each sampled/resampled view direction. For CTA, we additionally include the cluster membership of nearby view directions for linear interpolation, and thereby save the processing time of nearest neighbor searching. As for K-CTA, since the reconstruction with mixture clusters and mixing coefficients already resembles an interpolation process, we currently only utilize the nearest neighbor of the novel view direction for rendering. In our experience, nearest neighbor interpolation is usually unnecessary for K-CTA. The proposed resampling scheme effectively leads to smooth transitions when the viewpoint changes, at the cost of a dense resampling rate for view directions (typically 256×256 in the parabolic parameterization).

Experimental Results

The experiments and simulation timings of tensor approximation algorithms for BTF compression were conducted and measured on a workstation with an Intel Core 2 Extreme QX9650

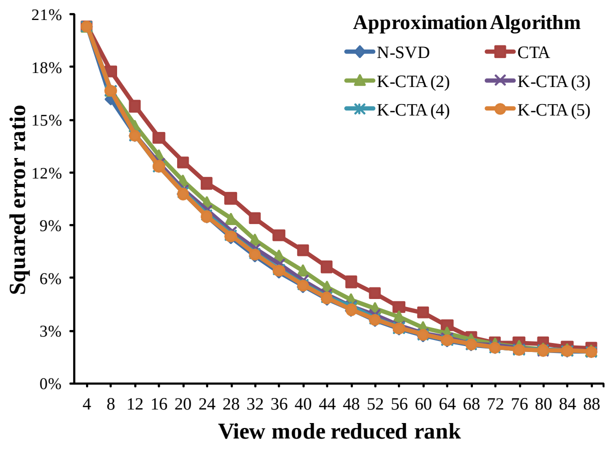
Table 8.2: Comparisons of the rendering performance of different tensor approximation algorithms for BTFs. The screen resolution and the number of directional light sources were respectively set to 640×480 and 2. Moreover, the resolution of each texture in the parabolic parameterization was set to 256×256. In the row *Coordinate texture*, we list the resolution of the synthesized coordinate texture of each BTF. Note that the statistics in the row *Total data* also include the amount of synthesized texture data. All floating point numbers were stored in half precision [65].

Model	Bunny			Cloth		
Material	Sponge			Wool		
Vertices	36k			30k		
Coordinate texture	2048×2048			1024×1024		
Algorithm	N-SVD	CTA	K-CTA	N-SVD	CTA	K-CTA
Total data (MB)	37.83	37.08	37.08	38.33	36.58	36.58
Frames per second	38.73	52.24	49.91	19.36	34.97	33.17

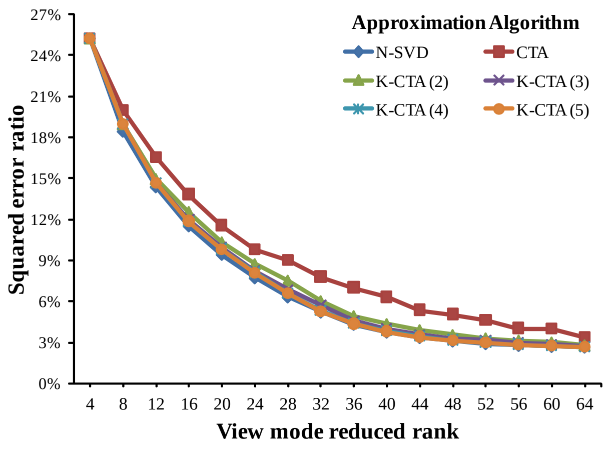
CPU, an NVIDIA GeForce GTX 280 graphics card, and 8 gigabytes main memory under Microsoft Windows Vista operating system. We employed Microsoft Direct3D 10 [15] application programming interface for run-time rendering on GPUs. The measured BTFs, including *Sponge* and *Wool*, were provided in courtesy of University of California at San Diego [81, 84] and University of Bonn [23, 153], whereas each simulated BTF, such as *Fiber* or *RoughHole*, was obtained by rendering a geometric surface using the photon mapping [69, 70] implementation from [137]. To improve quality, the meso-structure synthesis process was performed on CPUs to allow a higher-dimensional appearance-space exemplar (typically twenty-four-dimensional for a compressed BTF). In the experimental results of CTA and K-CTA, we adopted their iterative variants and especially set a single global mode-ω_l basis matrix of all clusters in Equation 8.2. This does not result in a perceptible loss of visual quality in our experience, but should lead to a more compact and efficient representation for run-time rendering on GPUs. In Chapter 9, we will further discuss the advantage of this configuration when using BTFs in bi-scale radiance transfer.

Table 8.1 compares the statistics and timing measurements of different tensor approximation algorithms, including N-SVD, CTA, and K-CTA. Figure 8.2 further plots the squared error ratio versus the mode-ω_v reduced rank for the three multi-linear models with various configurations. Note that we compare them based on the the same total number of mode-ω_v reduced rank, which leads to similar storage space. Moreover, while the run-time performance and rendered images with the compressed BTFs are shown in Table 8.2 and Figure 8.3, we also present the reconstructed BTFs in Figures 8.4–8.7.

In general, N-SVD provides the best perceptual quality among the three tensor approximation algorithms, but CTA and K-CTA allow much efficient rendering performance on GPUs for complex meso-structures. Although the compression time of K-CTA is slightly longer than N-SVD and CTA on average, K-CTA effectively compromises between image quality and run-time



(a) *Fiber*



(b) *RoughHole*

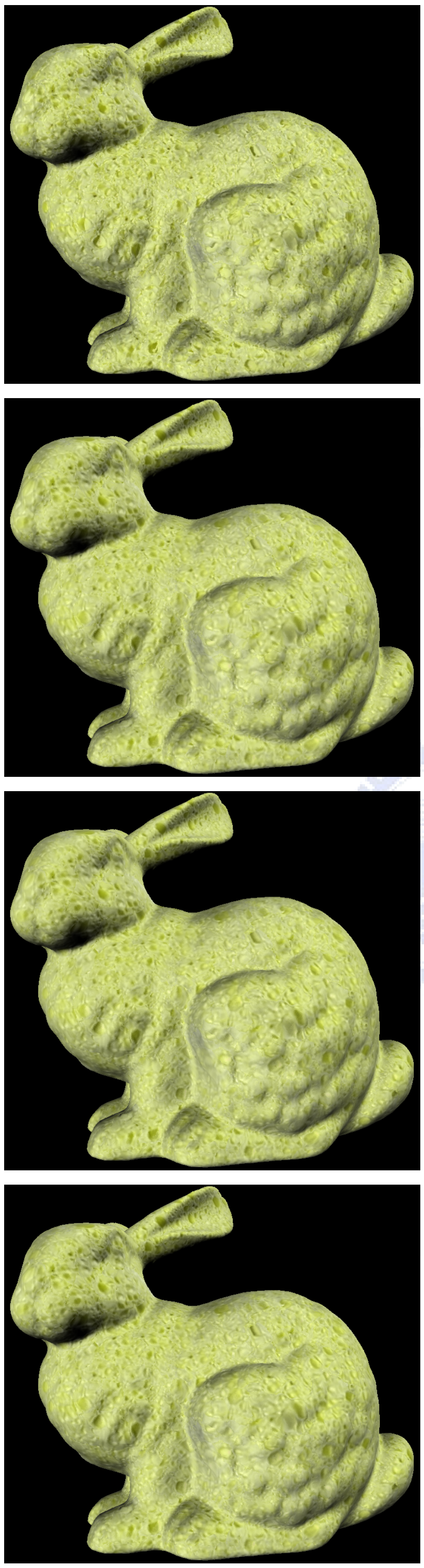
Figure 8.2: Plots of the squared error ratio versus the mode-ω_v reduced rank based on different tensor approximation algorithms for BTFs. Each value in parentheses for K-CTA specifies the number of mixture clusters. For CTA and K-CTA, the horizontal axis in each plot represents various total mode-ω_v reduced ranks whose values correspond to $R^{(B)}_{\omega_v} C^{(B)}$, with $R^{(B)}_{\omega_v} = 4$ for each cluster. For other configurations of tensor approximation algorithms, please refer to Table 8.1.

performance. Moreover, when the viewpoint changes at run-time, CTA with linear interpolation from the three nearest view directions sometimes produces noticeable seams. This is due to the intrinsic nature of CTA in which the inter-cluster coherence is ignored, so that significant discontinuities may occur at cluster boundaries, especially when inappropriate clustered results were found. A dense sampling rate for the view mode of raw BTFs or increasing the mode-ω_v reduced rank would solve this issue for CTA, but both of them will increase the amount of compressed data and rendering time. The acquisition of densely sampled BTFs is also a challenging problem. By contrast, K-CTA is less sensitive to the quality of clustering since approximation errors are compensated by additional mixture clusters. This compensation can be regarded as an optimal interpolation scheme in the least-squares sense to enable smooth transitions across different view directions.

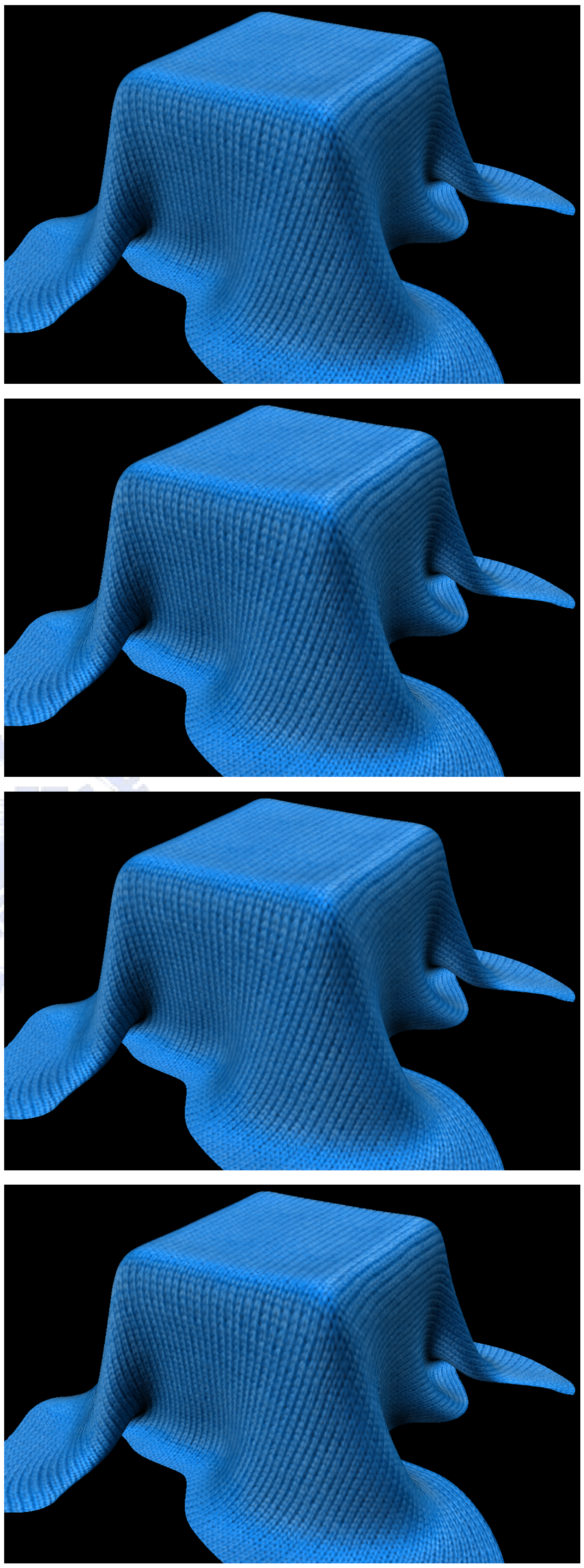
8.1.2 Multivariate SRBFs

Problem Formulation

Besides tensor approximation algorithms, we may also apply multivariate SRBFs to approximate a BTF $B(\omega_l, \omega_v, \mathbf{t})$. The key observation is that the appearance data of a BTF texel can be viewed as a BRDF. It is then intuitive to model a BTF as a set of BRDFs, where each BRDF is approximated based on the parameterized multivariate SRBF representation by using Algorithm 4.3 (Section 4.3). Nevertheless, such a brute-force approach may result in a computationally intractable problem even if the optimization process is accelerated by GPUs. Moreover, the spatial coherence in different BTF texels also can not be found with this approach.

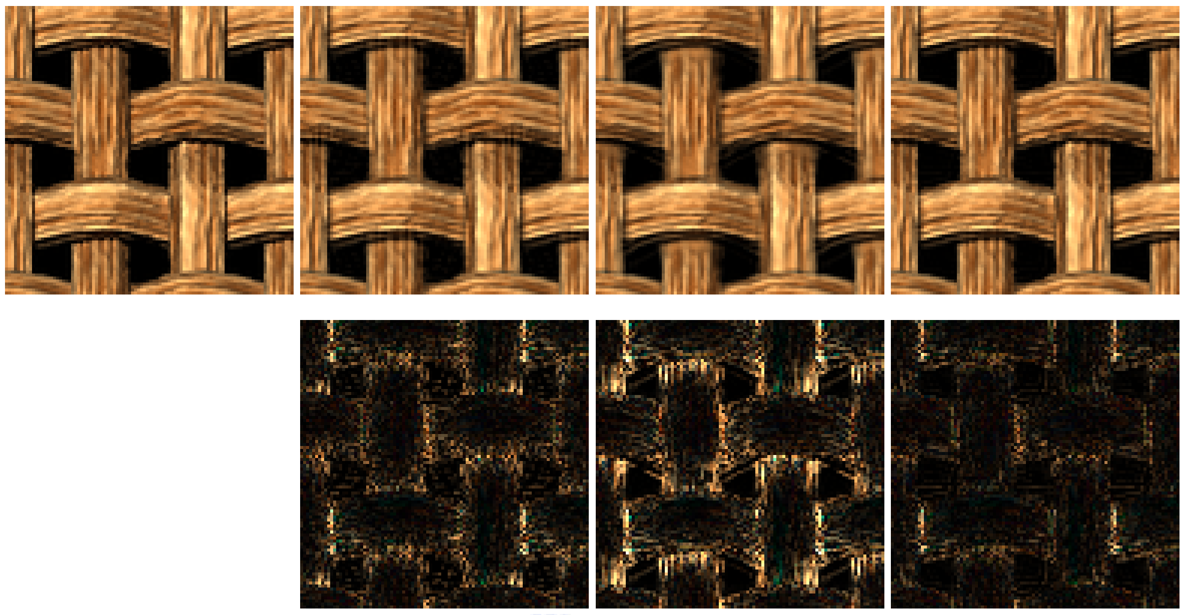


(a) *Bunny* with *Sponge*

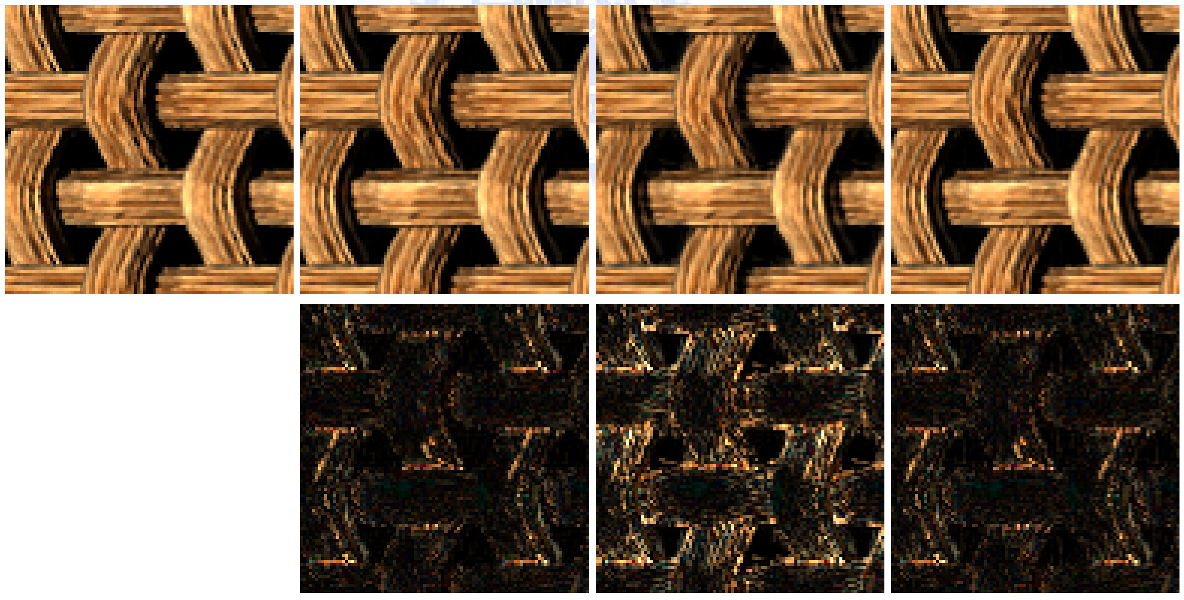


(b) *Cloth* with *Wool*

Figure 8.3: Rendered images based on different tensor approximation algorithms for BTFs. From top to bottom: raw data; N-SVD; CTA; K-CTA. For the configurations of tensor approximation algorithms and run-time rendering, please refer to Tables 8.1 and 8.2.

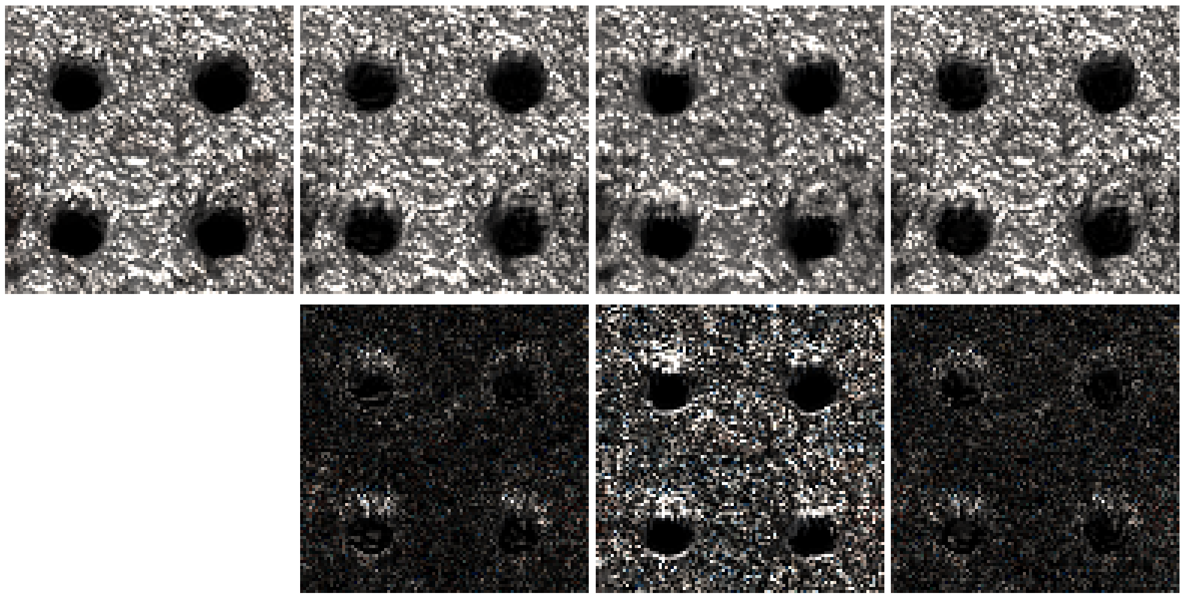


(a) Illumination direction – zenith angle: 9°, azimuth angle: 189°; view direction – zenith angle: 27°, azimuth angle: 279°

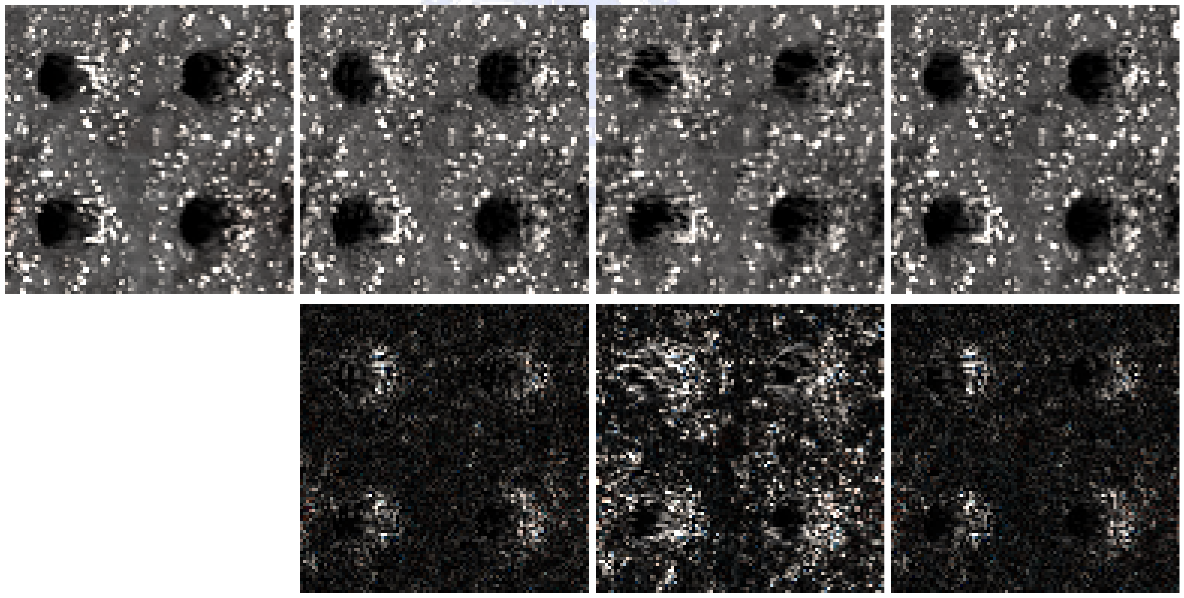


(b) Illumination direction – zenith angle: 9°, azimuth angle: 189°; view direction – zenith angle: 45°, azimuth angle: 9°

Figure 8.4: Reconstructed images of the BTF for the material *Fiber* based on different tensor approximation algorithms. From left to right: raw data; N-SVD; CTA; K-CTA. From top to bottom in each row: reconstructed images; absolute difference images scaled by a factor of 5.

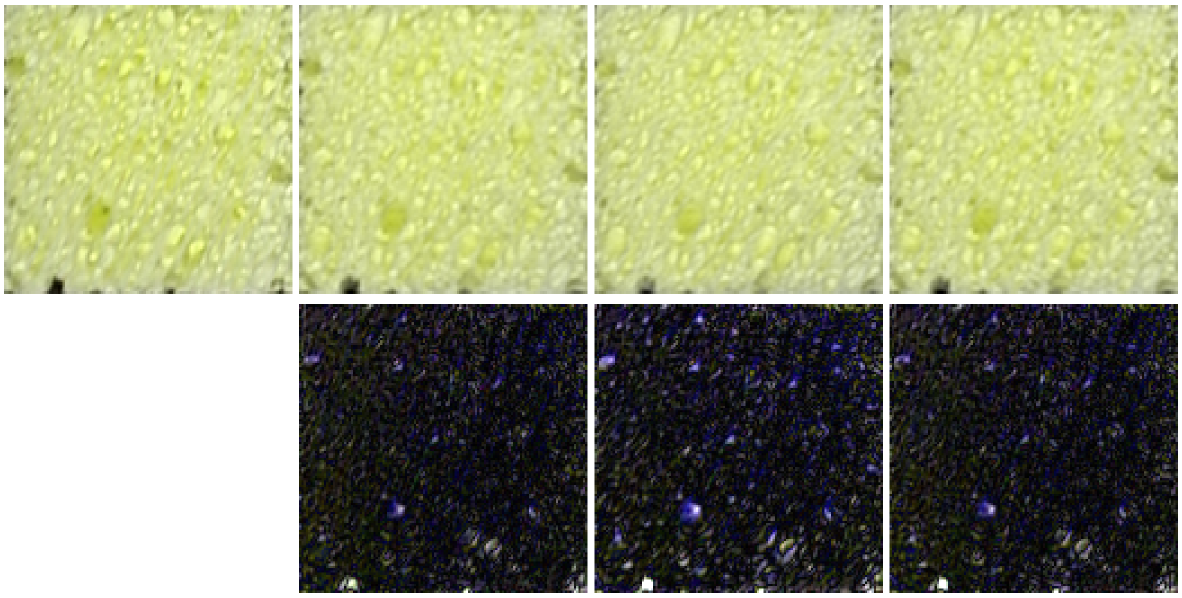


(a) Illumination direction – zenith angle: 9°, azimuth angle: 189°; view direction – zenith angle: 45°, azimuth angle: 279°



(b) Illumination direction – zenith angle: 9°, azimuth angle: 189°; view direction – zenith angle: 63°, azimuth angle: 189°

Figure 8.5: Reconstructed images of the BTF for the material *RoughHole* based on different tensor approximation algorithms. From left to right: raw data; N-SVD; CTA; K-CTA. From top to bottom in each row: reconstructed images; absolute difference images scaled by a factor of 3.


(a) Illumination direction – zenith angle: 15°, azimuth angle: 75°; view direction – zenith angle: 60°, azimuth angle: 70°

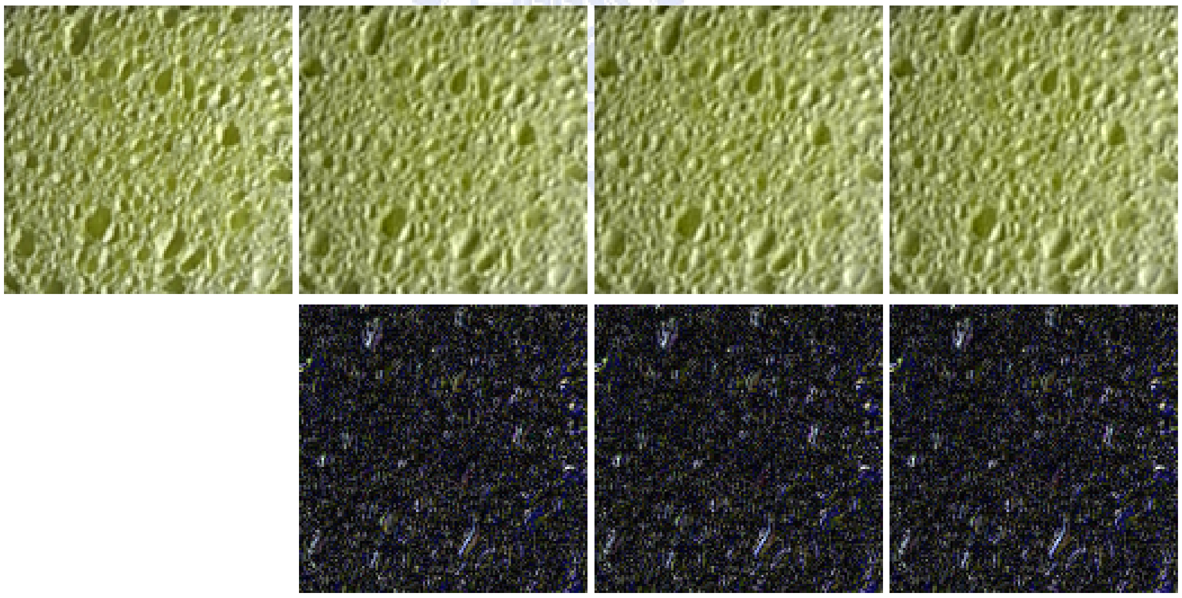
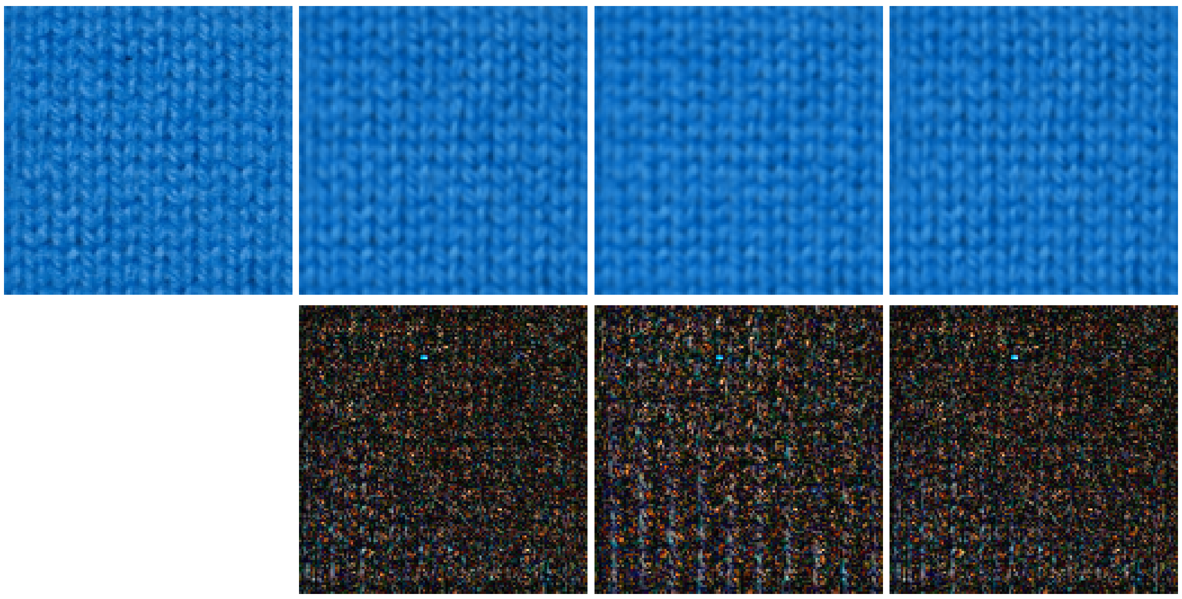

(b) Illumination direction – zenith angle: 60°, azimuth angle: 345°; view direction – zenith angle: 0°, azimuth angle: 70°

Figure 8.6: Reconstructed images of the BTF for the material *Sponge* based on different tensor approximation algorithms. From left to right: raw data; N-SVD; CTA; K-CTA. From top to bottom in each row: reconstructed images; absolute difference images scaled by a factor of 4.


(a) Illumination direction – zenith angle: 0°, azimuth angle: 0°; view direction – zenith angle: 45°, azimuth angle: 0°

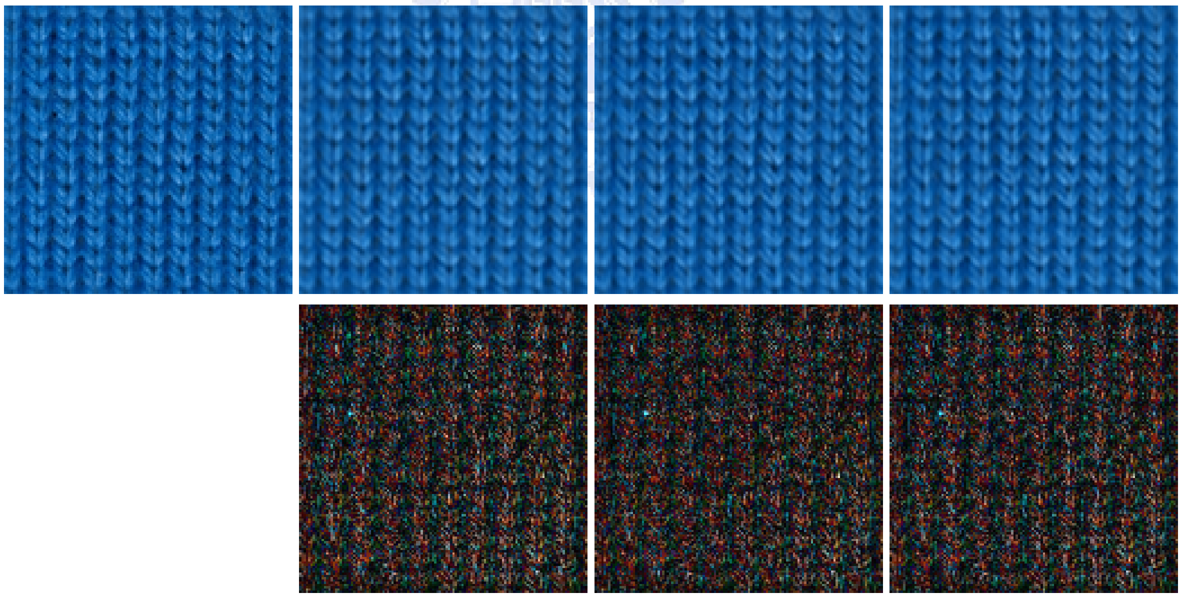

(b) Illumination direction – zenith angle: 45°, azimuth angle: 240°; view direction – zenith angle: 0°, azimuth angle: 0°

Figure 8.7: Reconstructed images of the BTF for the material *Wool* based on different tensor approximation algorithms. From left to right: raw data; N-SVD; CTA; K-CTA. From top to bottom in each row: reconstructed images; absolute difference images scaled by a factor of 6.

Fortunately, it is reasonable to assume that if the observations of nearby BTF texels have strong correlations, their derived SRBF parameters are also expected to be highly correlated. Based on this assumption, although multivariate SRBFs only account for the directional variables of a BTF, its spatial correlations can be exploited using multi-resolution analysis. We thus introduced a hierarchical fitting algorithm in Section 4.4 to solve this issue.

More formally, consider a hierarchical set of multi-resolution BTFs $\{B_i(\omega_l, \omega_v, \mathbf{t})\}_{i=0}^{I_h}$, where I_h is the number of levels, and the BTF at level i is denoted by $B_i(\omega_l, \omega_v, \mathbf{t})$. Suppose that BTFs in this set are ordered from the coarsest level 0 to the finest level I_h, while each of them is generated by spatially downsampling $B(\omega_l, \omega_v, \mathbf{t})$ by a factor of a user-specified constant. Our goal is to model each texel of the BTFs in the hierarchical set using the parameterized multivariate SRBF representation as

$$\begin{aligned}\forall i,\ \forall \mathbf{t},\ B_i(\omega_l, \omega_v, \mathbf{t}) &\approx \hat{B}'_i\big(\Psi^{(B_i)}(\mathbf{t})\big)\\ &= \sum_{j=1}^{J} \beta_j^{(B_i)}(\mathbf{t}) \prod_{n=1}^{N'} G\Big(\psi_n^{(B_i)}\big(\omega_l, \omega_v | \Theta_n^{(B_i)}(\mathbf{t})\big) \cdot \xi_{j,n}^{(B_i)}(\mathbf{t}) | \lambda_{j,n}^{(B_i)}(\mathbf{t})\Big),\end{aligned} \tag{8.6}$$

where $\Psi^{(B_i)}(\mathbf{t}) = \{\psi_n^{(B_i)}\big(\omega_l, \omega_v | \Theta_n^{(B_i)}(\mathbf{t})\big) \in \mathbb{S}^2\}_{n=1}^{N'}$ denotes the transformation function set for a texel $\mathbf{t}$ of the BTF at level i, $\{\beta_j^{(B_i)}(\mathbf{t}) \in \mathbb{R}\}_{j=1}^{J}$ similarly specifies its basis coefficient set, and $\{\xi_{j,n}^{(B_i)}(\mathbf{t}) \in \mathbb{S}^2\}_{j=1}^{J}$, $\{\lambda_{j,n}^{(B_i)}(\mathbf{t}) \in \mathbb{R}\}_{j=1}^{J}$, as well as $\Theta_n^{(B_i)}(\mathbf{t})$ respectively represent the center set, the bandwidth set, and the parameterization coefficient set for the n-th transformed variable. Note that each SRBF parameter set now depends on a BTF texel $\mathbf{t}$ at level i.

We then apply Algorithm 4.4 to learn the SRBF parameters in Equation 8.6 with a top-down approach (from the coarsest level to the finest one), while the parameters of a BTF texel $\mathbf{t}$ at level i are derived by solving the following unconstrained least-squares optimization problem:

$$\forall i,\ \forall \mathbf{t},\ \min_{\Upsilon^{(B_i)}(\mathbf{t})} \frac{1}{2} \int_{\mathbb{S}^2} \int_{\mathbb{S}^2} \Big(B_i(\omega_l, \omega_v, \mathbf{t}) - \hat{B}'_i\big(\Psi^{(B_i)}(\mathbf{t})\big)\Big)^2 d\omega_l d\omega_v + E_{addl}^{(B_i)}(\mathbf{t}), \tag{8.7}$$

where $\Upsilon^{(B_i)}(\mathbf{t}) = \Big\{\{\beta_j^{(B_i)}(\mathbf{t}), \Xi_j^{(B_i)}(\mathbf{t}), \Lambda_j^{(B_i)}(\mathbf{t})\}_{j=1}^{J}, \{\Theta_n^{(B_i)}(\mathbf{t})\}_{n=1}^{N'}\Big\}$ is the set of all SRBF parameters of texel $\mathbf{t}$ at level i for $\Xi_j^{(B_i)}(\mathbf{t}) = \{\xi_{j,n}^{(B_i)}(\mathbf{t})\}_{n=1}^{N'}$ and $\Lambda_j^{(B_i)}(\mathbf{t}) = \{\lambda_{j,n}^{(B_i)}(\mathbf{t})\}_{n=1}^{N'}$. To learn SRBF parameters with spatial correlations, the additional energy term $E_{addl}^{(B_i)}(\mathbf{t})$ of BTF texel $\mathbf{t}$ at level i controls the smoothness of neighboring BTF texels and is defined as follows:

$$E_{addl}^{(B_i)}(\mathbf{t}) = \mu_\beta^{(B_i)} E_\beta^{(B_i)}(\mathbf{t}) + \mu_\xi^{(B_i)} E_\xi^{(B_i)}(\mathbf{t}) + \mu_\lambda^{(B_i)} E_\lambda^{(B_i)}(\mathbf{t}) + \mu_{\xi,\lambda}^{(B_i)} E_{\xi,\lambda}^{(B_i)}(\mathbf{t}) + \mu_\theta^{(B_i)} E_\theta^{(B_i)}(\mathbf{t}), \tag{8.8}$$

$$E_\beta^{(B_i)}(\mathbf{t}) = \sum_{\mathbf{t}' \in \mathcal{N}'_i(\mathbf{t})} \sum_{j=1}^{J} \frac{1}{2} \big(\beta_j^{(B_i)}(\mathbf{t}) - \beta_j^{(B_i)}(\mathbf{t}')\big)^2, \tag{8.9}$$

$$E_{\xi}^{(B_i)}(\mathbf{t}) = \sum_{\mathbf{t}' \in \mathcal{N}_i'(\mathbf{t})} \sum_{j=1}^{J} \sum_{n=1}^{N} \left(1 - \xi_{j,n}^{(B_i)}(\mathbf{t}) \cdot \xi_{j,n}^{(B_i)}(\mathbf{t}')\right), \tag{8.10}$$

$$E_{\lambda}^{(B_i)}(\mathbf{t}) = \sum_{\mathbf{t}' \in \mathcal{N}_i'(\mathbf{t})} \sum_{j=1}^{J} \sum_{n=1}^{N} \frac{1}{2} \left(\lambda_{j,n}^{(B_i)}(\mathbf{t}) - \lambda_{j,n}^{(B_i)}(\mathbf{t}')\right)^2, \tag{8.11}$$

$$E_{\xi,\lambda}^{(B_i)}(\mathbf{t}) = \sum_{\mathbf{t}' \in \mathcal{N}_i'(\mathbf{t})} \sum_{j=1}^{J} \sum_{n=1}^{N} \frac{1}{2} \left\| \lambda_{j,n}^{(B_i)}(\mathbf{t}) \xi_{j,n}^{(B_i)}(\mathbf{t}) - \lambda_{j,n}^{(B_i)}(\mathbf{t}') \xi_{j,n}^{(B_i)}(\mathbf{t}') \right\|_2^2, \tag{8.12}$$

$$E_{\theta}^{(B_i)}(\mathbf{t}) = \sum_{\mathbf{t}' \in \mathcal{N}_i'(\mathbf{t})} \sum_{n=1}^{N'} \sum_{j=1}^{I_{\Theta_n}} \frac{1}{2} \left(\theta_{j,n}^{(B_i)}(\mathbf{t}) - \theta_{j,n}^{(B_i)}(\mathbf{t}')\right)^2, \tag{8.13}$$

where $\mathcal{N}_i'(\mathbf{t})$ denotes the set of participating BTF texels at levels i and $i-1$ for texel $\mathbf{t}$ at level i, and $\mu_{\beta}^{(B_i)}$, $\mu_{\xi}^{(B_i)}$, $\mu_{\lambda}^{(B_i)}$, $\mu_{\xi,\lambda}^{(B_i)}$, as well as $\mu_{\theta}^{(B_i)}$ are respectively the user-defined weights for $E_{\beta}^{(B_i)}(\mathbf{t})$, $E_{\xi}^{(B_i)}(\mathbf{t})$, $E_{\lambda}^{(B_i)}(\mathbf{t})$, $E_{\xi,\lambda}^{(B_i)}(\mathbf{t})$, and $E_{\theta}^{(B_i)}(\mathbf{t})$. Note that the smoothness energy term $E_{\xi,\lambda}^{(B_i)}(\mathbf{t})$ is employed only for multivariate Abel-Poisson and Gaussian SRBFs.

Run-Time Rendering

The rendering process of approximated BTFs based on multivariate SRBFs is similar to BRDF approximation (Section 7.2.2). To enable mipmap texture filtering on GPUs, we concatenate the SRBF parameters at each level into several two-dimensional texture arrays, with one (or more if necessary) texture array for one category of SRBF parameter sets[2]. For meso-structure synthesis, we apply the proposed method in Section 8.1.1 to obtain the spatial coordinate texture S. Note that this approach needs to additionally compress the same BTFs using tensor approximation algorithms. Performing meso-structure synthesis directly on the derived SRBF parameters is left as a research direction in the future.

The rendering process thus consists of the following steps:

1. For current pixel $\mathbf{p}$, sample the synthesized texture S for the BTF spatial coordinates $\mathbf{t}_{\mathbf{p}}$.
2. Sample the texture(s) for all the SRBF parameters that correspond to $\mathbf{t}_{\mathbf{p}}$.
3. The shading color of pixel $\mathbf{p}$ is then given by performing the reconstruction according to the adopted multivariate SRBF representation.

For a pixel, note that we do not reconstruct the shading color of each participating texel and then perform mipmap filtering, but instead filter the SRBF parameters of each participating texel first and reconstruct the final shading color. When the derived SRBF parameters of each BTF texel are smooth enough, this approach usually increases the rendering performance by a factor of about 6 for trilinear mipmap filtering without noticeable artifacts. In general, the

[2]One may pack the SRBF center and bandwidth sets into one two-dimensional texture array to slightly reduce texture access time.

Table 8.3: Statistics and timing measurements of the parameterized multivariate SRBF representation for BTFs. In this table, we list the results of two different parameterization methods, including *fixed* (Fix.) and *optimized* (Opt.) parameterization. The statistics and timing measurements of a BTF were recorded with respect to the whole BTF data in a hierarchical set, and all compressed data were stored as half-precision (16-bit) floating point numbers [65].

Material	Carpet		Impalla		Sponge		Wool	
Illumination directions	120		81		120		81	
View directions	90		81		90		81	
Spatial resolution	128×128		128×128		128×128		128×128	
Raw data (GB)	2.64		1.6		2.64		1.6	
Parameterization	Fix.	Opt.	Fix.	Opt.	Fix.	Opt.	Fix.	Opt.
SRBFs: J	16	12	12	12	8	8	12	12
Parameterized variates: N'	3	4	3	3	3	3	3	3
Compressed data (MB)	8.0	7.83	6.0	6.25	4.0	4.25	6.0	6.25
Squared error ratio (%)	4.01	3.62	3.18	2.73	0.86	0.81	1.77	1.76
Compression time (hr.)	5.81	13.96	5.13	8.09	3.32	5.21	7.01	12.15

performance gain strongly depends on the utilized filtering technique. If the more sophisticated the filtering technique is, the more performance gain this approach can provide.

Experimental Results

The experiments and simulation timings of BTF approximation based on the proposed parameterized multivariate SRBF representation were conducted and measured on a workstation with an Intel Core 2 Extreme QX9650 CPU, an NVIDIA GeForce GTX 280 graphics card, and 8 gigabytes main memory under Microsoft Windows Vista operating system. We also employed NVIDIA CUDA [128] to accelerate parts of the optimization process and Microsoft Direct3D 10 [15] for run-time rendering on GPUs. The multivariate Gaussian SRBFs were adopted to represent BTFs, and the SRBF center as well as bandwidth sets were additionally constrained to be the same for red, green, and blue channels of a BTF texel. The measured BTFs were provided in courtesy of University of California at San Diego [81, 84] and University of Bonn [23, 153].

Table 8.3 compares the statistics and timing measurements of the parameterized multivariate SRBF representation for multi-resolution BTF data sets, which includes the experimental results of traditional fixed parameterization and optimized parameterization (Section 4.3). In our experiments, at most three traditional fixed parameterizations: half-way, illumination, and view directions were employed. As for optimized parameterization, we instead utilized the paramterization function defined in Equation 4.11 and set the initial guess of the first three parameterization coefficient sets as the half-way, illumination, and view parameterizations, while the remainders were initialized to random real numbers. To account for the randomness of the

Table 8.4: Comparisons of the rendering performance of the parameterized multivariate SRBF representation for BTFs. The performance of two different parameterization methods are shown in this table, including *fixed* (Fix.) and *optimized* (Opt.) parameterization. The screen resolution and the number of directional light sources were respectively set to 640×480 and 2. In the row *Coordinate texture*, we list the resolution of the synthesized coordinate texture of each BTF. Note that the statistics in the row *Total data* also include the amount of synthesized texture data. All floating point numbers were stored in half precision [65].

Model	Bunny		Cloth	
Material	Sponge		Wool	
Vertices	36k		30k	
Coordinate texture	2048×2048		1024×1024	
Parameterization	Fix.	Opt.	Fix.	Opt.
Total data (MB)	20.0	20.25	10.0	10.25
Frames per second	294.79	269.63	122.18	116.42

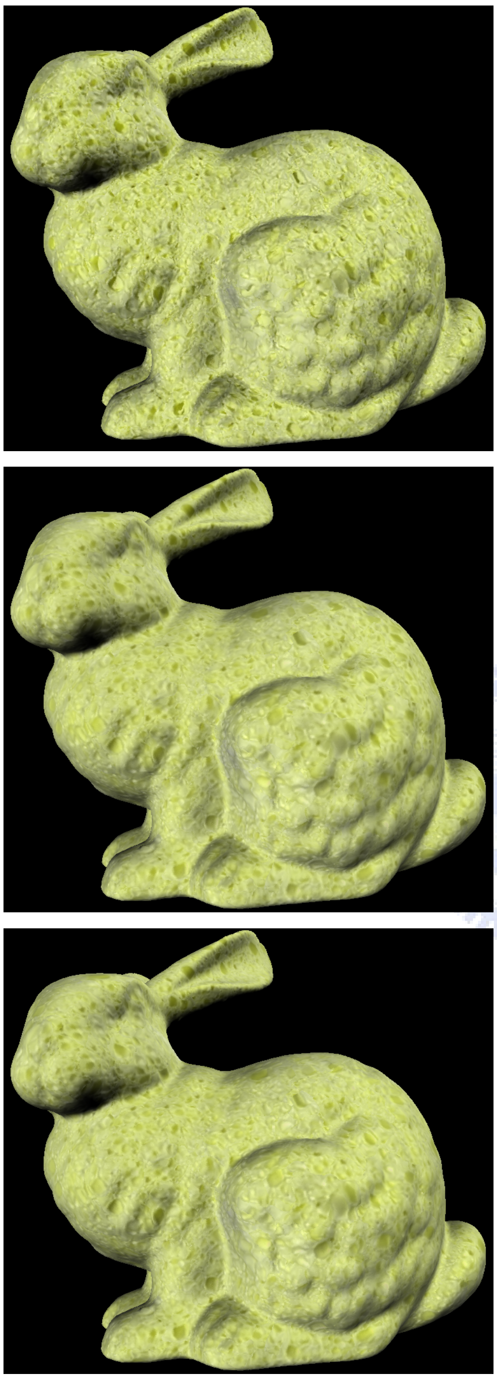
initial guess, the optimization stage at the coarsest level 0 was performed multiple times if the number of parameterization coefficient sets was more than 3, namely $N' > 3$. Then, only the SRBF parameters with the lowest approximation errors for BTF texels at level 0 were employed in the subsequent upsampling and optimization stages at higher levels.

While the run-time performance and rendered images based on the parameterized multivariate SRBF representation are respectively presented in Table 8.4 and Figure 8.8, we also demonstrate the reconstructed BTFs in Figures 8.9–8.12. The proposed optimized parameterization generally outperforms the traditional fixed approach in terms of approximation errors and visual quality, especially when the BTF data sets exhibit complex meso-structures, specular reflectance, or sharp shadows. Due to the access overhead of additional parameters, the rendering performance of optimized parameterization is slightly slower than that of fixed parameterization, but there are no significant differences between them. For a medium-size model, both methods can easily achieve real-time rendering rates at run-time.

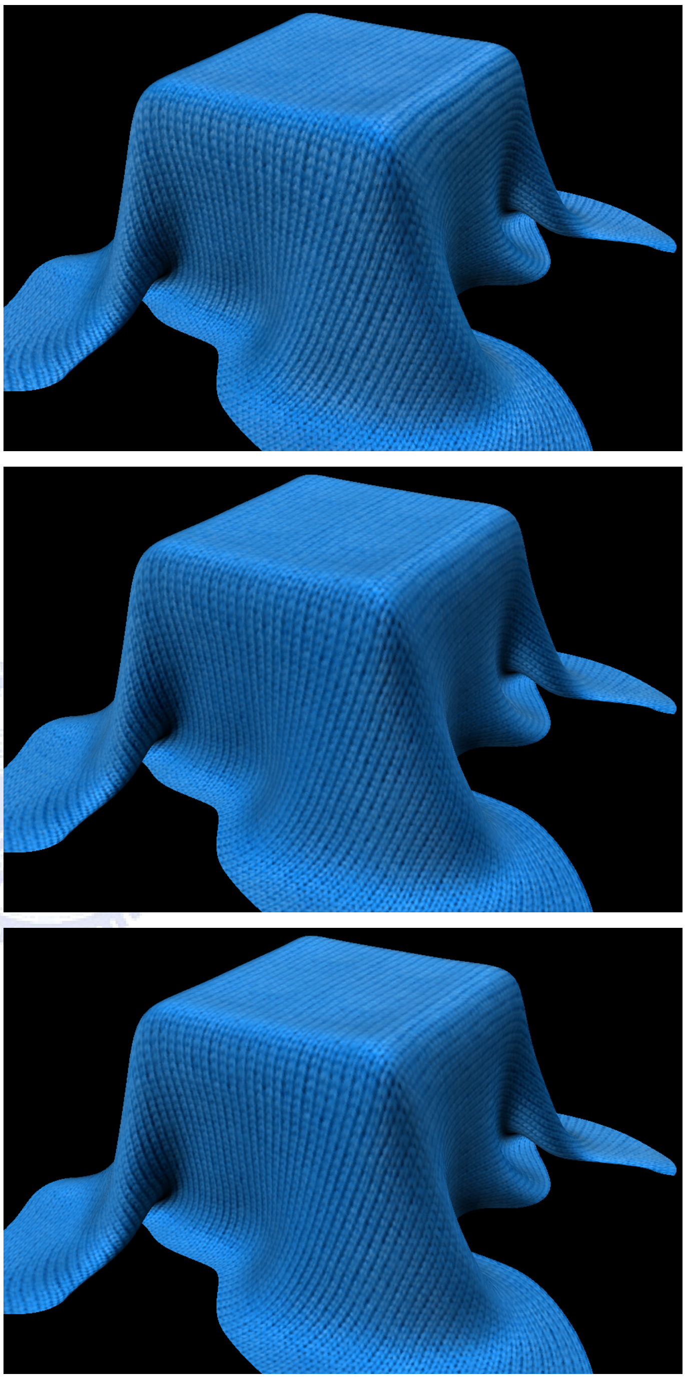
8.1.3 Comparisons and Discussions

In Sections 8.1.1 and 8.1.2, we introduced two categories of powerful compression methods for BTFs, including tensor approximation algorithms and multivariate SRBFs. Nevertheless, the decision on which one to use for a specific real-time application is still a non-trivial matter. In the following context, we will compare these two categories and discuss their advantages and disadvantages in details.

Table 8.5 summarizes the feature comparisons between tensor approximation algorithms and multivariate SRBFs for BTFs. Comparisons between these two categories of compression methods are similar to the traditional debates between parametric and non-parametric models in the statistics and machine learning communities. In general, tensor approximation algorithms

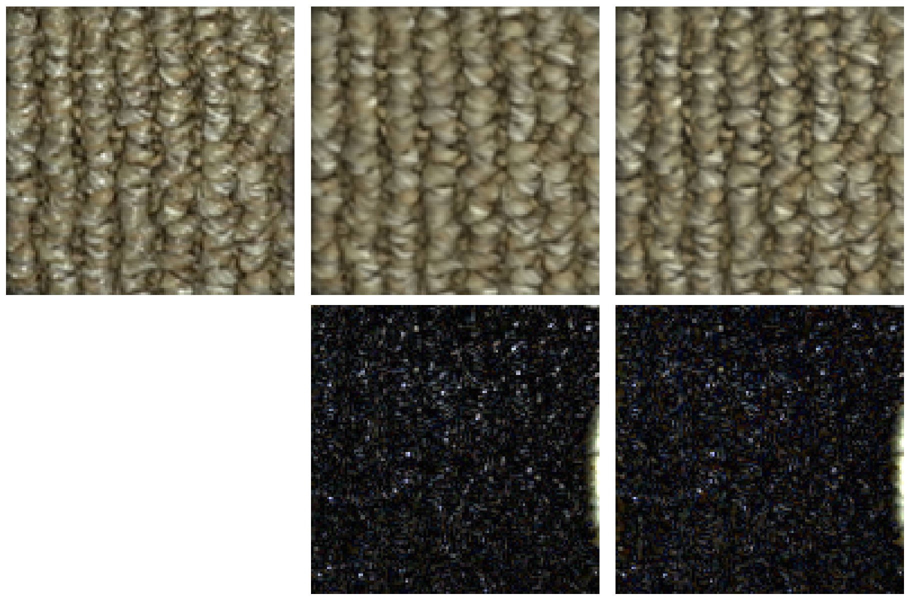


(a) *Bunny* with *Sponge*

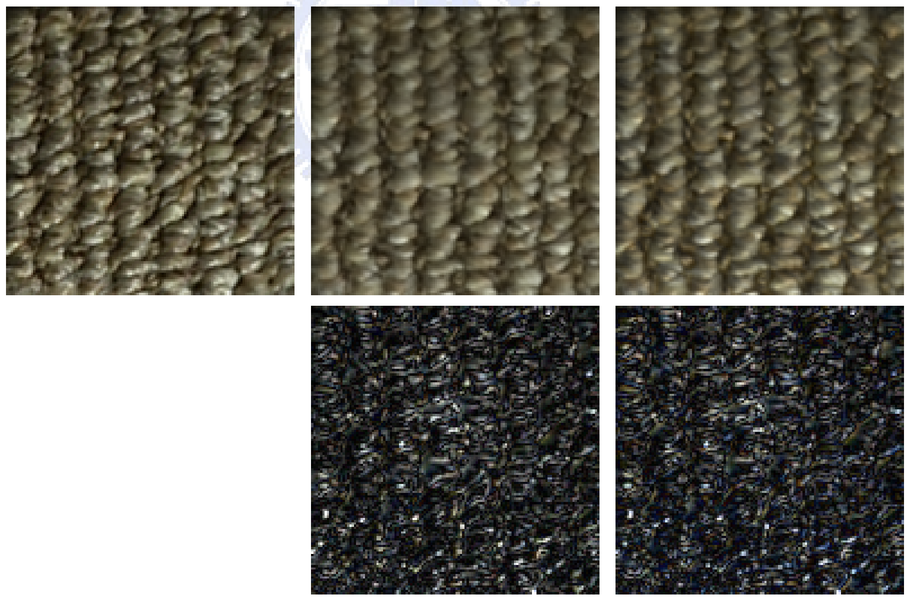


(b) *Cloth* with *Wool*

Figure 8.8: Rendered images based on the parameterized multivariate SRBF representation for BTFs. From top to bottom: raw data; fixed parameterization; optimized parameterization. For the configurations of parameterizations and run-time rendering, please refer to Tables 8.3 and 8.4.

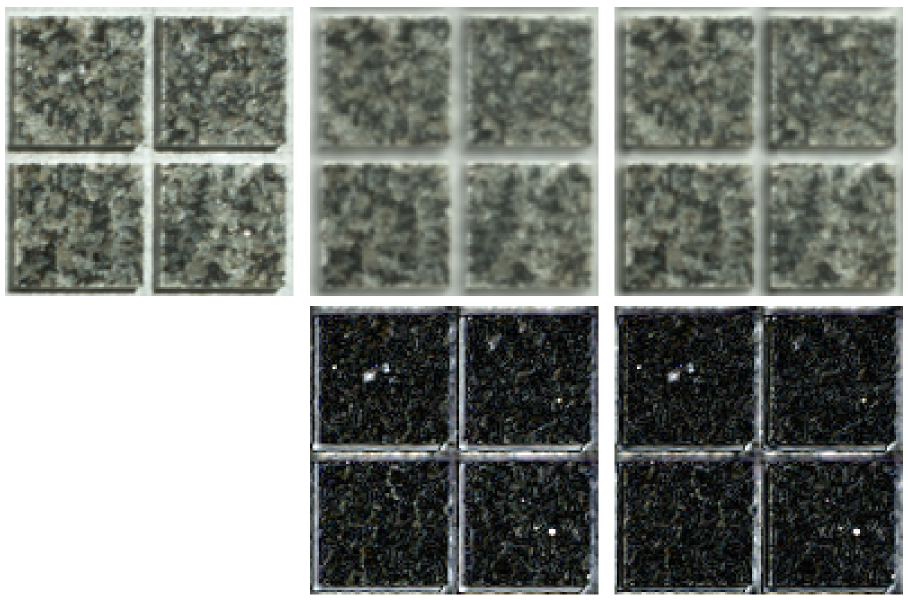


(a) Illumination direction – zenith angle: 15°, azimuth angle: 15°; view direction – zenith angle: 0°, azimuth angle: 10°

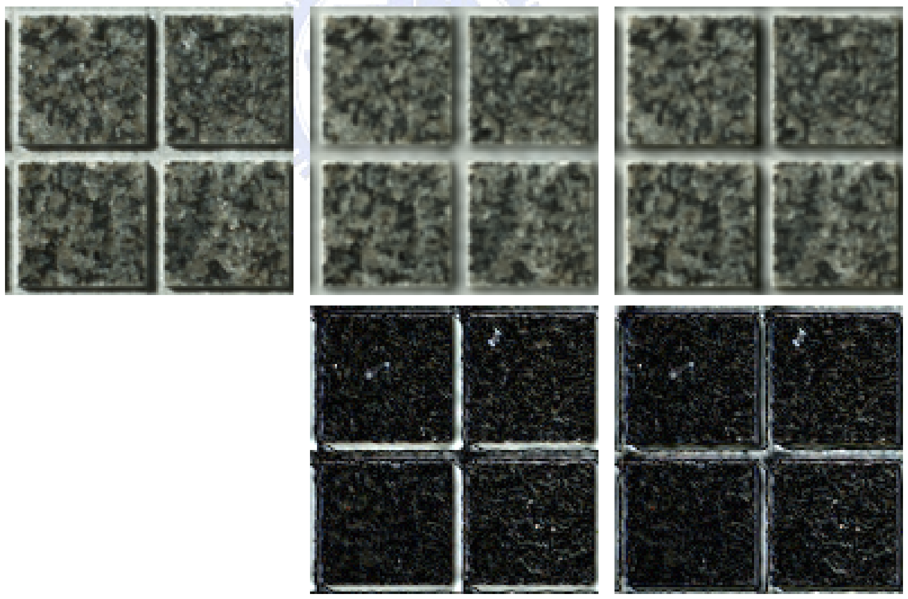


(b) Illumination direction – zenith angle: 60°, azimuth angle: 285°; view direction – zenith angle: 40°, azimuth angle: 230°

Figure 8.9: Reconstructed images of the BTF for the material *Carpet* based on the parameterized multivariate SRBF representation. From left to right: raw data; fixed parameterization; optimized parameterization. From top to bottom in each row: reconstructed images; absolute difference images scaled by a factor of 3.

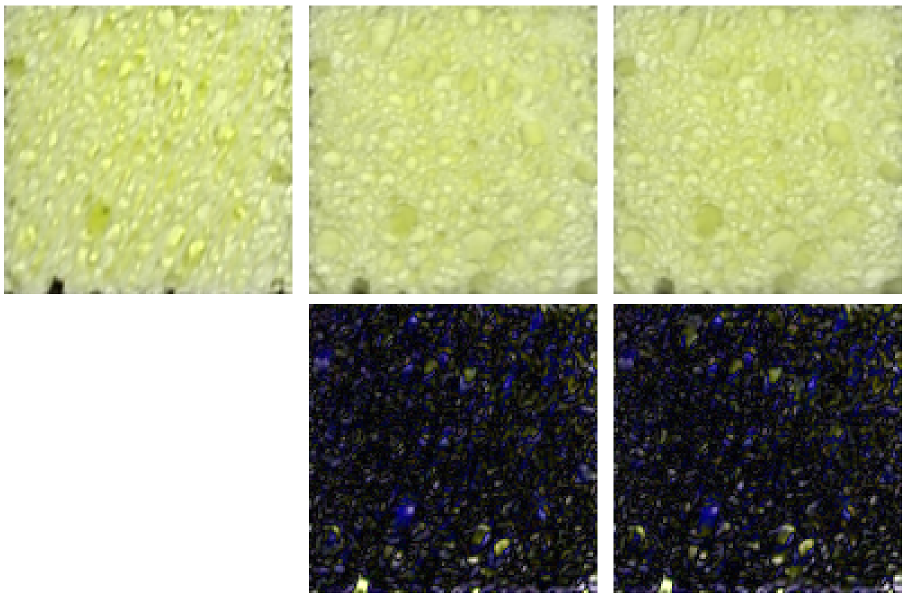


(a) Illumination direction – zenith angle: 45°, azimuth angle: 80°; view direction – zenith angle: 45°, azimuth angle: 180°



(b) Illumination direction – zenith angle: 60°, azimuth angle: 144°; view direction – zenith angle: 0°, azimuth angle: 0°

Figure 8.10: Reconstructed images of the BTF for the material *Impalla* based on the parameterized multivariate SRBF representation. From left to right: raw data; fixed parameterization; optimized parameterization. From top to bottom in each row: reconstructed images; absolute difference images scaled by a factor of 3.


(a) Illumination direction – zenith angle: 15°, azimuth angle: 75°; view direction – zenith angle: 60°, azimuth angle: 70°

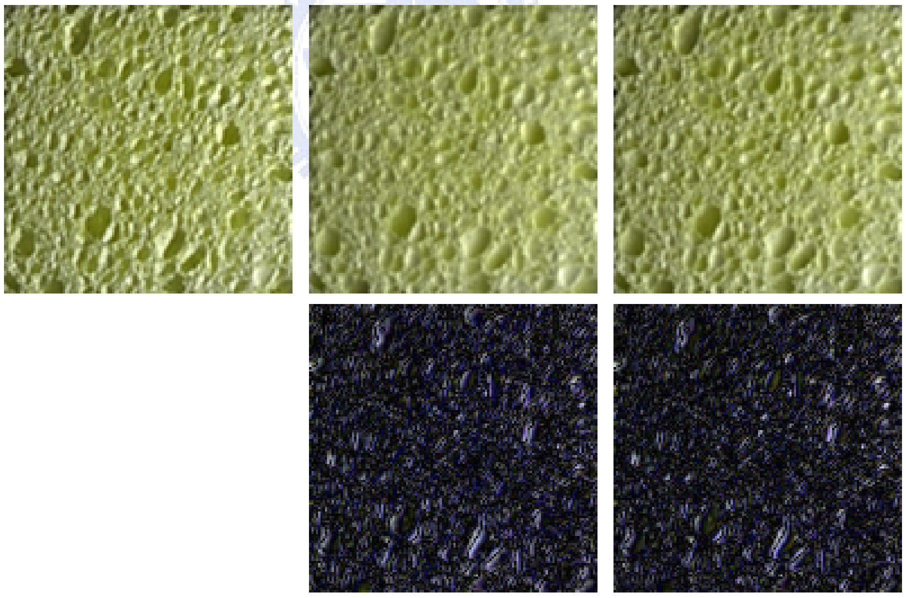
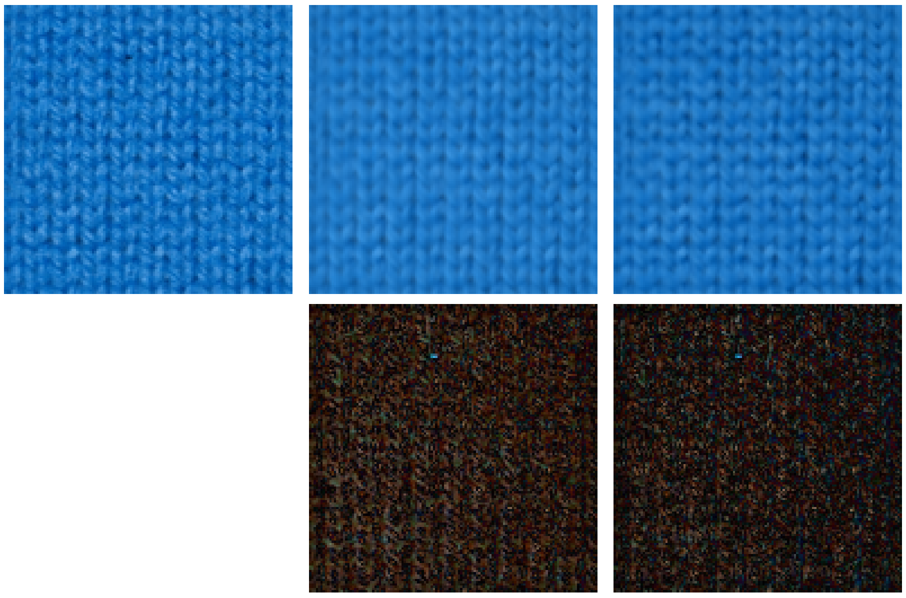
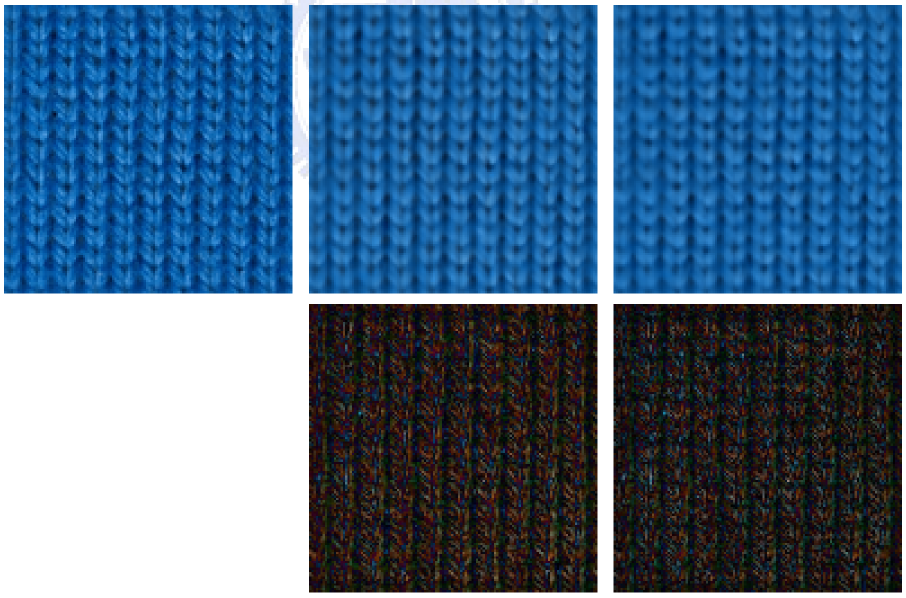

(b) Illumination direction – zenith angle: 60°, azimuth angle: 345°; view direction – zenith angle: 0°, azimuth angle: 70°

Figure 8.11: Reconstructed images of the BTF for the material *Sponge* based on the parameterized multivariate SRBF representation. From left to right: raw data; fixed parameterization; optimized parameterization. From top to bottom in each row: reconstructed images; absolute difference images scaled by a factor of 3.



(a) Illumination direction – zenith angle: 0°, azimuth angle: 0°; view direction – zenith angle: 45°, azimuth angle: 0°



(b) Illumination direction – zenith angle: 45°, azimuth angle: 240°; view direction – zenith angle: 0°, azimuth angle: 0°

Figure 8.12: Reconstructed images of the BTF for the material *Wool* based on the parameterized multivariate SRBF representation. From left to right: raw data; fixed parameterization; optimized parameterization. From top to bottom in each row: reconstructed images; absolute difference images scaled by a factor of 3.

Table 8.5: Feature comparisons between tensor approximation algorithms and multivariate SRBFs for BTFs.

Method	Multivariate SRBFs	Tensor Approximation
Assumption	Parametric	Non-parametric
Compression error	Moderate	Low
Compression ratio	Moderate*	High
Compression time	Time-consuming	Moderate
Formulation	Continuous	Discrete
Missing data	Yes	Yes†
Auxiliary data for rendering	No	Yes
Rendering performance	Fast	Moderate

*Recall that multivariate SRBFs can only handle directional variables, but tensor approximation algorithms can reduce the dimensionality of all kinds of variables at the same time. We currently do not apply any other approximation methods, such as wavelets, matrix factorization, or even tensor approximation, to compress the spatial domain of SRBF parameters for a BTF. If these methods were additionally employed, the resulting compression ratio is expected to be higher than merely tensor approximation algorithms. However, compression errors and rendering costs would increase instead.

†One of the missing data handling methods for multi-linear models can be found in [189].

(non-parametric models) provide a more accurate representation for visual data sets, while multivariate SRBFs (parametric models) lead to more efficient rendering performance at run-time.

An additional major advantage of multivariate SRBFs is that approximating a hierarchical set of multi-resolution BTFs can be achieved using the proposed hierarchical fitting algorithm, while sophisticated run-time mipmap texture filtering on GPUs is allowed by including smoothness energy terms in the objective function. By contrast, tensor approximation algorithms rely on a post-process to construct a hierarchical set of compressed BTFs by downsampling the spatial modes of the decomposed core tensor(s), but computational and memory overhead for efficient interpolation and smooth transitions at run-time is inevitable. Moreover, if a non-linear downsampling method is employed to obtain low-resolution BTFs with high quality, the reconstructed BTFs at levels other than the finest one are not guaranteed to be optimal solutions in the least-squares sense. In this case, tensor approximation algorithms should first create a hierarchical set of raw BTFs, and then decompose each BTF in this set based on different tensor models.

Except high-quality approximation, another obvious advantage of tensor approximation algorithms is their low off-line computational costs. Since only linear algebra operations are needed, it usually takes only tens of minutes to decompose a given BTF data set based on tensor approximation algorithms. Nevertheless, approximating a BTF with multivariate SRBFs is computationally expensive (typically several hours) due to the non-linear optimization process, even if our implementation already achieves considerable acceleration with GPUs. Thus, there is always a trade-off between off-line and run-time costs for these two categories of compression methods.

8.2 View-Dependent Occlusion Texture Functions

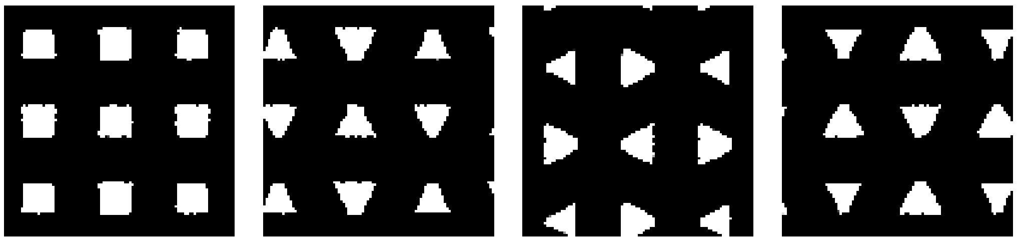
BTFs mainly focus on modeling realistic fine-scale lighting and shadowing effects of object surfaces, but neither contain shape information nor actually modify the surface micro-geometry. As a result, meso-scale shape details and shadow boundaries owing to meso-structures can not be faithfully captured. Apart from BTFs, we propose a spatially-varying appearance model, namely the *view-dependent occlusion texture function* (VOTF), to visualize complex micro-geometry of object surfaces. A VOTF is a set of two-dimensional textures in which each texture contains spatially-varying meso-scale occlusions from a sampled view direction and can be precomputed over a geometric surface by using ray tracing algorithms for visibility test. It is therefore a four-dimensional function of two variables: ω_v and $\mathbf{t} = \left[x, y\right]^T$, where ω_v represents the view direction on the unit sphere $\mathbb{S}^2$, and $\mathbf{t}$ denotes the two-dimensional spatial coordinates, x and y, of a texel. In this way, VOTFs can be employed to enhance the surface appearance of objects with shape details and silhouettes. Examples of VOTFs are shown in Figure 8.13.

Although the amount of VOTF data is usually smaller than that of BTF data, a VOTF may still consume tens of megabytes of storage space. An efficient and compact representation for VOTFs still provides a prospect of performance gains. Therefore, we propose to adopt view-dependent signed-distance functions rather than conventional binary values to encode the raw occlusion information in VOTFs. Tensor approximation algorithms then can be directly applied to decompose a VOTF for efficient run-time reconstruction.

8.2.1 Problem Formulation

To accurately approximate a VOTF, we represent it with a set of two-dimensional signed-distance textures instead of conventional binary visibility masks. Signed-distance functions [29] have been shown to successfully preserve sharp features in vector textures [142] and image structures for texture synthesis [97]. Recently, they were also applied to the approximation of visibility integrals for precomputed radiance transfer [206], further showing their potentials for modeling occlusion information. According to our experiments, the advantages of signed-distance functions still hold even after compression with tensor approximation algorithms. The intuitive explanation for this outcome is that sharp boundaries in the binary visibility masks are in fact high-frequency signals. Converting them into signed-distance functions instead produces smooth and continuous functions that would facilitate subsequent approximation. Figure 8.13(b) shows examples of a VOTF in the signed-distance representation.

Therefore, the VOTF $O(\omega_v, \mathbf{t}) \in \{0, 1\}$ is first converted into signed-distance-to-boundary textures from binary visibility masks, one texture for each view direction. For a given texel $\mathbf{t}$ in the binary mask of a view direction, we compute its nearest distance to the boundaries. Positive distance is stored if the value of texel $\mathbf{t}$ in the binary mask is equal to 0 (occluded). Otherwise, negative distance is adopted instead. The transformed VOTF is then normalized into the interval $[-1, +1]$ and organized as a third order tensor $\boldsymbol{\mathcal{A}}^{(O)} \in \mathbb{R}^{I_{\omega_v}^{(O)} \times I_x^{(O)} \times I_y^{(O)}}$ for compression using


(a) Binary visibility masks

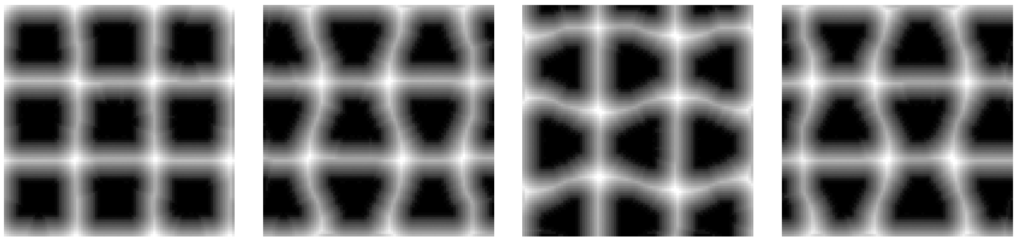

(b) Signed-distance functions

Figure 8.13: Examples of the proposed VOTFs. Each image records visibility information from a sampled view direction. In this dissertation, we consider VOTFs encoded in (a) binary visibility values or (b) signed-distance functions. The values of signed-distance functions were normalized into the interval $[-1, +1]$.

tensor approximation algorithms, where $I_{\omega_v}^{(O)}$ represents the number of sampled view directions, and $I_x^{(O)}$ as well as $I_y^{(O)}$ are respectively the spatially horizontal and vertical resolutions.

Similar to BTF compression, given the reduced ranks of each mode, namely $R_{\omega_v}^{(O)}$, $R_x^{(O)}$, and $R_y^{(O)}$, $\boldsymbol{\mathcal{A}}^{(O)}$ is decomposed as the following equation based on N-SVD:

$$\boldsymbol{\mathcal{A}}^{(O)} \approx \hat{\boldsymbol{\mathcal{A}}}^{(O)} = \boldsymbol{\mathcal{Z}}^{(O)} \times_{\omega_v} \mathbf{U}_{\omega_v}^{(O)} \times_x \mathbf{U}_x^{(O)} \times_y \mathbf{U}_y^{(O)}, \tag{8.14}$$

where $\boldsymbol{\mathcal{Z}}^{(O)} \in \mathbb{R}^{R_{\omega_v}^{(O)} \times R_x^{(O)} \times R_y^{(O)}}$ is the core tensor, and $\mathbf{U}_{\omega_v}^{(O)} \in \mathbb{R}^{I_{\omega_v}^{(O)} \times R_{\omega_v}^{(O)}}$, $\mathbf{U}_x^{(O)} \in \mathbb{R}^{I_x^{(O)} \times R_x^{(O)}}$, as well as $\mathbf{U}_y^{(O)} \in \mathbb{R}^{I_y^{(O)} \times R_y^{(O)}}$ are respectively the mode-ω_v, mode-x, and mode-y basis matrices.

For CTA and K-CTA, $\boldsymbol{\mathcal{A}}^{(O)}$ is instead approximated with total $C^{(O)}$ clusters for the view mode (and $K_{\omega_v}^{(O)}$ mixture clusters for K-CTA) by the following equation:

$$\boldsymbol{\mathcal{A}}^{(O)} \approx \hat{\boldsymbol{\mathcal{A}}}^{(O)} = \sum_{c=1}^{C^{(O)}} \left(\boldsymbol{\mathcal{Z}}_c^{(O)} \times_{\omega_v} \mathbf{U}_{\omega_v,c}^{(O)} \times_x \mathbf{U}_{x,c}^{(O)} \times_y \mathbf{U}_{y,c}^{(O)} \right), \tag{8.15}$$

where $\boldsymbol{\mathcal{Z}}_c^{(O)} \in \mathbb{R}^{R_{\omega_v}^{(O)} \times R_x^{(O)} \times R_y^{(O)}}$ represents the core tensor of cluster c, and $\mathbf{U}_{\omega_v,c}^{(O)} \in \mathbb{R}^{I_{\omega_v}^{(O)} \times R_{\omega_v}^{(O)}}$, $\mathbf{U}_{x,c}^{(O)} \in \mathbb{R}^{I_x^{(O)} \times R_x^{(O)}}$, as well as $\mathbf{U}_{y,c}^{(O)} \in \mathbb{R}^{I_y^{(O)} \times R_y^{(O)}}$ respectively specify the mode-ω_v, mode-x, and mode-y basis matrices of cluster c.

8.2.2 Rendering Issues

Meso-Structure Synthesis

The compressed VOTFs can be combined with other spatially-varying appearance models for meso-structure synthesis over arbitrary surfaces. Similar to BTFs, we simply follow almost the same approach as described in Section 8.1.1 to first reconstruct some modes of each (cluster) core tensor of a compressed VOTF as

$$\begin{cases} \hat{\boldsymbol{\mathcal{Z}}}^{(O)} = \boldsymbol{\mathcal{Z}}^{(O)} \times_x \mathbf{U}_x^{(O)} \times_y \mathbf{U}_y^{(O)} & (N\text{-SVD}), \\ \forall c,\ \hat{\boldsymbol{\mathcal{Z}}}_c^{(O)} = \boldsymbol{\mathcal{Z}}_c^{(O)} \times_x \mathbf{U}_{x,c}^{(O)} \times_y \mathbf{U}_{y,c}^{(O)} & (\text{CTA}), \\ \forall c,\ \hat{\boldsymbol{\mathcal{Z}}}_c^{(O)} = \boldsymbol{\mathcal{Z}}_c^{(O)} \times_{\omega_v} \mathbf{U}_{\omega_v,c}^{(O)} \times_x \mathbf{U}_{x,c}^{(O)} \times_y \mathbf{U}_{y,c}^{(B)} & (\text{K-CTA}), \end{cases} \tag{8.16}$$

where $\hat{\boldsymbol{\mathcal{Z}}}^{(O)} \in \mathbb{R}^{R_{\omega_v}^{(O)} \times I_x^{(O)} \times I_y^{(O)}}$ and $\hat{\boldsymbol{\mathcal{Z}}}_c^{(O)} \in \mathbb{R}^{R_{\omega_v}^{(O)} \times I_x^{(O)} \times I_y^{(O)}}$ respectively denote the reconstructed core tensor based on N-SVD and the reconstructed core tensor of cluster c based on CTA (based on K-CTA, $\hat{\boldsymbol{\mathcal{Z}}}_c^{(O)} \in \mathbb{R}^{I_{\omega_v}^{(O)} \times I_x^{(O)} \times I_y^{(O)}}$ instead). Then, each reconstructed (cluster) core tensor is concatenated along the view mode if necessary to form the VOTF exemplar for ASTS. Note that we do not combine the VOTF exemplar with the exemplar of other appearance models right now to compute the exemplar texture in the appearance space, but separately compute their appearance-space exemplars in advance. For meso-structure synthesis based on ASTS, the transformed results of the VOTF exemplar are finally included as additional image channels of the appearance-space exemplar of other appearance models. In this way, the compressed VOTF behaves as a set of view-dependent feature textures, so that ASTS can preserve both visible and invisible image structures during meso-structure synthesis.

Run-Time Rendering

At run-time, the shading color of a pixel is computed on GPUs from the compressed VOTFs, other appearance models, and the synthesized coordinate texture S. Here, we omit a detailed description of the rendering pre-process, since it is similar to that discussed in Section 8.1.1, except for steps that are related to mode-ω_l basis matrices. In brief, the decomposed results of VOTFs are densely resampled, and the spatial modes of each (cluster) core tensor are reconstructed before rendering to take advantage of hardware texture filtering. For CTA and K-CTA, the sparsity of mode-ω_v basis matrices is also fully exploited with the aid of an index texture.

To combine VOTFs with other appearance models, the run-time rendering process should be slightly modified to consist of the following steps:

1. In the pixel shader, sample the synthesized texture S for the VOTF spatial coordinates $\mathbf{t_p}$ of current pixel $\mathbf{p}$.

2. Sample the texture(s) of VOTF core tensor(s) for all the data that correspond to $\mathbf{t_p}$, and

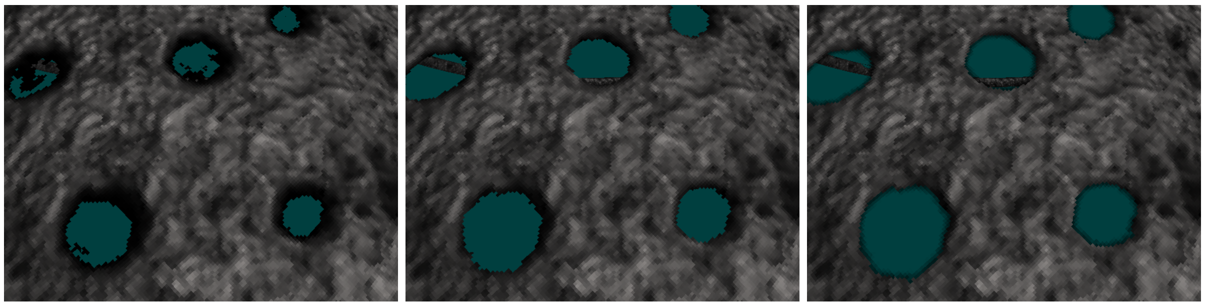

(a) Binary visibility masks (b) Signed-distance functions (c) Signed-distance functions (anti-aliased)

Figure 8.14: Comparison between different encoding schemes for VOTFs. Inside triangle faces of the model are culled to clearly show the difference. For binary visibility masks, the threshold $T^{(O)}$ and weight $W^{(O)}$ in Equation 8.17 were, respectively, set to 0.5 and 1. For signed-distance functions, $T^{(O)}$ and $W^{(O)}$ were, respectively, set to 0.05 and 12.

the texture of VOTF mode-ω_v basis matrices for all the components of current novel view direction.

3. The approximated VOTF value of pixel $\mathbf{p}$, $\hat{O}_{\mathbf{p}}$, is given by reconstructing the view mode according to the tensor approximation algorithm.

4. A user-defined visibility mapping function, for example Equation 8.17, is then applied to map $\hat{O}_{\mathbf{p}}$ into a visibility value $\hat{O}'_{\mathbf{p}}$ within the interval $[0, 1]$.

5. If pixel $\mathbf{p}$ is visible, namely $\hat{O}'_{\mathbf{p}} > 0$, compute its shading color based on other appearance models. Otherwise, discard it.

The main purpose of the visibility mapping function in Step 4 is to determine final visibility values and avoid the aliasing problem that results from meso-scale visibility. It should be designed to generate visually pleasing effects at silhouette boundaries. In the experiments, the mapping function is defined as

$$\hat{O}'_{\mathbf{p}} = \begin{cases} 1 & \text{if } \hat{O}_{\mathbf{p}} \geq T^{(O)} & \text{(fully visible)}, \\ W^{(O)}\hat{O}_{\mathbf{p}} & \text{if } T^{(O)} > \hat{O}_{\mathbf{p}} > 0 & \text{(partially visible)}, \\ 0 & \text{otherwise} & \text{(invisible)}, \end{cases} \tag{8.17}$$

where $T^{(O)}$ and $W^{(O)}$ are respectively the user-defined threshold and weight. In this way, $\hat{O}'_{\mathbf{p}}$ can be utilized as an alpha value to blend a partially visible pixel with its background as shown in Figure 8.14(c). To approximate the blending effects without sorting, fully visible pixels are rendered first. We then keep the contents of frame buffer and blend partially visible pixels that pass the depth test. This correctly captures the order of a fully visible pixel and a partially visible pixel, but disregards the blending effects between two partially visible pixels. The resulting artifacts are ignorable in practice since the amount of partially visible pixels is usually small.

8.2.3 Experimental Results

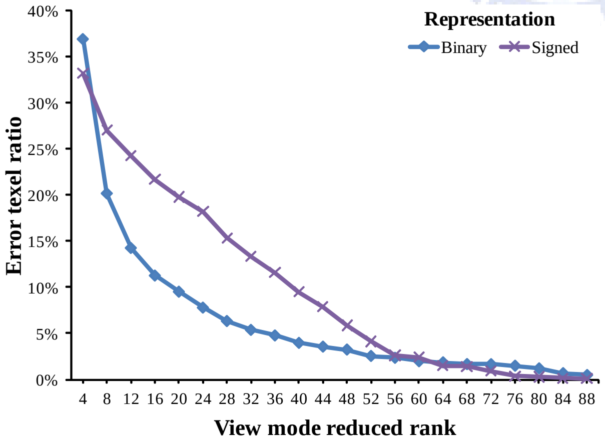
The experiments and simulation timings of tensor approximation algorithms for VOTF compression were conducted and measured on a workstation with an Intel Core 2 Extreme QX9650 CPU, an NVIDIA GeForce GTX 280 graphics card, and 8 gigabytes main memory under Microsoft Windows Vista operating system. Each raw VOTF was obtained by ray-tracing a geometric surface from different view directions. In the experimental results of CTA and K-CTA, their iterative variants were employed instead of static ones. For meso-structure synthesis, we typically transformed a compressed VOTF into a twelve-dimensional appearance-space exemplar.

Figure 8.15 plots the error texel ratio versus the mode-ω_v reduced rank based on binary visibility masks or signed-distance functions for VOTFs. Note that we compare the two encoding schemes based on the the same total number of mode-ω_v reduced rank, which leads to similar storage space. Figure 8.15 shows that signed-distance functions begin to outperform binary visibility masks from a certain cross point, for example, 60 in Figure 8.15(a) and 36 in Figure 8.15(b). This implies that the high-frequency signals in binary visibility masks significantly limit the approximation ability of tensor approximation algorithms. In practice, we have found that a higher mode-ω_v reduced rank than the value of this cross point is frequently necessary to render high-quality images.

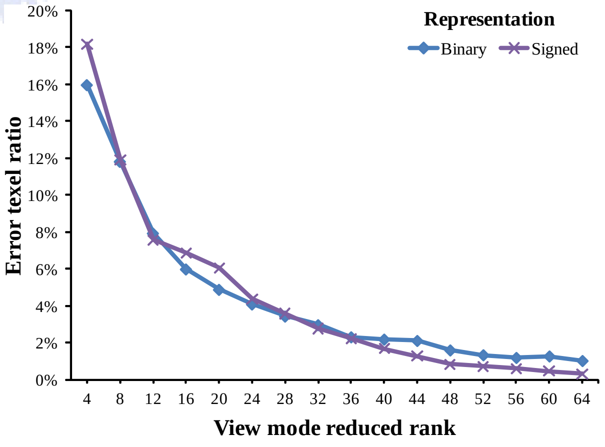
Table 8.6 compares the statistics and timing measurements of different tensor approximation algorithms, including N-SVD, CTA, and K-CTA. In Figures 8.16–8.18, we also demonstrate the plots and the reconstructed VOTFs of the three multi-linear models. From these figures and Table 8.6, signed-distance functions obviously outperform binary visibility masks in visual quality and approximation errors (in the number of error texels) for encoding VOTFs. After compression, signed-distance functions tend to reconstruct more continuous signals and less noises at silhouette boundaries than traditional binary masks (Figure 8.14). In Chapter 9, we will further present the experimental results of combining VOTFs with BTFs for run-time rendering based on the proposed framework of bi-scale radiance transfer.

Table 8.6: Statistics and timing measurements of different tensor approximation algorithms for VOTFs. We list the total number of error texels for the two representations: (B) binary visibility masks; (S) signed-distance functions. For binary visibility masks, a threshold value of 0.5 was adopted to determine the visibility of a texel from the reconstructed data, whereas a threshold value of 0 was employed instead for signed-distance functions. All decomposed data were stored as half-precision (16-bit) floating point numbers [65].

Material	Fiber			RoughHole		
$I^{(O)}_{\omega_v} \times I^{(O)}_x \times I^{(O)}_y$	100×96×96			100×96×96		
Raw data (MB)	3.52			3.52		
Algorithm	N-SVD	CTA	K-CTA	N-SVD	CTA	K-CTA
$R^{(O)}_{\omega_v}$	80	4	4	44	4	4
$R^{(O)}_x \times R^{(O)}_y$	80×80	80×80	80×80	80×80	80×80	80×80
Clusters: $C^{(O)}$	1	20	20	1	11	11
Mixture clusters: $K^{(O)}_{\omega_v}$	1	1	3	1	1	3
Compressed data (MB)	1.02	1.58	1.61	0.57	0.87	0.88
Error texels (B)	732	2222	1046	572	1938	1029
Error texels (S)	71	2027	291	325	969	597
Compression time (sec.)	1.83	17.24	130.68	0.67	12.43	114.13

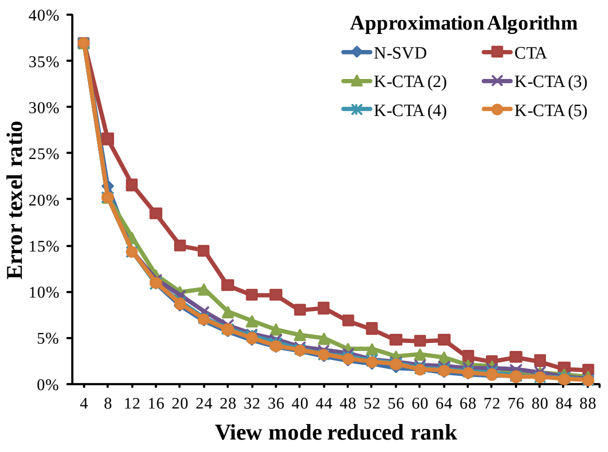


(a) *Fiber*

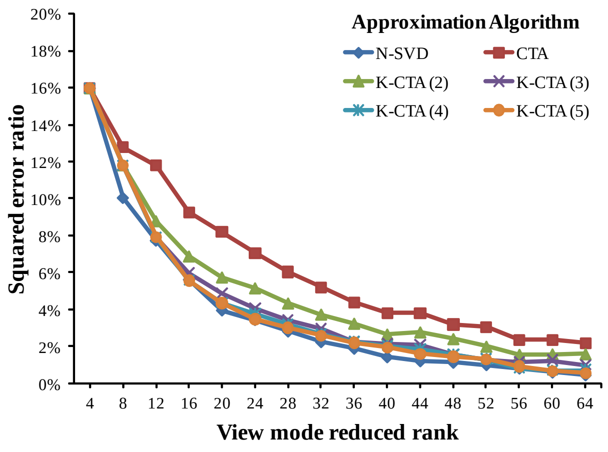


(b) *RoughHole*

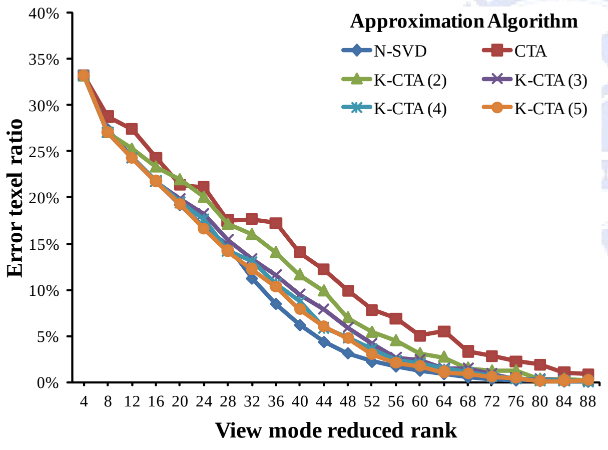
Figure 8.15: Plots of the error texel ratio versus the mode-ω_v reduced rank based on different encoding schemes for VOTFs, including *binary visibility masks* (Binary) and *signed-distance functions* (Signed). The VOTFs were approximated using K-CTA with three mixture clusters for each mode-ω_v sub-tensor. The horizontal axis in each plot represents various total mode-ω_v reduced ranks whose values correspond to $R^{(O)}_{\omega_v} C^{(O)}$, with $R^{(O)}_{\omega_v} = 4$ for each cluster. For other configurations of tensor approximation algorithms, please refer to Table 8.6.



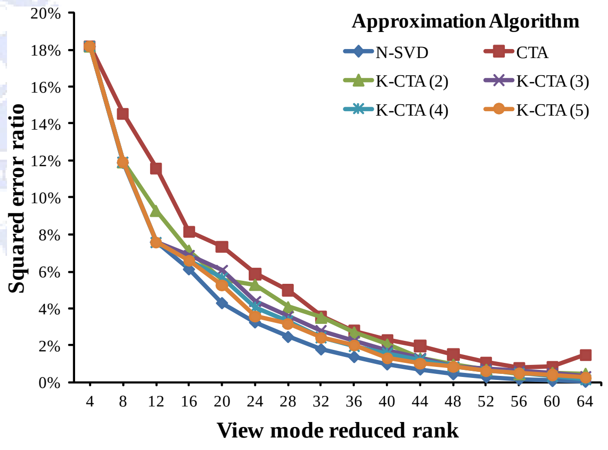
(a) *Fiber* (binary visibility masks)



(b) *RoughHole* (binary visibility masks)

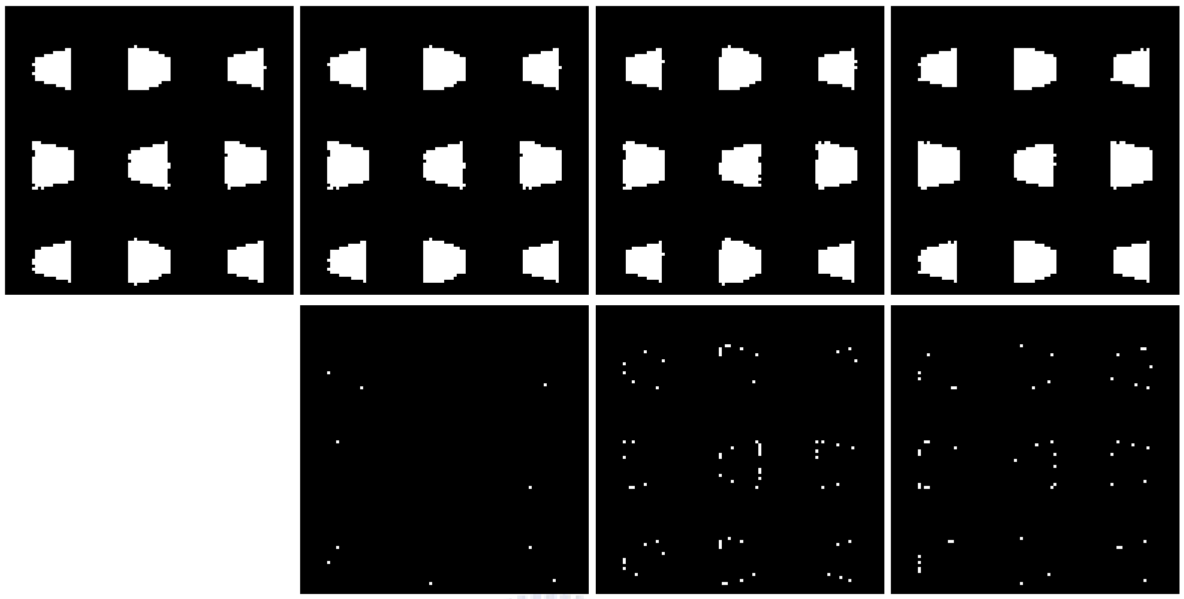


(c) *Fiber* (signed-distance functions)

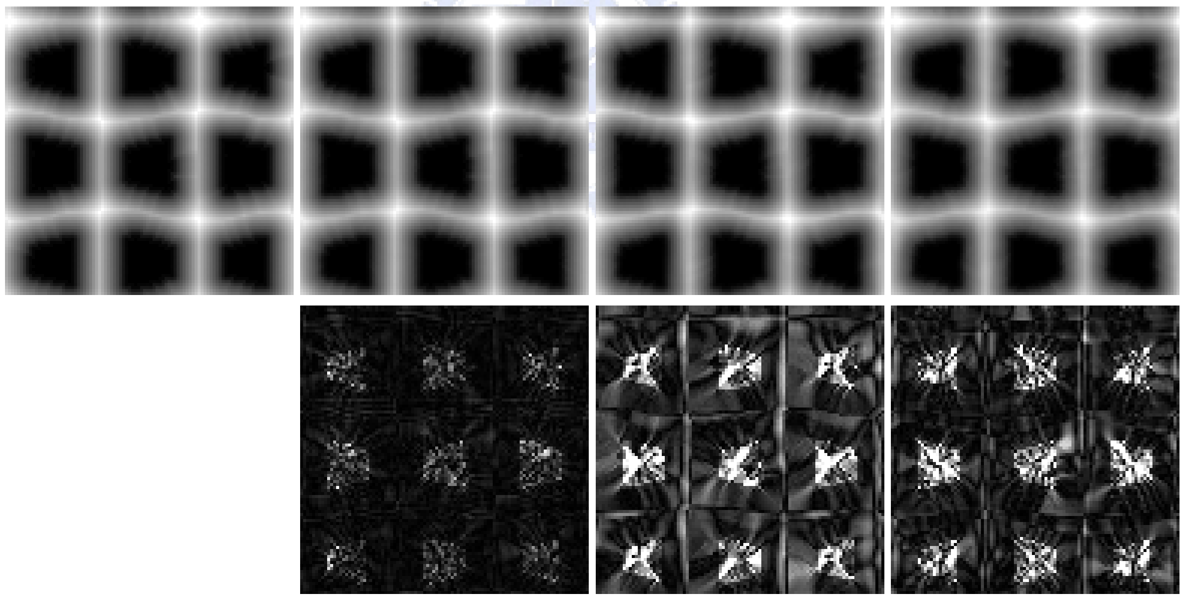


(d) *RoughHole* (signed-distance functions)

Figure 8.16: Plots of the error texel ratio versus the mode-ω_v reduced rank based on different tensor approximation algorithms for VOTFs. Each value in parentheses for K-CTA specifies the number of mixture clusters. For CTA and K-CTA, the horizontal axis in each plot represents various total mode-ω_v reduced ranks whose values correspond to $R^{(O)}_{\omega_v} C^{(O)}$, with $R^{(O)}_{\omega_v} = 4$ for each cluster. For other configurations of tensor approximation algorithms, please refer to Table 8.6.

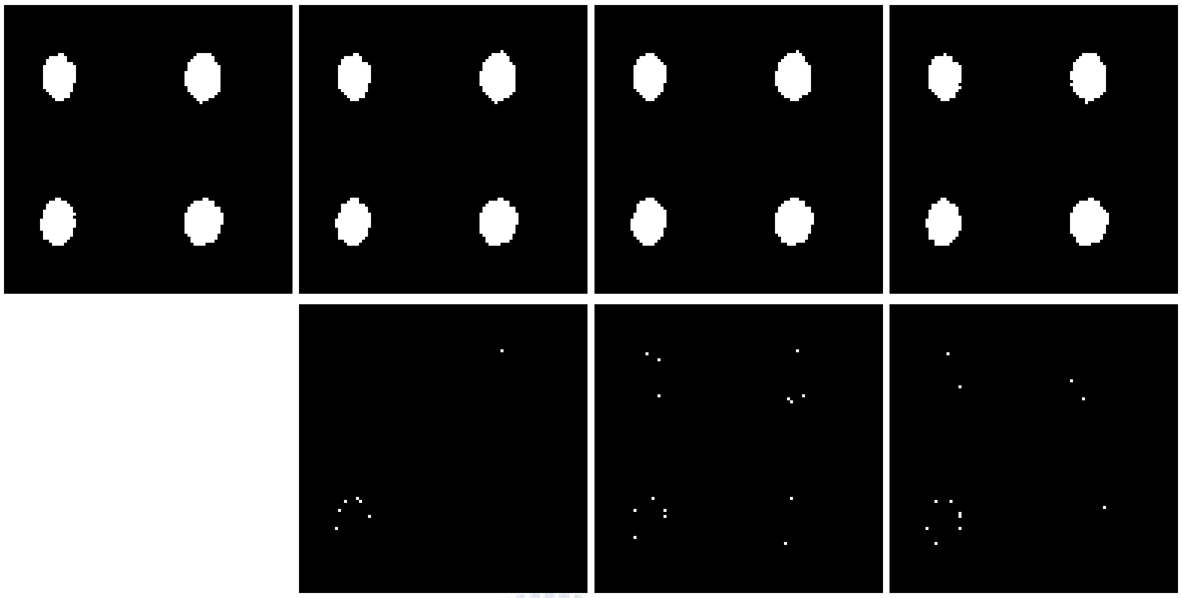


(a) Binary visibility masks (view direction – zenith angle: 27°, azimuth angle: 279°)



(b) Signed-distance functions (view direction – zenith angle: 27°, azimuth angle: 279°)

Figure 8.17: Reconstructed images of the VOTF for the material *Fiber* based on different tensor approximation algorithms. From left to right: raw data; N-SVD; CTA; K-CTA. From top to bottom in each row: reconstructed images; absolute difference images (scaled by a factor of 20 for signed-distance functions).


(a) Binary visibility masks (view direction – zenith angle: 45°, azimuth angle: 189°)

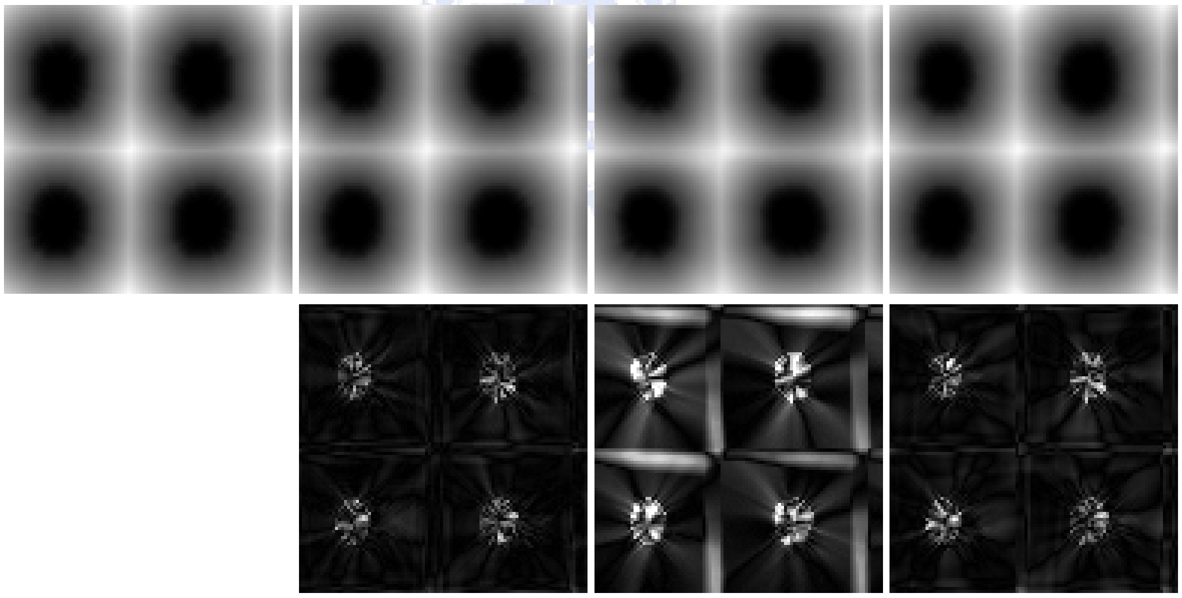

(b) Signed-distance functions (view direction – zenith angle: 45°, azimuth angle: 189°)

Figure 8.18: Reconstructed images of the VOTF for the material *RoughHole* based on different tensor approximation algorithms. From left to right: raw data; N-SVD; CTA; K-CTA. From top to bottom in each row: reconstructed images; absolute difference images (scaled by a factor of 20 for signed-distance functions).

Chapter 9

Application III: Radiance Transfer

Reflectance functions and spatially-varying appearance models only account for *local* illumination responses of object surfaces. They can not be utilized to cast shadows or simulate interreflections, since *global* lighting interactions between different surface points or even different objects are totally ignored. Nevertheless, rendering global illumination effects in real time remains a major challenge in computer graphics. It usually requires a time-consuming iterative or recursive algorithm to accurately simulate global light transport. *Precomputed radiance transfer* (PRT) [162] has recently attracted much attention owing to its ability to solve this issue in dynamic low-frequency lighting environments. The successive *bi-scale radiance transfer* (BRT) [163] further integrated spatially-varying materials into the PRT framework to greatly enhance the visual quality of rendered images.

In the first section of this chapter, we introduce a novel all-frequency PRT algorithm based on the proposed univariate *spherical radial basis functions* (SRBFs) and tensor approximation algorithms to accurately approximate all-frequency signals. Compared with previous PRT algorithms, the proposed approach allows real-time rendering with comparable quality in high-frequency lighting environments. The data storage is also more compact than previous related algorithms. In the second section of this chapter, we additionally present an all-frequency BRT algorithm to enhance the rendered results with complex surface appearance and micro-geometry. Our BRT algorithm not only extends previous all-frequency PRT methods with complex meso-scale surface appearance, but also permits view-dependent rendering of complex objects on GPUs at real-time rates.

9.1 Precomputed Radiance Transfer

PRT [125, 162] can capture complex global illumination effects, such as inter-reflections, subsurface scattering, caustics, and shadows, from dynamic lighting environments. As a preprocess, PRT precomputes a solution to the global light transport of a static scene, and only records the final incident/exitant radiance of each sample point on object surfaces in different illumination/view directions. To further decrease storage space and computational costs, the recorded data should be approximated for efficient run-time rendering. In this way, PRT transforms the time-consuming global illumination problem into mainly a data compression and decompression issue.

For photo-realistic image synthesis, although all-frequency PRT [125] provides significantly better image quality than its low-frequency counterpart [162], the amount of raw all-frequency PRT data frequently exceeds tens of gigabytes. In this section, we thus introduce a novel algorithm to approximate the enormous all-frequency PRT data. Our approach relies on the proposed univariate SRBFs to efficiently model high-frequency lighting information and *clustered tensor approximation* (CTA) to exploit coherence in the PRT data of different sample points.

This section is a rewritten version of some parts of our published paper [179]. Here, we focus on the PRT application of the proposed univariate SRBFs as well as tensor approximation algorithms, and present the univariate SRBF representation for PRT in more details.

9.1.1 Problem Formulation

Consider a static scene and distant illumination. The direct illumination term in the rendering equation [73] over the unit sphere $\mathbb{S}^2$ is defined as

$$L_{o,\mathbf{p}}(\omega_v) = \int_{\mathbb{S}^2} L_{in}(\omega_l)\rho_{\mathbf{p}}(\omega_l,\omega_v)V_{\mathbf{p}}(\omega_l)\left|\omega_l \cdot \mathbf{n}_{\mathbf{p}}\right| d\omega_l, \tag{9.1}$$

where $L_{o,\mathbf{p}}(\omega_v) \in \mathbb{R}$ represents the exitant radiance from a sample point $\mathbf{p} \in \mathbb{R}^3$ on an object surface in view direction $\omega_v \in \mathbb{S}^2$, $L_{in}(\omega_l) \in \mathbb{R}$ is the incident radiance from distant illumination direction $\omega_l \in \mathbb{S}^2$, and $\rho_{\mathbf{p}}(\omega_l,\omega_v) \in \mathbb{R}$, $V_{\mathbf{p}}(\omega_l) \in \{x \in \mathbb{R} \mid 0 \le x \le 1\}$, as well as $\mathbf{n}_{\mathbf{p}} \in \mathbb{S}^2$ respectively denote the *bidirectional reflectance distribution function* (BRDF), the visibility function from direction ω_l, and the surface normal at point $\mathbf{p}$.

Suppose that the BRDF at point $\mathbf{p}$ is approximated with total $I_{\rho_{\mathbf{p}}}$ sets of illumination- and view-dependent basis functions by using singular value decomposition [105, 195]:

$$\rho_{\mathbf{p}}(\omega_l,\omega_v) \approx \sum_{i=1}^{I_{\rho_{\mathbf{p}}}} D_{\mathbf{p},i}(\omega_l)E_{\mathbf{p},i}(\omega_v), \tag{9.2}$$

where $D_{\mathbf{p},i}(\omega_l) \in \mathbb{R}$ and $E_{\mathbf{p},i}(\omega_v) \in \mathbb{R}$ respectively specify the i-th set of illumination- and view-dependent basis functions of the BRDF at point $\mathbf{p}$. Equation 9.1 then becomes

$$L_{o,\mathbf{p}}(\omega_v) \approx \sum_{i=1}^{I_{\rho_{\mathbf{p}}}} \Big(E_{\mathbf{p},i}(\omega_v)\int_{\mathbb{S}^2} T_{\mathbf{p},i}(\omega_l)L_{in}(\omega_l)d\omega_l\Big), \tag{9.3}$$

$$T_{\mathbf{p},i}(\omega_l) = D_{\mathbf{p},i}(\omega_l)V_{\mathbf{p}}(\omega_l)\left|\omega_l \cdot \mathbf{n}_{\mathbf{p}}\right|, \tag{9.4}$$

where $T_{\mathbf{p},i}(\omega_l) \in \mathbb{R}$ is known as the radiance transfer function for the i-th illumination-dependent basis function $D_{\mathbf{p},i}(\omega_l)$ at point $\mathbf{p}$, which describes the transport of light for $D_{\mathbf{p},i}(\omega_l)$ between the distant illumination source and the incident radiance to point $\mathbf{p}$ in direction ω_l.

Our approach represents the distant illumination $L_{in}(\omega_l)$ and the radiance transfer functions $T_{\mathbf{p},1}(\omega_l), T_{\mathbf{p},2}(\omega_l), \ldots, T_{\mathbf{p},I_{\rho_\mathbf{p}}}(\omega_l)$ in univariate SRBF expansions as

$$L_{in}(\omega_l) \approx \sum_{j=1}^{J} \beta_j^{(L_{in})} G\big(\omega_l \cdot \xi_j^{(L_{in})} | \lambda_j^{(L_{in})}\big), \tag{9.5}$$

$$\forall i,\ T_{\mathbf{p},i}(\omega_l) \approx \sum_{k=1}^{K} \beta_k^{(T_{\mathbf{p},i})} G\big(\omega_l \cdot \xi_k^{(T_\mathbf{p})} | \lambda_k^{(T_\mathbf{p})}\big), \tag{9.6}$$

where $\{\beta_j^{(L_{in})} \in \mathbb{R}\}_{j=1}^{J}$, $\{\xi_j^{(L_{in})} \in \mathbb{S}^2\}_{j=1}^{J}$, and $\{\lambda_j^{(L_{in})} \in \mathbb{R}\}_{j=1}^{J}$ denote the basis coefficient set, the center set, and the bandwidth set of the distant illumination $L_{in}(\omega_l)$, and $\{\beta_k^{(T_{\mathbf{p},i})} \in \mathbb{R}\}_{k=1}^{K}$ $\{\xi_k^{(T_\mathbf{p})} \in \mathbb{S}^2\}_{k=1}^{K}$, as well as $\{\lambda_k^{(T_\mathbf{p})} \in \mathbb{R}\}_{k=1}^{K}$ are respectively the basis coefficient set, the center set, and the bandwidth set of the i-th radiance transfer function $T_{\mathbf{p},i}(\omega_l)$ at point $\mathbf{p}$. To simplify the proposed algorithm, note that we use the same center and bandwidth sets for all radiance transfer functions at point $\mathbf{p}$ in Equation 9.6.

By combining Equation 9.3 with Equations 9.5 and 9.6, the exitant radiance at point $\mathbf{p}$ is approximated by

$$L_{o,\mathbf{p}}(\omega_v) \approx \sum_{i=1}^{I_{\rho_\mathbf{p}}} \Big(E_{\mathbf{p},i}(\omega_v) \sum_{j=1}^{J} \sum_{k=1}^{K} \int_{\mathbb{S}^2} \beta_j^{(L_{in})} \beta_k^{(T_{\mathbf{p},i})} G\big(\omega_l \cdot \xi_j^{(L_{in})} | \lambda_j^{(L_{in})}\big) G\big(\omega_l \cdot \xi_k^{(T_\mathbf{p})} | \lambda_k^{(T_\mathbf{p})}\big) d\omega_l \Big). \tag{9.7}$$

For discrete observations, Equation 9.7 can be further rewritten in a matrix form as

$$\mathbf{l}_{o,\mathbf{p}} \approx \mathbf{E_p T_p A_p l}_{in}, \tag{9.8}$$

where $\mathbf{l}_{o,\mathbf{p}}$ denotes the exitance radiance vector at point $\mathbf{p}$, $\mathbf{E_p}$ specifies the view-dependent basis matrix of the BRDF at point $\mathbf{p}$, and

$$\mathbf{T_p} = \begin{bmatrix} \beta_1^{(T_{\mathbf{p},1})} & \cdots & \beta_K^{(T_{\mathbf{p},1})} \\ \vdots & \ddots & \vdots \\ \beta_1^{(T_{\mathbf{p},I_{\rho_\mathbf{p}}})} & \cdots & \beta_K^{(T_{\mathbf{p},I_{\rho_\mathbf{p}}})} \end{bmatrix}, \tag{9.9}$$

$$\mathbf{A_p} = \begin{bmatrix} (G \star_2 G)\big(\xi_1^{(L_{in})} \cdot \xi_1^{(T_\mathbf{p})} | \lambda_1^{(L_{in})}, \lambda_1^{(T_\mathbf{p})}\big) & \cdots & (G \star_2 G)\big(\xi_J^{(L_{in})} \cdot \xi_1^{(T_\mathbf{p})} | \lambda_J^{(L_{in})}, \lambda_1^{(T_\mathbf{p})}\big) \\ \vdots & \ddots & \vdots \\ (G \star_2 G)\big(\xi_1^{(L_{in})} \cdot \xi_K^{(T_\mathbf{p})} | \lambda_1^{(L_{in})}, \lambda_K^{(T_\mathbf{p})}\big) & \cdots & (G \star_2 G)\big(\xi_J^{(L_{in})} \cdot \xi_K^{(T_\mathbf{p})} | \lambda_J^{(L_{in})}, \lambda_K^{(T_\mathbf{p})}\big) \end{bmatrix}, \tag{9.10}$$

$$\mathbf{l}_{in} = \begin{bmatrix} \beta_1^{(L_{in})} & \cdots & \beta_J^{(L_{in})} \end{bmatrix}^T. \tag{9.11}$$

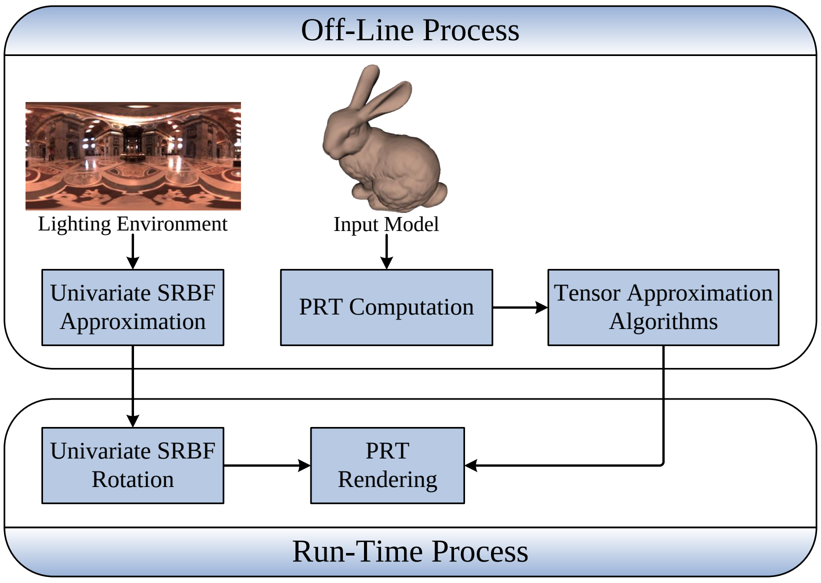


Figure 9.1: System diagram of the proposed all-frequency PRT algorithm.

Moreover, $\mathbf{A}_\mathbf{p}$ is known as the interpolation matrix at point $\mathbf{p}$ between the radiance transfer matrix $\mathbf{T}_\mathbf{p}$ and the incident illumination vector $\mathbf{l}_{in}$. Since univariate SRBFs are non-orthonormal basis functions, $\mathbf{A}_\mathbf{p}$ is necessary to account for the spherical singular integral between two univariate spherical functions in univariate SRBF expansions.

9.1.2 Algorithm

Overview

As illustrated in Figure 9.1, the proposed all-frequency PRT algorithm consists of off-line and run-time processes. In the off-line process, the radiance transfer matrix at each object vertex is precomputed based on a set of uniformly distributed univariate SRBFs. To exploit inter-vertex coherence, the radiance transfer matrices are further organized as a tensor and then compressed using the proposed CTA (or K-CTA) algorithm. The lighting environment, frequently a high dynamic range environment map, is also modeled with scattered univariate SRBFs for efficient run-time rendering. In the run-time process, the lighting environment is first rotated to align with objects. The shading color of each vertex then can be reconstructed on GPUs from the rotated lighting environment and the decomposed results of CTA.

Off-Line Process

PRT computation: For an input scene, we first precompute the radiance transfer matrix at each vertex. The visibility function at vertex $\mathbf{p}$, namely $V_\mathbf{p}(\omega_l)$, is obtained by rendering the scene into a cube map with flat shading. To avoid aliasing artifacts, the cube map is super-sampled with a $6\times128\times128$ resolution, and then down-sampled to $6\times32\times32$ texels. Our implementation currently only considers direct illumination. Nevertheless, inter-reflections can be handled by applying GPUs to render triangle index textures. It is thus quite intuitive to incorporate indirect illumination into the proposed PRT framework.

The BRDF at vertex $\mathbf{p}$, namely $\rho_{\mathbf{p}}(\omega_l, \omega_v)$, is assumed to be the same everywhere on an object without losing generality, and approximated using singular value decomposition by retaining $I_{\rho_{\mathbf{p}}}$ terms of illumination- and view-dependent basis functions with the first $I_{\rho_{\mathbf{p}}}$ largest singular values. This scheme will provide a preliminary data reduction for the raw PRT data sets, while preserving most of the important information for further analysis in the following CTA stage. In our current configuration, $I_{\rho_{\mathbf{p}}}$ simply equals to 1 for diffuse objects, whereas $I_{\rho_{\mathbf{p}}}$ is set to 16 for glossy objects. The value of $I_{\rho_{\mathbf{p}}}$ depends on the complexity of the BRDF, and a setting of $I_{\rho_{\mathbf{p}}} \leq 16$ is typically sufficient for analytic BRDFs [105, 195]. For complex BRDFs, we can set $I_{\rho_{\mathbf{p}}}$ to a value such that the sum of the first $I_{\rho_{\mathbf{p}}}$ largest singular values exceeds a certain percentage, for example 95%, of the total sum of all singular values.

The SRBF center set for approximating radiance transfer functions is generated by repeatedly subdividing an icosahedron, while the bandwidth set is assigned with the same value determined from the minimum geodesic distance between two centers, which is related to the variance with respect to the center of a univariate SRBF. We currently subdivide an icosahedron into a mesh with 642 vertices as the centers, and set the variance to $\pi/40$ radians for deriving the bandwidths. For more details about the relationship between the variance and the bandwidth parameters for Abel-Poisson and Gaussian SRBFs, please refer to Section 3.4.2 and [45, 123]. Although this configuration is sufficient for general cases according to our experiments, a finer mesh on $\mathbb{S}^2$ and a smaller value of variance (a larger value of bandwidth) can always be adopted to handle more complicated radiance transfer functions. Finally, each radiance transfer function is projected onto the univariate SRBFs using ordinary or regularized least-squares projection (Section 3.3), and the resulting coefficients are converted to half-precision (16-bit) floating point numbers [65].

The radiance transfer functions are not modeled with scattered univariate SRBFs for two reasons. First, the non-linear optimization process (Section 3.4.1) is computationally too expensive to be performed at each object vertex, even accelerated with GPUs. Second, after approximating radiance transfer matrices using CTA, if the SRBF center (or bandwidth) set of each vertex is different, we must compute the interpolation matrix at each vertex, and can not take advantage of the per-cluster operation at run-time (refer to next sub-section *Run-Time Process* for more details about the per-cluster operation). We aim to investigate these issues in the future.

Clustered tensor approximation: After approximating the radiance transfer matrix at each vertex with uniform univariate SRBFs, the amount of PRT data is still cumbersome for real-time rendering applications. To solve this problem, the radiance transfer matrices of an object are further organized as a third order tensor $\boldsymbol{\mathcal{A}}^{(\mathbf{T})} \in \mathbb{R}^{I_{\omega_v}^{(\mathbf{T})} \times I_{\omega_l}^{(\mathbf{T})} \times I_{vert}^{(\mathbf{T})}}$ (Figure 9.2), where $I_{\omega_v}^{(\mathbf{T})} = I_{\rho_{\mathbf{p}}}$, $I_{\omega_l}^{(\mathbf{T})} = K$, and $I_{vert}^{(\mathbf{T})}$ is the number of object vertices.

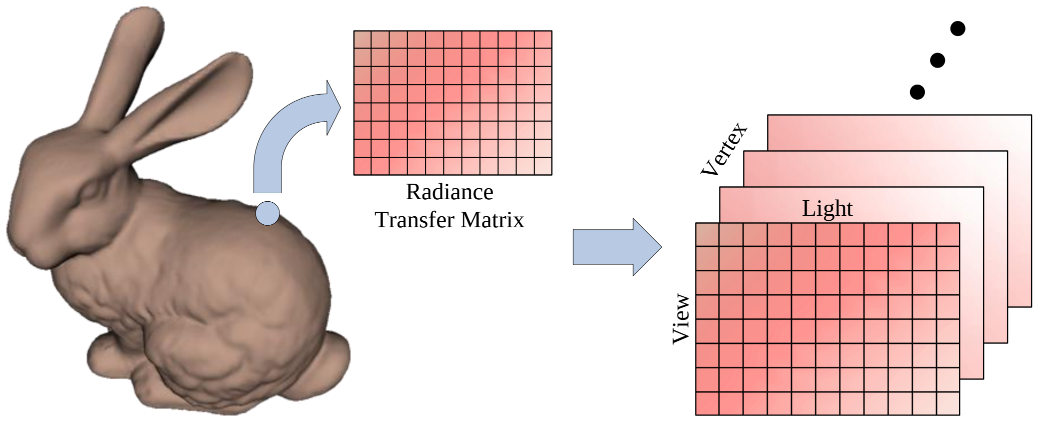


Figure 9.2: Tensor representation for radiance transfer matrices in the proposed all-frequency PRT algorithm. The transfer matrices of an object are organized as a third order tensor $\boldsymbol{\mathcal{A}}^{(\mathbf{T})}$ with three modes: view (the number of BRDF terms after singular value decomposition), light (the number of univariate SRBFs), and vertex.

Given the reduced ranks of each mode, namely $R^{(\mathbf{T})}_{\omega_v}$, $R^{(\mathbf{T})}_{\omega_l}$, and $R^{(\mathbf{T})}_{vert}$, we set total $C^{(\mathbf{T})}$ clusters for the vertex mode, and then decompose $\boldsymbol{\mathcal{A}}^{(\mathbf{T})}$ based on CTA as

$$\boldsymbol{\mathcal{A}}^{(\mathbf{T})} \approx \hat{\boldsymbol{\mathcal{A}}}^{(\mathbf{T})} = \sum_{c=1}^{C^{(\mathbf{T})}} \left(\boldsymbol{\mathcal{Z}}^{(\mathbf{T})}_c \times_{\omega_v} \mathbf{U}^{(\mathbf{T})}_{\omega_v,c} \times_{\omega_l} \mathbf{U}^{(\mathbf{T})}_{\omega_l,c} \times_{vert} \mathbf{U}^{(\mathbf{T})}_{vert,c} \right), \tag{9.12}$$

where $\boldsymbol{\mathcal{Z}}^{(\mathbf{T})}_c \in \mathbb{R}^{R^{(\mathbf{T})}_{\omega_v} \times R^{(\mathbf{T})}_{\omega_l} \times R^{(\mathbf{T})}_{vert}}$ denotes the core tensor of cluster c, and $\mathbf{U}^{(\mathbf{T})}_{\omega_v,c} \in \mathbb{R}^{I^{(\mathbf{T})}_{\omega_v} \times R^{(\mathbf{T})}_{\omega_v}}$, $\mathbf{U}^{(\mathbf{T})}_{\omega_l,c} \in \mathbb{R}^{I^{(\mathbf{T})}_{\omega_l} \times R^{(\mathbf{T})}_{\omega_l}}$, as well as $\mathbf{U}^{(\mathbf{T})}_{vert,c} \in \mathbb{R}^{I^{(\mathbf{T})}_{vert} \times R^{(\mathbf{T})}_{vert}}$ are respectively the mode-ω_v, mode-ω_l, and mode-$vert$ basis matrices of cluster c. Instead of CTA, note that one can also apply K-CTA to approximate $\boldsymbol{\mathcal{A}}^{(\mathbf{T})}$, but the mode-$vert$ basis matrices should additionally satisfy the sparse and membership constraints in Equation 6.1.

Lighting environment: To efficiently shade objects whose radiance transfer functions are represented in univariate SRBFs, the lighting environment $L_{in}(\omega_l)$ is also modeled with univariate SRBFs. Although $L_{in}(\omega_l)$ can be approximated by adopting a set of uniform univariate SRBFs, we propose to approximate $L_{in}(\omega_l)$ with a compact set of scattered univariate SRBFs as Equation 9.5. Similar to high dynamic range environment maps (Section 7.1), Algorithm 3.1 can be applied to obtain the SRBF parameters in Equation 9.5.

Run-Time Process

The run-time PRT rendering process consists of the following steps:

1. Align $\mathbf{l}_{in}$ with objects. This includes rotating the SRBF center set $\{\xi^{(L_{in})}_j \in \mathbb{S}^2\}_{j=1}^{J}$, performing the spherical singular integrals to acquire $\mathbf{A_p}$, and computing the matrix-vector product $\mathbf{l}'_{in} = \mathbf{A_p}\mathbf{l}_{in}$.

2. For each cluster c, obtain the representative radiance transfer matrix $\hat{\mathbf{Z}}_c^{(\mathbf{T})}$ by

$$\forall c,\ \hat{\mathbf{Z}}_c^{(\mathbf{T})} = uf_{\omega_v}\Big(\boldsymbol{\mathcal{Z}}_c^{(\mathbf{T})} \times_{\omega_l} \big((\mathbf{l}'_{in})^T \mathbf{U}^{(\mathbf{T})}_{\omega_l,c}\big)\Big). \tag{9.13}$$

3. For each vertex $\mathbf{p}$, reconstruct the radiance transfer vector $\mathbf{r}_\mathbf{p}$ by

$$\forall \mathbf{p},\ \mathbf{r}_\mathbf{p} = \sum_{c=1}^{C^{(\mathbf{T})}} \hat{\mathbf{Z}}_c^{(\mathbf{T})} \big(\mathbf{U}^{(\mathbf{T})}_{vert,c}\big)^T_{\mathbf{p}*}. \tag{9.14}$$

4. For each pixel, sample the view-dependent basis matrix $\mathbf{E}_\mathbf{p}$ for current novel view direction. The final shading color is then the dot product of the sampled results and $\mathbf{r}_\mathbf{p}$.

Steps 1 and 2 are performed on CPUs, whereas the other two steps are executed on GPUs. Step 1 is a per-object operation that is performed when the alignment of a lighting environment with an object is required. Step 2 is a per-cluster operation rather than a per-vertex one. Since the number of clusters is much smaller than the number of object vertices, Step 2 greatly reduces the computational costs at run-time. After that, the representative radiance transfer matrix of each cluster is transferred to GPUs for executing Step 3 in the vertex shader and Step 4 in the pixel shader. Note that the per-vertex operation in Step 3 can be computed from only the non-zero entries of $\mathbf{U}^{(\mathbf{T})}_{vert,c}$. Moreover, the sampling of $\mathbf{E}_\mathbf{p}$ in Step 4 can be conducted using a pre-filtered two-dimensional texture in the parabolic parameterization [62]. To prevent aliasing artifacts, this texture is super-sampled and then filtered down to a desired resolution (typically 256×256).

9.1.3 Experimental Results

The experiments and simulation timings of the proposed all-frequency PRT algorithm were conducted and measured on a workstation with an Intel Core 2 Extreme QX6700 CPU, an NVIDIA GeForce 8800 Ultra graphics card, and 4 gigabytes main memory under Microsoft Windows Vista operating system. We employed NVIDIA CUDA [128] to accelerate the off-line process and Microsoft Direct3D 9 for run-time rendering on GPUs. In our experiments, the univariate Gaussian SRBFs were adopted to represent radiance transfer functions and lighting environments. The SRBF center and bandwidth sets were also constrained to be the same for red, green, and blue channels of a radiance transfer function or a lighting environment. To improve rendering performance, the super-clustering technique [161] was also applied to decrease the number of redrawn triangles.

Table 9.1 lists the statistics and timing measurements of the proposed all-frequency PRT algorithm for various models. It shows that proposed approach can achieve high compression ratios and real-time rendering performance using modern GPUs. For our configurations, the uniform univariate SRBF representation followed by conversion to half-precision (16-bit) float-

Table 9.1: Statistics and timing measurements of the proposed all-frequency PRT algorithm using the Cook-Torrance model [25] and the high dynamic range lighting environment *St. Peter's Basilica*. In the row *Frames per second*, we list the rendering performance when the viewpoint or the lighting environment changes.

Model	Buddha	Bunny	Teapot
Vertices: $I_{vert}^{(\mathbf{T})}$	52k	61k	41k
$I_{\omega_v}^{(\mathbf{T})} \times I_{\omega_l}^{(\mathbf{T})}$	16×642	16×642	16×642
Raw PRT data (GB)	19.12	22.35	15.11
PRT data in SRBFs (GB)	1.00	1.17	0.79
$R_{\omega_v}^{(\mathbf{T})} \times R_{\omega_l}^{(\mathbf{T})} \times R_{vert}^{(\mathbf{T})}$	8×48×24	8×48×24	8×48×24
Clusters: $C^{(\mathbf{T})}$	140	170	110
Mixture clusters: $K_{vert}^{(\mathbf{T})}$	1	1	1
CTA compressed data (MB)	13.18	15.89	10.37
Squared error ratio (SRBF)	1.41%	1.28%	1.23%
Squared error ratio (SRBF + CTA)	3.75%	3.64%	3.91%
PRT computation time (hr.)	0.94	1.12	0.75
CTA compression time (hr.)	12.71	14.19	10.47
Frames per second (view/light)	99.81/19.47	96.43/16.93	129.07/19.64

ing point numbers reduces the raw PRT data by 94.7%, and CTA further compresses the PRT data in SRBFs by a factor of 73 on average. The total compression ratio is thus beyond 1000:1.

Figures 9.3 and 9.4 demonstrate the rendered images based on the proposed approach. In all rendered results, a diffuse plane was placed under each model to demonstrate the ability of univariate SRBFs to handle all-frequency shadows. The uniform univariate SRBF representation typically provides visually pleasing shadows with only hundreds of univariate SRBFs. Although occasionally the contours of shadow boundaries appear to be too smooth so that some sharp features of shadowing objects are lost, a set of denser univariate SRBFs can always be adopted to faithfully model the raw PRT data at the cost of more compression time and slightly slower rendering performance when the lighting environment rotates.

Table 9.2 and Figure 9.5 compare the proposed algorithm with all-frequency *clustered principal component analysis* (CPCA) [105]. While the proposed approach renders objects with comparable quality in real time, the performance of all-frequency CPCA is only at interactive rates. There are two main reasons that enable the proposed approach to achieve real-time performance. First, we adopt univariate SRBFs to approximate radiance transfer functions with much fewer coefficients. Second, to accelerate compression time and allow CPCA adaptive to the large dimension of the illumination mode, all-frequency CPCA statically partitions the illumination mode into several segments. Nevertheless, while performing per-vertex operations on GPUs at run-time, objects should be rendered several times for each segment separately. Memory bandwidths are consumed for transferring the partially reconstructed data of each segment to GPUs. By contrast, the proposed approach does not partition the illumination mode, since

Table 9.2: Comparisons of statistics and timing measurements between the proposed all-frequency PRT algorithm and all-frequency CPCA [105] for the model *Buddha*. The raw PRT data were obtained by setting the BRDF terms to 16. In the row *Projection coefficients*, we list the mode-$vert$ reduced rank $R^{(\mathbf{T})}_{vert}$ for the proposed approach, which can be regarded as analogous to the projection coefficients for all-frequency CPCA. In the row *Frames per second*, we also show the rendering performance when the viewpoint or the lighting environment changes.

Algorithm	All-frequency CPCA [105]	Proposed approach
Light basis dimensions	0	642×48
Vertex basis dimensions	256×8	48×8
Projection coefficients	12	24
Clusters	140×24	140
Compressed data (MB)	48.41	13.18
Squared error ratio	3.09%	3.75%
Frames per second (view/light)	18.53/2.38	99.81/19.47

the dimension of this mode is usually not so large. Instead, we increase the reduced rank of the vertex mode to decrease approximation errors. Therefore, objects can be rendered on GPUs at real-time rates, without much overhead.

The main off-line cost of the proposed approach lies in the clustering stage of CTA, since we search for a locally optimal solution to the decomposed results of each cluster. Other heuristic clustering approaches can be applied to decrease the compression time, while increasing the number of clusters or the values of reduced ranks to obtain similar image quality. However, this scheme generally leads to lower compression ratios as well as slower run-time performance. We intend to reduce the compression time, or develop heuristic error metrics to approximate a locally optimal solution in the future.

9.1.4 Discussions

Compared with previous all-frequency PRT algorithms, the advantages of the proposed approach are summarized as follows:

- The proposed method is able to render all-frequency effects from compact compressed data at real-time rates.
- With the univariate SRBF representation, radiance functions can be modeled in their intrinsic domain to avoid false boundaries, distortions, and unnecessary parameterization.
- The univariate SRBFs behave as noise filters to prevent aliasing artifacts, and usually lead to visually pleasing results.
- CTA can further reduce the light mode to exploit the data coherence that can not be found by all-frequency CPCA.

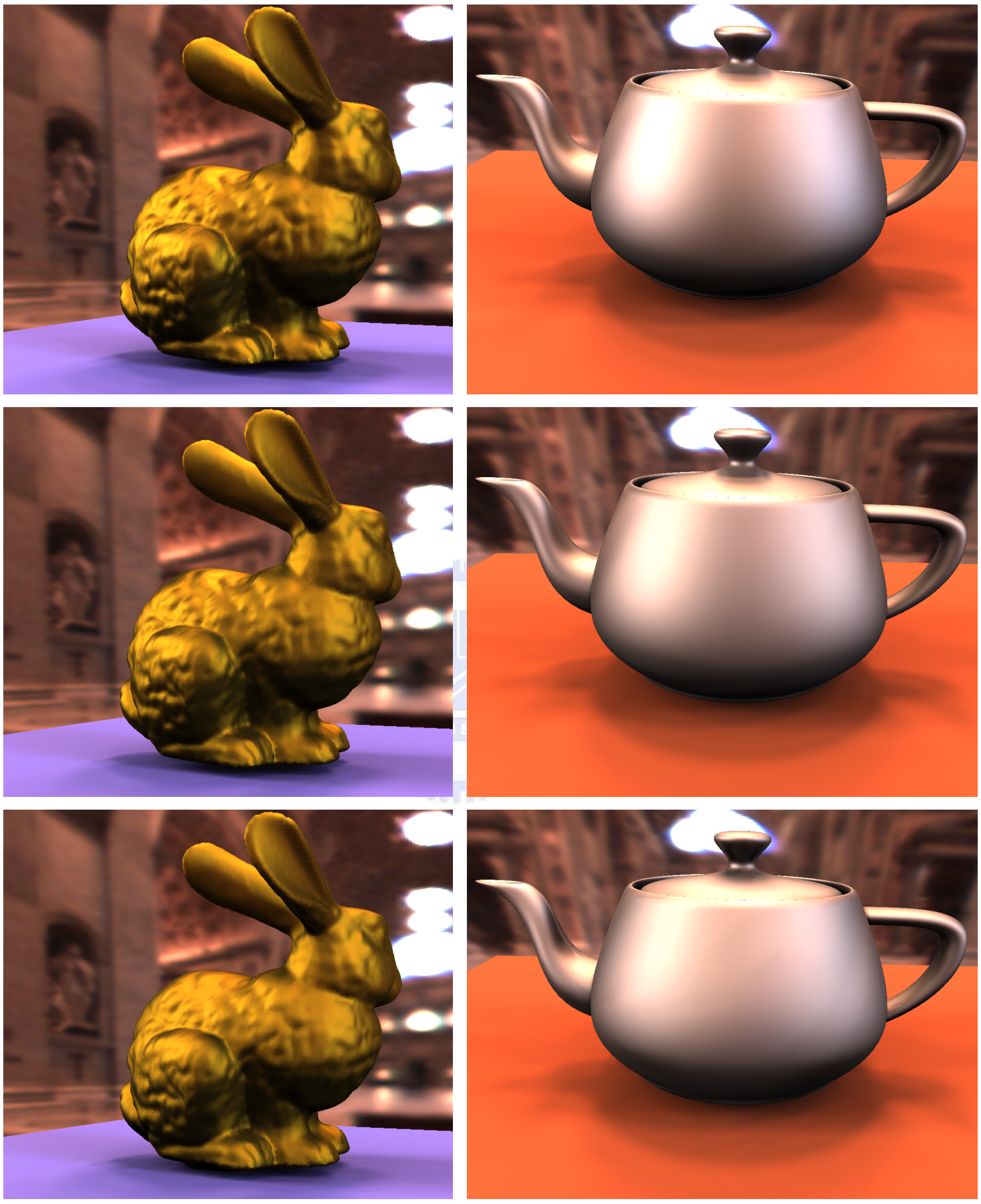
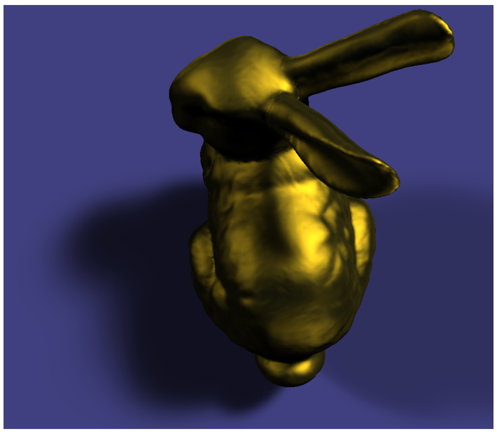
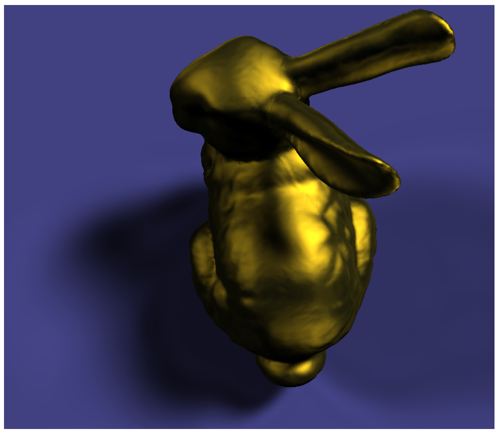
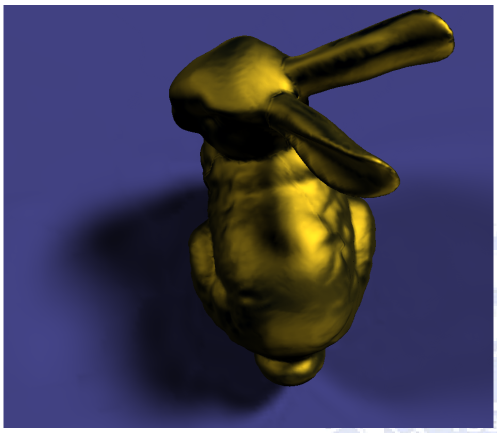
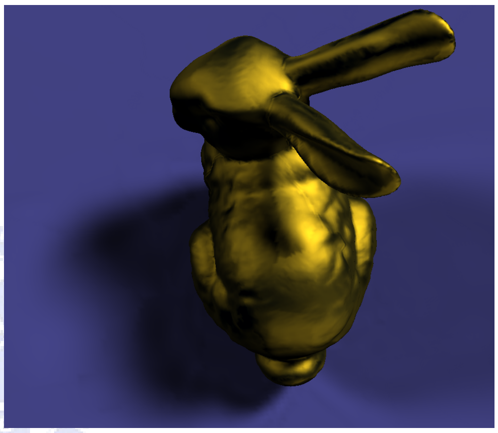

(a) *Bunny* (b) *Teapot*

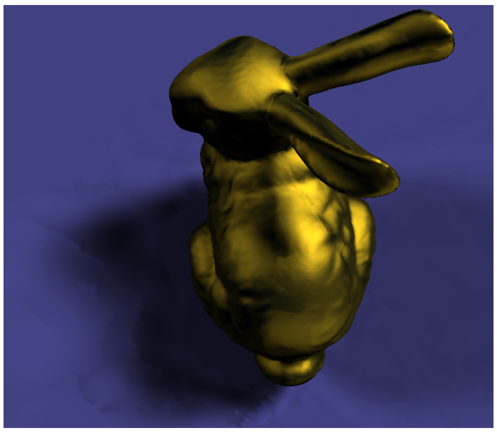
Figure 9.3: Rendered images based on the proposed all-frequency PRT algorithm. From top to bottom: raw PRT data; compressed PRT data (SRBF); compressed PRT data (SRBF + CTA). For algorithm configurations, please refer to Table 9.1.


(a) Raw PRT data


(b) PRT data in univariate SRBF expansions


(c) $R^{(\mathbf{T})}_{\omega_l} = 48$, $R^{(\mathbf{T})}_{vert} = 24$
(15.89 MB, 91.34/14.73 FPS)


(d) $R^{(\mathbf{T})}_{\omega_l} = 24$, $R^{(\mathbf{T})}_{vert} = 24$
(9.40 MB, 92.63/19.92 FPS)


(e) $R^{(\mathbf{T})}_{\omega_l} = 48$, $R^{(\mathbf{T})}_{vert} = 12$
(13.00 MB, 99.86/15.25 FPS)

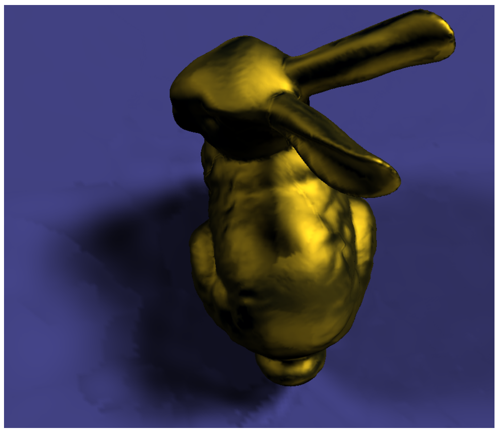
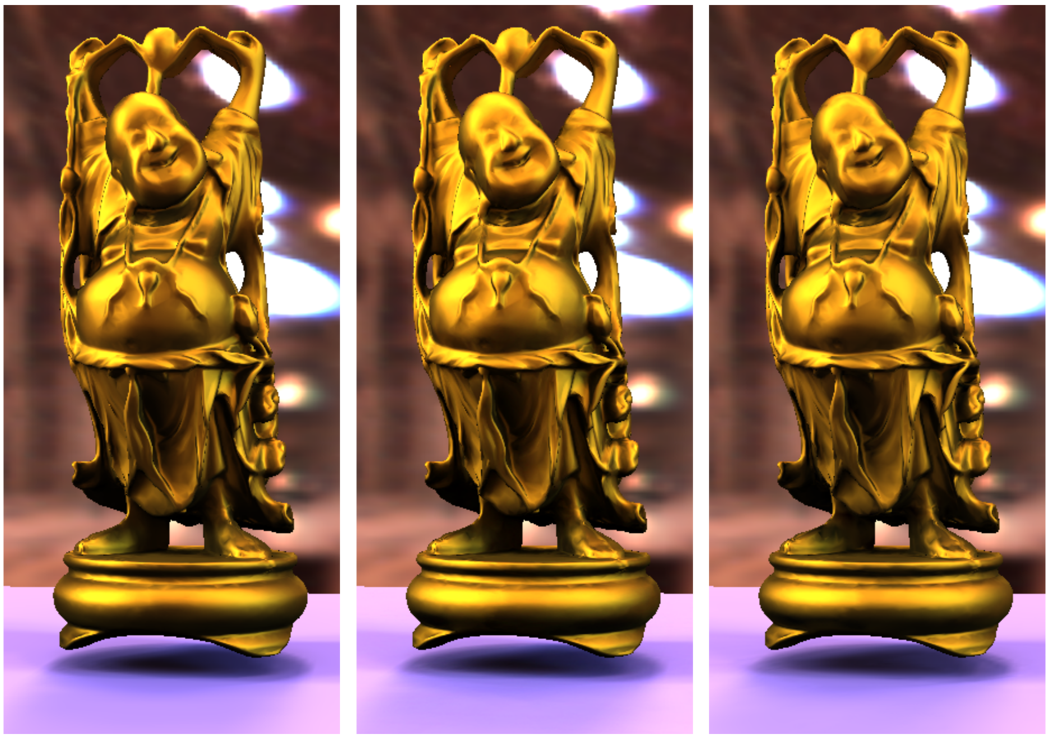

(f) $R^{(\mathbf{T})}_{\omega_l} = 24$, $R^{(\mathbf{T})}_{vert} = 12$
(7.26 MB, 100.79/20.48 FPS)

Figure 9.4: Comparisons of rendered images with various configurations of CTA based on the proposed all-frequency PRT algorithm for the model *Bunny*. The amount of compressed data and rendering performance in *frames per second* (FPS) are shown in parentheses. For rendering performance, we list the frame rate when the viewpoint or the lighting environment changes (view/light). For algorithm configurations other than $R^{(\mathbf{T})}_{\omega_l}$ and $R^{(\mathbf{T})}_{vert}$, please refer to Table 9.1.



(a) Raw PRT data (b) All-frequency CPCA (18.53 frames per second) (c) Proposed approach (99.81 frames per second)

Figure 9.5: Comparison of rendered images between the proposed all-frequency PRT algorithm and all-frequency CPCA [105] for the model *Buddha*. The lighting environment adopted in the result of all-frequency CPCA was approximated with 324 basis functions (total 972 coefficients) using the area-weighted Haar wavelet method [125], whereas the proposed approach modeled the environment with 162 scattered univariate SRBFs (total 972 parameters). Note that the proposed approach achieves real-time performance with image quality comparable to all-frequency CPCA. For algorithm configurations, please refer to Tables 9.1 and 9.2.

When the lighting environment changes, CTA particularly allows efficient computation of Step 2 (Section 9.1.2 on Page 132), whose computational cost roughly depends on $I_{\omega_l}^{(\mathbf{T})} \times R_{\omega_l}^{(\mathbf{T})}$ and $R_{\omega_v}^{(\mathbf{T})} \times R_{\omega_l}^{(\mathbf{T})} \times R_{vert}^{(\mathbf{T})}$. For all-frequency CPCA [105], the lighting environment is convoluted with the vertex basis matrices of each cluster. Therefore, the computational cost instead depends on $I_{\omega_v}^{(\mathbf{T})} \times I_{\omega_l}^{(\mathbf{T})} \times R_{vert}^{(\mathbf{T})}$, which is more expensive than the proposed approach.

There are also some disadvantages of the proposed all-frequency PRT algorithm:

- The off-line computational costs are higher than previous PRT algorithms.
- Unlike zonal harmonics [164], per-vertex rotations of radiance transfer functions currently can not be efficiently handled.
- After modeling radiance transfer functions with uniform univariate SRBFs, some sharp features, such as all-frequency shadows, may be slightly more blurred than wavelet-based methods.
- For highly specular BRDFs, such as all-frequency mirroring effects, the technique pro-

posed by Green et al. [50] may be better than the proposed approach.

- The proposed method does not support fully dynamic changes of the lighting environment at real-time rates. The lighting environment is only allowed to rotate efficiently at run-time.

However, in Section 7.1, we showed that the scattered univariate SRBF representation can outperform the area-weighted Haar wavelets [125] for modeling high dynamic range environment maps. If the scattered univariate SRBF representation is applied to model the radiance transfer function of each object vertex, the second and third disadvantages may be overwhelmed. Our current scheme is merely a trade-off between approximation errors and computational costs. As for highly specular BRDFs, the technique of Green et al. [50] and the proposed approach could be integrated as a two-pass rendering process to overcome the fourth disadvantage.

9.2 Bi-Scale Radiance Transfer

Although PRT can capture realistic global illumination effects for efficient run-time rendering at real-time rates, it only considers spatially-uniform surface appearance. BRT [163] thus generalizes PRT with spatially-varying materials, which is called radiance transfer textures, in low-frequency lighting environments to improve image quality with detailed surface appearance. The main concept of BRT is to separate the light transport problem into macro-scale radiance transfer (coarsely-sampled global illumination data) and meso-scale radiance transfer (spatially-varying appearance models). This not only enriches PRT with illumination- and view-dependent meso-textures, but also prevents full PRT simulation all over the object surfaces at the meso-scale. Nevertheless, previous BRT algorithms are limited to low-frequency light transport and bump-mapping-like surface appearance. The radiance transfer textures only model fine-scale lighting and shadowing effects, but neither contain shape information nor actually modify the surface geometry. Therefore, meso-scale shape details and shadow boundaries owing to complex meso-structures can not be faithfully captured.

In this dissertation, we address the problem of all-frequency BRT with spatially-varying reflectance and complex meso-structures. The proposed approach combines *bidirectional texture functions* (BTFs) and *view-dependent occlusion texture functions* (VOTFs) for meso-scale radiance transfer to model not only spatially-varying illumination effects but also view-dependent occlusions at the meso-scale. To obtain an accurate and compact representation, we apply tensor approximation algorithms to decompose BTFs and VOTFs into a few low-order factors and the reduced multi-dimensional core tensor(s).

As for macro-scale radiance transfer, a solution to global illumination based on all-frequency PRT is precomputed. The low-order factors extracted by tensor approximation algorithms especially facilitate the computation of the radiance transfer matrix at each object vertex. Moreover,

Table 9.3: Feature comparisons of the proposed all-frequency BRT algorithm with various meso-scale appearance models, including *bidirectional texture functions* (BTFs) [28], *generalized displacement maps* (GDMs) [197], *relief mapping* (RM) [138, 139], *radiance transfer textures* (RTTs) [163], *radiance transfer volume* (RTV) [197], *shell radiance texture functions* (SRTFs) [166], *shell texture functions* (STFs) [21], and *view-dependent displacement mapping* (VDM) [194].

Appearance models	BTFs	GDMs	Proposed BRT	RM	RTTs	RTV	SRTFs	STFs	VDM
All-frequency	$\surd$	$\surd$	$\surd$	$\surd$				$\surd$	$\surd$
Bi-scale transfer			$\surd$		$\surd$	$\surd$	$\surd$	$\surd$	
Complex geometry	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	
Inter-reflections	$\surd$		$\surd$		$\surd$		$\surd$	$\surd$	
Meso-scale occlusions		$\surd$	$\surd$	$\surd$		$\surd$		$\surd^*$	$\surd$
Meso-structure synthesis	$\surd$	$\dagger$	$\surd$	$\dagger$	$\surd$	$\dagger$	$\surd$	$\surd$	$\dagger$
Real-time rendering	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$		$\surd$
Shadows	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Sub-surface scattering	$\surd^\ddagger$		$\surd$				$\surd$	$\surd$	

*For shell texture functions, meso-scale occlusions are determined by the ray tracing process at run-time.

†To the best of our knowledge, although it is possible to extend these appearance models for meso-structure synthesis, no articles have conducted complete experiments on this issue.

‡One of the acquisition methods for spatially-varying sub-surface scattering materials can be found in [135].

we also employ the meso-scale occlusion information in VOTFs to cast shadows due to meso-structures.

Table 9.3 lists the feature comparisons of the proposed all-frequency BRT algorithm with several meso-scale appearance models. It can be shown from this table that the proposed BRT algorithm supports most features, including sub-surface scattering if the utilized BTFs contain this effect.

9.2.1 Problem Formulation

Consider a static scene, distant illumination, and spatially-varying appearance models. The rendering equation [73] over the unit sphere $\mathbb{S}^2$ can be rewritten as

$$L_{o,\mathbf{p}}(\omega_v) = \int_{\mathbb{S}^2} \left(L_{o,\mathbf{p}}^{(D)}(\omega_v) + L_{o,\mathbf{p}}^{(I)}(\omega_v) \right) |\omega_l \cdot \mathbf{n}_\mathbf{p}| \, d\omega_l, \tag{9.15}$$

$$L_{o,\mathbf{p}}^{(D)}(\omega_v) = L_{in}(\omega_l) B_\mathbf{p}(\omega_l, \omega_v, \mathbf{t}_\mathbf{p}) \prod_{\mathbf{q} \in Q_{\mathbf{p},\omega_l}} O_\mathbf{q}(-\omega_l, \mathbf{t}_\mathbf{q}), \tag{9.16}$$

$$L_{o,\mathbf{p}}^{(I)}(\omega_v) = L_{o,\mathbf{q}_{\mathbf{p},\omega_l}^{(h)}}(-\omega_l) B_\mathbf{p}(\omega_l, \omega_v, \mathbf{t}_\mathbf{p}) \Big(1 - \prod_{\mathbf{q} \in Q_{\mathbf{p},\omega_l}} O_\mathbf{q}(-\omega_l, \mathbf{t}_\mathbf{q}) \Big), \tag{9.17}$$

where $L_{o,\mathbf{p}}(\omega_v) \in \mathbb{R}$ denotes the exitant radiance from a sample point $\mathbf{p} \in \mathbb{R}^3$ on an object surface in view direction $\omega_v \in \mathbb{S}^2$, $L_{o,\mathbf{p}}^{(D)}(\omega_v) \in \mathbb{R}$ and $L_{o,\mathbf{p}}^{(I)}(\omega_v) \in \mathbb{R}$ are respectively known as the

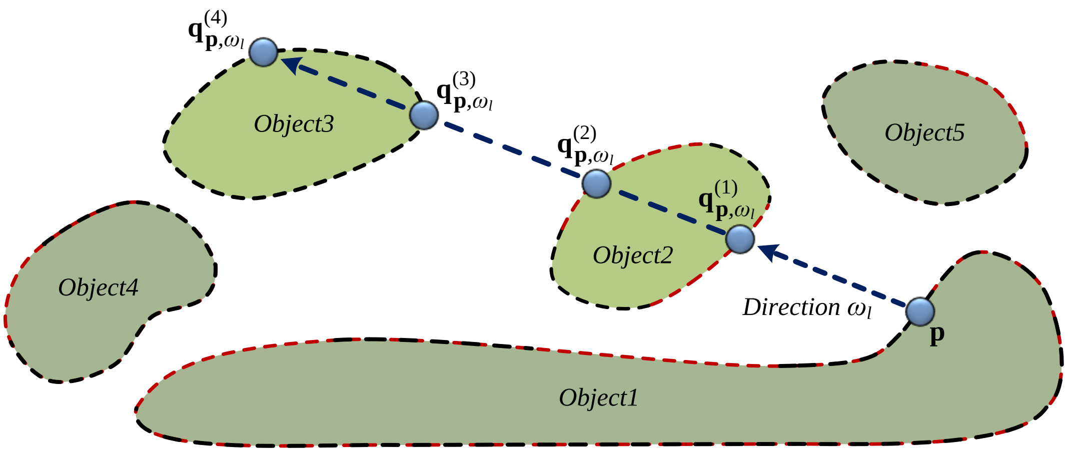

$\mathbf{q}_{\mathbf{p},\omega_l}^{(4)}$
$\mathbf{q}_{\mathbf{p},\omega_l}^{(3)}$
Object3
$\mathbf{q}_{\mathbf{p},\omega_l}^{(2)}$
$\mathbf{q}_{\mathbf{p},\omega_l}^{(1)}$
Object5
Object4
Object2
Direction ω_l
$\mathbf{p}$
Object1

Figure 9.6: The proposed all-frequency BRT algorithm determines visibility values at the meso-scale. Black solid lines indicate true object surfaces. Red dotted lines instead show that there are object surfaces at the macro-scale, but no occlusions at the meso-scale. From a surface point $\mathbf{p}$ in direction ω_l, the true intersection point thus is not $\mathbf{q}_{\mathbf{p},\omega_l}^{(1)}$ but $\mathbf{q}_{\mathbf{p},\omega_l}^{(3)}$, which is the first point whose exitant ray in direction $-\omega_l$ is occluded at the meso-scale.

direct and indirect illumination terms of exitant radiance, $L_{in}(\omega_l) \in \mathbb{R}$ represents the incident radiance from distant illumination direction $\omega_l \in \mathbb{S}^2$, and $\mathbf{n}_\mathbf{p} \in \mathbb{S}^2$ specifies the surface normal at point $\mathbf{p}$. The BTF $B_\mathbf{p}(\omega_l, \omega_v, \mathbf{t}_\mathbf{p}) \in \mathbb{R}$ and the VOTF $O_\mathbf{p}(\omega_v, \mathbf{t}_\mathbf{p}) \in \{x \in \mathbb{R} \mid 0 \le x \le 1\}$ are the meso-scale reflectance distributions and view-dependent occlusions at point $\mathbf{p}$, where the spatial coordinates of point $\mathbf{p}$, denoted by $\mathbf{t}_\mathbf{p}$, are obtained from the meso-structure synthesis. Moreover, $Q_{\mathbf{p},\omega_l} = \{\mathbf{q}_{\mathbf{p},\omega_l}^{(1)}, \ldots, \mathbf{q}_{\mathbf{p},\omega_l}^{(h)}\}$ represents the set of intersection points from point $\mathbf{p}$ in direction ω_l. The elements of $Q_{\mathbf{p},\omega_l}$ are ordered according to their Euclidean distances to point $\mathbf{p}$, from the nearest hit point $\mathbf{q}_{\mathbf{p},\omega_l}^{(1)}$ to the first point $\mathbf{q}_{\mathbf{p},\omega_l}^{(h)}$ whose exitant ray in direction $-\omega_l$ is occluded at the meso-scale (Figure 9.6). Note that visibility values are determined at the meso-scale, which is different from previous PRT methods that only model macro-scale visibility.

To obtain a compact representation for meso-scale radiance transfer, $B_\mathbf{p}(\omega_l, \omega_v, \mathbf{t}_\mathbf{p})$ and $O_\mathbf{p}(\omega_v, \mathbf{t}_\mathbf{p})$ are respectively organized as a fourth order tensor $\boldsymbol{\mathcal{A}}^{(B_\mathbf{p})} \in \mathbb{R}^{I_{\omega_l}^{(B_\mathbf{p})} \times I_{\omega_v}^{(B_\mathbf{p})} \times I_x^{(B_\mathbf{p})} \times I_y^{(B_\mathbf{p})}}$ and a third order tensor $\boldsymbol{\mathcal{A}}^{(O_\mathbf{p})} \in \mathbb{R}^{I_{\omega_v}^{(O_\mathbf{p})} \times I_x^{(O_\mathbf{p})} \times I_y^{(O_\mathbf{p})}}$ for compression using various tensor approximation algorithms as

N-SVD:

$$\boldsymbol{\mathcal{A}}^{(B_\mathbf{p})} \approx \hat{\boldsymbol{\mathcal{A}}}^{(B_\mathbf{p})} = \boldsymbol{\mathcal{Z}}^{(B_\mathbf{p})} \times_{\omega_l} \mathbf{U}_{\omega_l}^{(B_\mathbf{p})} \times_{\omega_v} \mathbf{U}_{\omega_v}^{(B_\mathbf{p})} \times_x \mathbf{U}_x^{(B_\mathbf{p})} \times_y \mathbf{U}_y^{(B_\mathbf{p})}, \tag{9.18}$$

$$\boldsymbol{\mathcal{A}}^{(O_\mathbf{p})} \approx \hat{\boldsymbol{\mathcal{A}}}^{(O_\mathbf{p})} = \boldsymbol{\mathcal{Z}}^{(O_\mathbf{p})} \times_{\omega_v} \mathbf{U}_{\omega_v}^{(O_\mathbf{p})} \times_x \mathbf{U}_x^{(O_\mathbf{p})} \times_y \mathbf{U}_y^{(O_\mathbf{p})}, \tag{9.19}$$

CTA or K-CTA:

$$\boldsymbol{\mathcal{A}}^{(B_\mathbf{p})} \approx \hat{\boldsymbol{\mathcal{A}}}^{(B_\mathbf{p})} = \sum_{c=1}^{C^{(B_\mathbf{p})}} \left(\boldsymbol{\mathcal{Z}}_c^{(B_\mathbf{p})} \times_{\omega_l} \mathbf{U}_{\omega_l,c}^{(B_\mathbf{p})} \times_{\omega_v} \mathbf{U}_{\omega_v,c}^{(B_\mathbf{p})} \times_x \mathbf{U}_{x,c}^{(B_\mathbf{p})} \times_y \mathbf{U}_{y,c}^{(B_\mathbf{p})} \right), \tag{9.20}$$

$$\boldsymbol{\mathcal{A}}^{(O_{\mathbf{p}})} \approx \hat{\boldsymbol{\mathcal{A}}}^{(O_{\mathbf{p}})} = \sum_{c=1}^{C^{(O_{\mathbf{p}})}} \left(\boldsymbol{\mathcal{Z}}_c^{(O_{\mathbf{p}})} \times_{\omega_v} \mathbf{U}_{\omega_v,c}^{(O_{\mathbf{p}})} \times_x \mathbf{U}_{x,c}^{(O_{\mathbf{p}})} \times_y \mathbf{U}_{y,c}^{(O_{\mathbf{p}})} \right). \tag{9.21}$$

Here, we omit a detailed description of the notation in Equations 9.18–9.21. As described in Sections 8.1.1 and 8.2.1, readers can refer to Equations 8.1, 8.2, 8.14, and 8.15 for the mathematical notation.

Equations 9.18 and 9.20 can be further rewritten in matrix forms as

N-SVD:

$$uf_{\omega_l}\Big(\hat{\boldsymbol{\mathcal{A}}}^{(B_{\mathbf{p}})}\Big)^T = uf_{\omega_l}\Big(\boldsymbol{\mathcal{Z}}^{(B_{\mathbf{p}})} \times_{\omega_v} \mathbf{U}_{\omega_v}^{(B_{\mathbf{p}})} \times_x \mathbf{U}_x^{(B_{\mathbf{p}})} \times_y \mathbf{U}_y^{(B_{\mathbf{p}})}\Big)^T \mathbf{U}_{\omega_l}^T, \tag{9.22}$$

CTA or K-CTA:

$$uf_{\omega_l}\Big(\hat{\boldsymbol{\mathcal{A}}}^{(B_{\mathbf{p}})}\Big)^T = \sum_{c=1}^{C^{(B_{\mathbf{p}})}} uf_{\omega_l}\Big(\boldsymbol{\mathcal{Z}}_c^{(B_{\mathbf{p}})} \times_{\omega_v} \mathbf{U}_{\omega_v,c}^{(B_{\mathbf{p}})} \times_x \mathbf{U}_{x,c}^{(B_{\mathbf{p}})} \times_y \mathbf{U}_{y,c}^{(B_{\mathbf{p}})}\Big)^T \mathbf{U}_{\omega_l,c}^T. \tag{9.23}$$

From Equations 9.22 and 9.23, it is easy to identify that mode-ω_l basis matrices resemble the illumination-dependent basis functions in BRDF factorization [105, 196]. By following the same approach as in the derivation of Equation 9.8, Equation 9.15 then becomes

N-SVD:

$$\mathbf{l}_{o,\mathbf{p}} \approx uf_{\omega_l}\Big(\boldsymbol{\mathcal{Z}}^{(B_{\mathbf{p}})} \times_{\omega_v} \mathbf{U}_{\omega_v}^{(B_{\mathbf{p}})} \times_x \mathbf{U}_x^{(B_{\mathbf{p}})} \times_y \mathbf{U}_y^{(B_{\mathbf{p}})}\Big)^T \mathbf{T}_{\mathbf{p}}\mathbf{A}_{\mathbf{p}}\mathbf{l}_{in}, \tag{9.24}$$

CTA or K-CTA:

$$\mathbf{l}_{o,\mathbf{p}} \approx \sum_{c=1}^{C^{(B_{\mathbf{p}})}} uf_{\omega_l}\Big(\boldsymbol{\mathcal{Z}}^{(B_{\mathbf{p}})} \times_{\omega_v} \mathbf{U}_{\omega_v}^{(B_{\mathbf{p}})} \times_x \mathbf{U}_x^{(B_{\mathbf{p}})} \times_y \mathbf{U}_y^{(B_{\mathbf{p}})}\Big)^T \mathbf{T}_{\mathbf{p},c}\mathbf{A}_{\mathbf{p}}\mathbf{l}_{in}, \tag{9.25}$$

where $\mathbf{l}_{o,\mathbf{p}}$ denotes the exitance radiance vector at point $\mathbf{p}$, $\mathbf{T}_{\mathbf{p}}$ (or $\mathbf{T}_{\mathbf{p},c}$) denotes the radiance transfer matrix at point $\mathbf{p}$ (for cluster c), and the interpolation matrix $\mathbf{A}_{\mathbf{p}}$ as well as the incident illumination vector $\mathbf{l}_{in}$ are respectively defined in Equations 9.10 and 9.11.

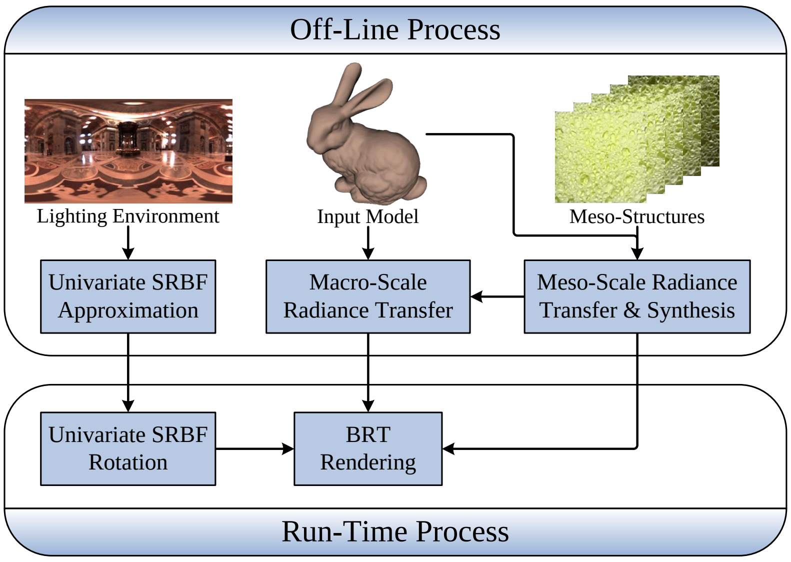
From [196], $\mathbf{T}_{\mathbf{p}}$ (or $\mathbf{T}_{\mathbf{p},c}$) can be further modeled with the sum of direct radiance transfer matrix $\mathbf{T}_{\mathbf{p}}^{(D)}$ (or $\mathbf{T}_{\mathbf{p},c}^{(D)}$) and an infinite series of j-bounce radiance transfer matrices $\{\mathbf{T}_{\mathbf{p}}^{(j)}\}_{j=1}^{\infty}$ (or $\{\mathbf{T}_{\mathbf{p},c}^{(j)}\}_{j=1}^{\infty}$) as

N-SVD:

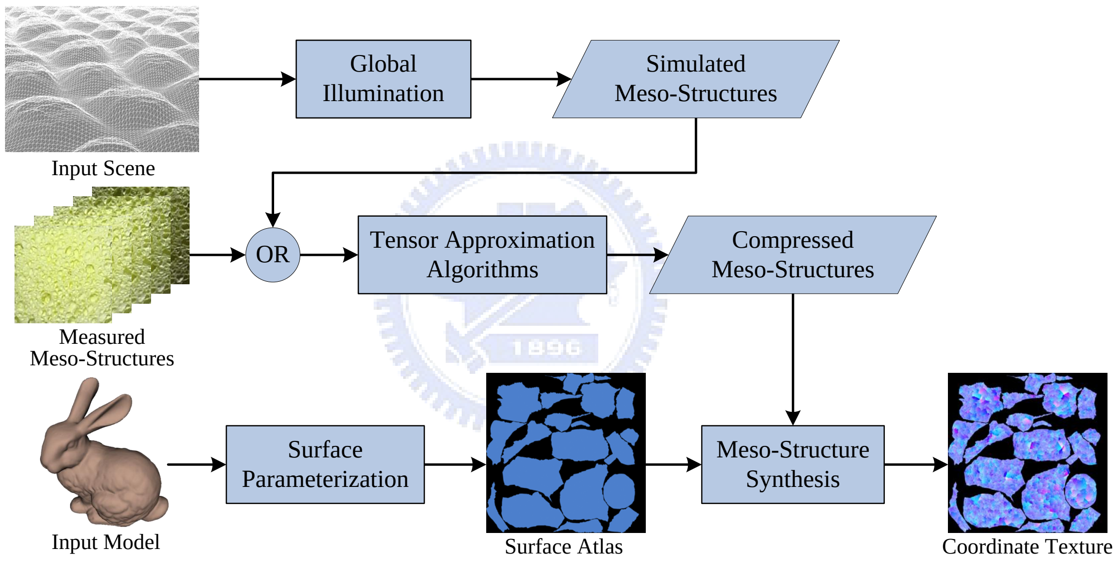
$$\mathbf{T}_{\mathbf{p}} = \mathbf{T}_{\mathbf{p}}^{(D)} + \mathbf{T}_{\mathbf{p}}^{(1)} + \mathbf{T}_{\mathbf{p}}^{(2)} + \cdots, \tag{9.26}$$

CTA or K-CTA:

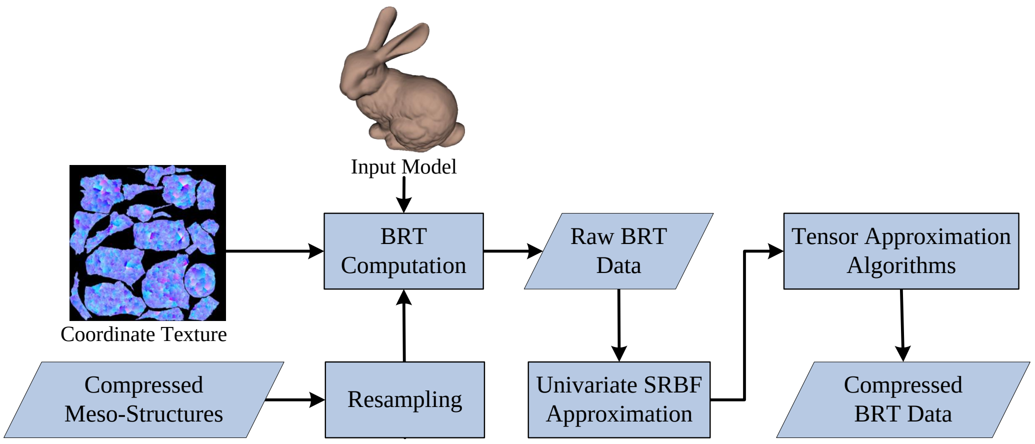
$$\mathbf{T}_{\mathbf{p},c} = \mathbf{T}_{\mathbf{p},c}^{(D)} + \mathbf{T}_{\mathbf{p},c}^{(1)} + \mathbf{T}_{\mathbf{p},c}^{(2)} + \cdots. \tag{9.27}$$



(a) System overview



(b) Meso-scale radiance transfer and synthesis



Input Model
Coordinate Texture
BRT Computation
Raw BRT Data
Tensor Approximation Algorithms
Compressed Meso-Structures
Resampling
Univariate SRBF Approximation
Compressed BRT Data

(c) Macro-scale radiance transfer

Figure 9.7: System diagrams of the proposed all-frequency BRT algorithm.

9.2.2 Algorithm

Overview

As illustrated in Figure 9.7(a), the proposed all-frequency BRT algorithm consists of off-line and run-time processes. In the off-line process, the surface meso-structures, namely BTFs and VOTFs, are either simulated by rendering three-dimensional scenes using global illumination techniques or measured from real-world objects. After that, the meso-structures are compressed using tensor approximation algorithms and then synthesized over object surfaces, as illustrated in Figure 9.7(b). For macro-scale radiance transfer (Figure 9.7(c)), the all-frequency PRT algorithm (Section 9.1) is extended to derive a global illumination solution that considers geometry details at both scales. In the run-time process, the compressed meso-structures and macro-scale BRT data, along with the synthesized results, are combined to render objects in all-frequency lighting environments.

Off-Line Process

Meso-scale radiance transfer: To reduce storage space, the meso-scale surface appearance models, including BTFs and VOTFs, are compressed using tensor approximation algorithms. For VOTFs, view-dependent signed-distance functions are recommended instead of binary visibility masks to preserve sharp geometric features at the meso-scale (Section 8.2.1). After compressing BTFs and VOTFs, we also employ the proposed resampling and meso-structure synthesis methods as described in Section 8.1.1 to prepare necessary data for macro-scale radiance transfer and run-time rendering.

Macro-scale radiance transfer: Based on the all-frequency PRT framework in Section 9.1, the raw radiance transfer matrices are computed at each object vertex, approximated with a set of uniform univariate SRBFs, and finally compressed using CTA (or K-CTA) to exploit inter-vertex coherence. During BRT computation, the meso-scale visibility at point $\mathbf{p}$ in direction ω_v, namely $O_\mathbf{p}(\omega_v, \mathbf{t_p})$, is obtained by sampling $\hat{\boldsymbol{\mathcal{Z}}}^{(O_\mathbf{p})}$ (or $\hat{\boldsymbol{\mathcal{Z}}}_c^{(O_\mathbf{p})}$) at spatial coordinates $\mathbf{t_p}$ in direction ω_v and performing construction on GPUs, where

$$N\text{-SVD:} \qquad \hat{\boldsymbol{\mathcal{Z}}}^{(O_\mathbf{p})} = \boldsymbol{\mathcal{Z}}^{(O_\mathbf{p})} \times_x \mathbf{U}_x^{(O_\mathbf{p})} \times_y \mathbf{U}_y^{(O_\mathbf{p})}, \tag{9.28}$$

$$\text{CTA or K-CTA:} \qquad \forall c,\ \hat{\boldsymbol{\mathcal{Z}}}_c^{(O_\mathbf{p})} = \boldsymbol{\mathcal{Z}}_c^{(O_\mathbf{p})} \times_x \mathbf{U}_{x,c}^{(O_\mathbf{p})} \times_y \mathbf{U}_{y,c}^{(O_\mathbf{p})}. \tag{9.29}$$

For $\{\mathbf{T}_\mathbf{p}^{(j)}\}_{j=1}^{\infty}$ (or $\{\mathbf{T}_{\mathbf{p},c}^{(j)}\}_{j=1}^{\infty}$), a triangle index texture, a barycentric coordinate texture, and a core transfer vector texture are rendered at each vertex on GPUs to identify the information of hit points. These textures are sampled at low resolutions as suggested by Sun et al. [171] to decrease precomputation time, and then employed for the computation of $\{\mathbf{T}_\mathbf{p}^{(j)}\}_{j=1}^{\infty}$ (or $\{\mathbf{T}_{\mathbf{p},c}^{(j)}\}_{j=1}^{\infty}$) on GPUs. The core transfer vector of point $\mathbf{p}$ in direction ω_v is derived from sampling $\hat{\boldsymbol{\mathcal{Z}}}^{(B_{\mathbf{p}})}$ (or $\hat{\boldsymbol{\mathcal{Z}}}_c^{(B_{\mathbf{p}})}$) at spatial coordinates $\mathbf{t_p}$ in direction ω_v, where

$$N\text{-SVD:} \qquad \hat{\boldsymbol{\mathcal{Z}}}^{(B_{\mathbf{p}})} = \boldsymbol{\mathcal{Z}}^{(B_{\mathbf{p}})} \times_x \mathbf{U}_x^{(B_{\mathbf{p}})} \times_y \mathbf{U}_y^{(B_{\mathbf{p}})}, \tag{9.30}$$

$$\text{CTA or K-CTA:} \qquad \forall c,\ \hat{\boldsymbol{\mathcal{Z}}}_c^{(B_{\mathbf{p}})} = \boldsymbol{\mathcal{Z}}_c^{(B_{\mathbf{p}})} \times_x \mathbf{U}_{x,c}^{(B_{\mathbf{p}})} \times_y \mathbf{U}_{y,c}^{(B_{\mathbf{p}})}. \tag{9.31}$$

Similar to the off-line process of the proposed all-frequency PRT algorithm (Section 9.1.2), the radiance transfer matrices of an object are approximated with a set of uniform univariate SRBFs, and further organized as a third order tensor $\boldsymbol{\mathcal{A}}^{(\mathbf{T})} \in \mathbb{R}^{I_{\omega_v}^{(\mathbf{T})} \times I_{\omega_l}^{(\mathbf{T})} \times I_{vert}^{(\mathbf{T})}}$. Given the reduced ranks of each mode, namely $R_{\omega_v}^{(\mathbf{T})}$, $R_{\omega_l}^{(\mathbf{T})}$, and $R_{vert}^{(\mathbf{T})}$, $\boldsymbol{\mathcal{A}}^{(\mathbf{T})}$ is then compressed using CTA (or K-CTA) with total $C^{(\mathbf{T})}$ clusters for the vertex mode. Here, note that the utilized tensor approximation algorithm for macro-scale radiance transfer (CTA or K-CTA) is independent of that for meso-scale radiance transfer. One can always apply different tensor approximation algorithm to model the radiance transfer data at each scale.

Run-Time Process

The BRT rendering process is similar to the run-time process of the proposed all-frequency PRT algorithm and consists of the following steps:

1. Perform Steps 1–3 as described in the run-time process of the proposed all-frequency PRT algorithm (Section 9.1.2 on Page 132) to obtain the per-vertex radiance transfer vector $\mathbf{r_p}$.
2. In the pixel shader, sample the synthesized texture S for the BTF/VOTF spatial coordinates $\mathbf{t_p}$ of current pixel $\mathbf{p}$.
3. If the meso-structure of pixel $\mathbf{p}$ does not contain VOTF, set pixel $\mathbf{p}$ as visible and go to Step 6. Otherwise, continue to execute Step 4.
4. Perform Steps 3 and 4 as described in the run-time process of VOTFs (Section 8.2.2 on Page 121) to obtain the meso-scale visibility value $\hat{O}'_{\mathbf{p}}$. Set pixel $\mathbf{p}$ to visible if $\hat{O}'_{\mathbf{p}} > 0$.
5. If pixel $\mathbf{p}$ is visible, compute its shading color by Step 6. Otherwise, discard it.
6. Sample the texture of mode-ω_v basis matrix for all the components of current novel view direction. The shading color of pixel $\mathbf{p}$ is then given by the dot product of the sampled results and $\mathbf{r_p}$.

Here, we omit a detailed description of the above rendering steps. Readers can refer to Sections 8.1.1, 8.2.2, and 9.1.2 for more details about the rendering issues of BTFs, VOTFs, and PRT.

Table 9.4: Statistics and timing measurements of the proposed all-frequency BRT algorithm. In the row *Frames per second*, we list the rendering performance with/without visibility anti-aliasing when the viewpoint changes. For the configurations of each meso-scale material, please refer to Tables 8.1 and 8.6.

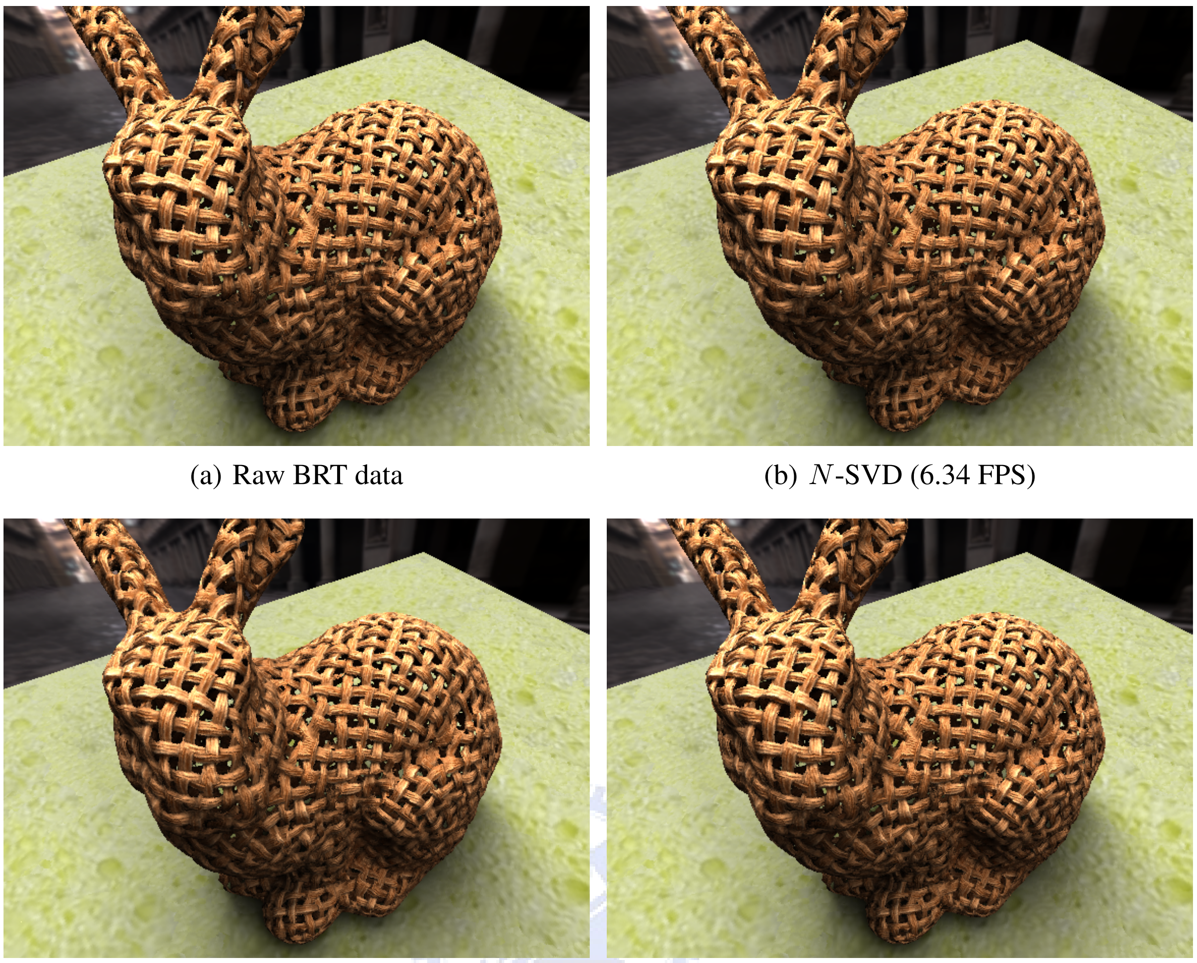
Model	Bunny	BunnyPlane	Cloth	Teapot
Material(s)	Sponge	Fiber + Sponge	Wool + Sponge	RoughHole + Sponge
Vertices: $I_{vert}^{(\mathbf{T})}$	36k	98k	55k	75k
SRBFs: $I_{\omega_l}^{(\mathbf{T})}$	642	2562	642	642
Raw BRT data (GB)	40.11	108.35	69.23	123.26
$R_{\omega_l}^{(\mathbf{T})} \times R_{vert}^{(\mathbf{T})}$	64×12	64×12	64×12	64×12
Clusters: $C^{(\mathbf{T})}$	100	180	110	150
Compressed data (MB)	15.77	71.4	18.79	29.44
BRT computation time (hr.)	2.65	22.55	7.86	10.08
CTA compression time (hr.)	8.28	16.98	12.51	13.66
Frames per second (w/wo)	-/50.45	21.38/28.13	-/31.83	19.46/24.21

9.2.3 Experimental Results

The experiments and simulation timings of the proposed all-frequency BRT algorithm were conducted and measured on a workstation with an Intel Core 2 Extreme QX6700 CPU, an NVIDIA GeForce 8800 Ultra graphics card, and 4 gigabytes main memory under Microsoft Windows Vista operating system. We employed NVIDIA CUDA [128] to accelerate the off-line process and Microsoft Direct3D 9 for run-time rendering on GPUs. In our experiments, the univariate Gaussian SRBFs were adopted to represent the radiance transfer functions and the lighting environments. The SRBF center and bandwidth sets were also constrained to be the same for red, green, and blue channels of a radiance transfer function or a lighting environment. To improve rendering performance, the super-clustering technique [161] was also applied to decrease the number of redrawn triangles.

Table 9.4 lists the experimental statistics of the proposed all-frequency BRT algorithm in various configurations. For macro-scale radiance transfer, we simulated light paths with at most two inter-reflections to precompute a 6×32×32 radiance transfer matrix at each vertex and approximated the raw BRT data with a set of uniform univariate SRBFs. Due to enormous amount of BRT data, we did not directly employ CTA (or K-CTA) but first applied clustered principal component analysis [161] to classify vertices, as suggested by Sun et al. [171], and fine-tuned cluster membership using CTA (or K-CTA). The reduction of the view mode was omitted to accelerate the compression process, and the reduced ranks of the light and vertex modes were respectively set to 64 and 12.

Figure 9.8 compares the rendered images with different tensor approximation algorithms for meso-structures based on the proposed approach. It shows that K-CTA achieves image quality comparable to N-SVD, while providing almost the same rendering performance as CTA. Al-



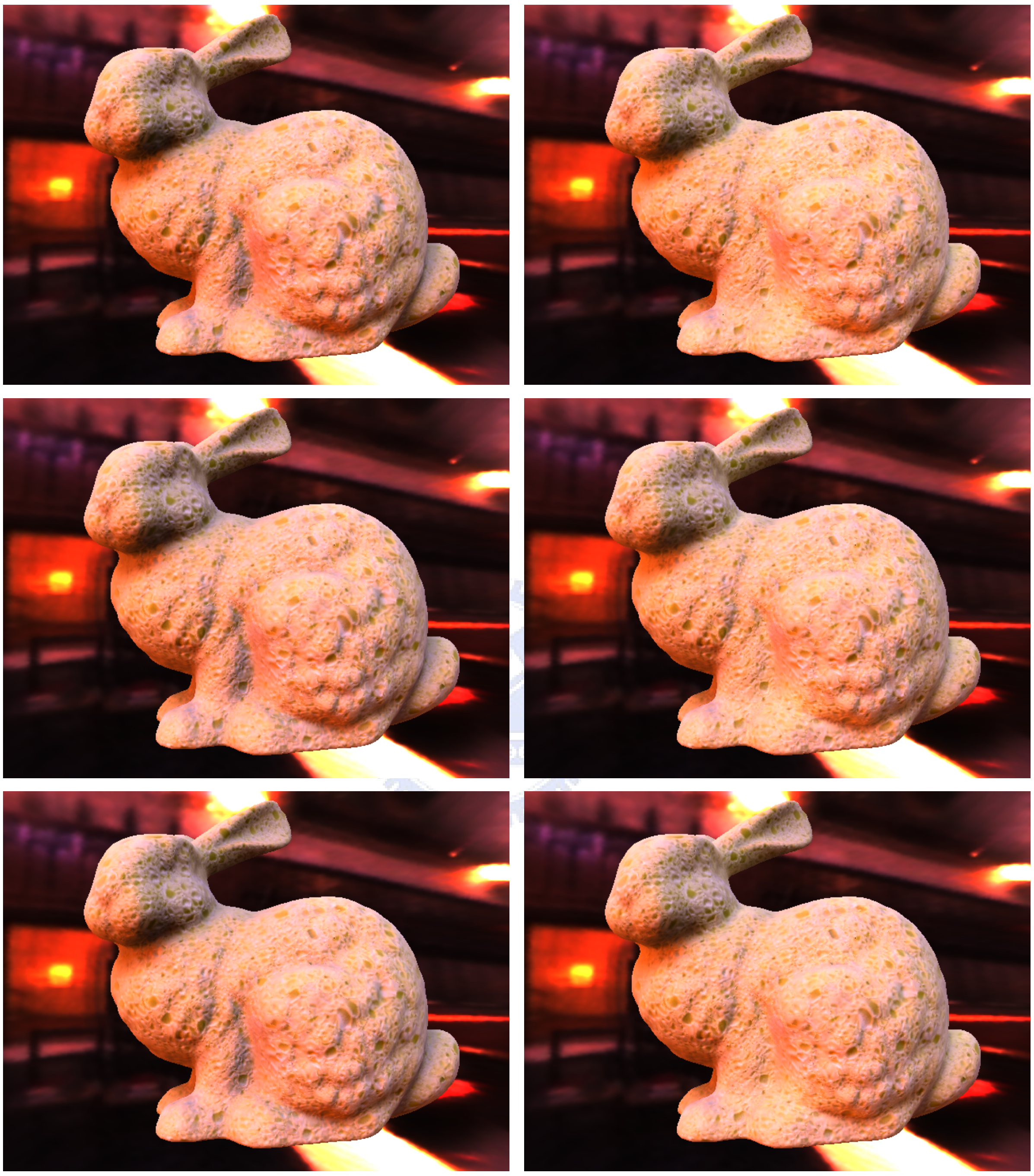
(a) Raw BRT data (b) N-SVD (6.34 FPS)

(c) CTA (22.69 FPS) (d) K-CTA (21.38 FPS)

Figure 9.8: Comparisons of rendered images with different tensor approximation algorithms for meso-structures based on the proposed all-frequency BRT algorithm for the model *BunnyPlane* with the materials *Fiber* and *Sponge*. The rendering performance in *frames per second* (FPS) are shown in parentheses. The configurations of macro-scale and meso-scale radiance transfer are listed in Tables 8.1, 8.6, and 9.4.

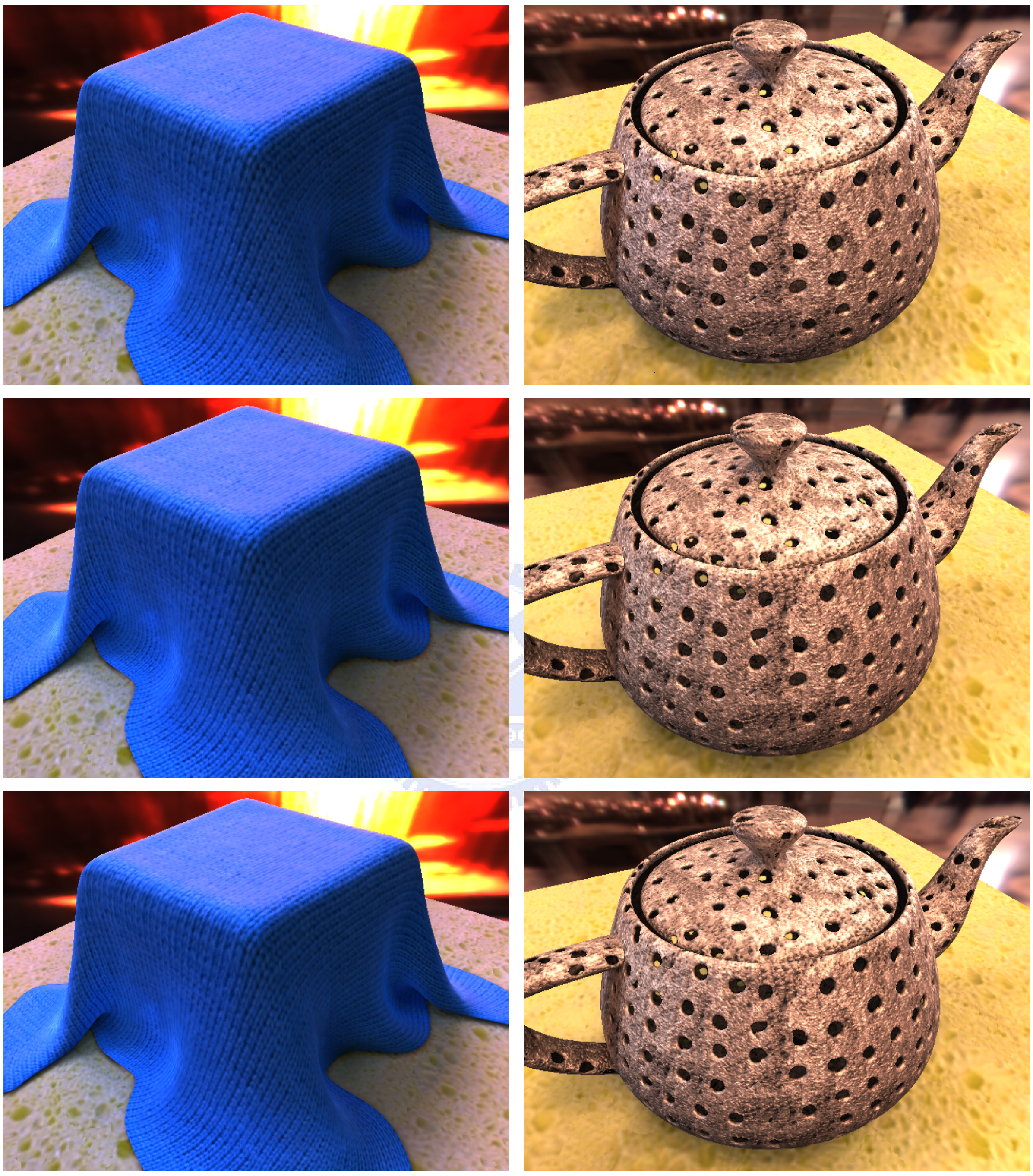
though both CTA and K-CTA allow real-time rendering performance, CTA sometimes produces noticeable seams when the viewpoint changes. This result is consistent with our experiments of BTF and VOTF compression in Chapter 8.

Figure 9.9 compares the rendered images with/without indirect illumination based on the proposed approach for the model *Bunny*. Without indirect illumination, as shown in Figure 9.9(a), inter-reflections between object surfaces can not be faithfully captured. This produces some false hard and soft shadows that result in many over dark regions, especially around the neck and forelegs of *Bunny*. In Figure 9.10, we further present more rendered images based on the proposed approach. From these images, the proposed BRT algorithm certainly provides more reflectance and geometric details of meso-scale surface appearance than previous PRT methods. Furthermore, VOTFs particularly allow rendering complex geometric features without consuming hundreds of thousands of polygons to explicitly model the surface micro-geometry.



(a) Direct illumination (b) Direct and indirect illumination

Figure 9.9: Comparisons of rendered images with/without indirect illumination based on the proposed all-frequency BRT algorithm for the model *Bunny* with the material *Sponge*. From top to bottom: raw BRT data; compressed BRT data (SRBF); compressed BRT data (SRBF + CTA). The configurations of macro-scale and meso-scale radiance transfer are listed in Tables 8.1 and 9.4.



(a) *Cloth* with *Wool* and *Sponge* (b) *Teapot* with *RoughHole* and *Sponge*

Figure 9.10: Rendered images based on the proposed all-frequency BRT algorithm. From top to bottom: raw BRT data; compressed BRT data (SRBF); compressed BRT data (SRBF + CTA). The configurations of macro-scale and meso-scale radiance transfer are listed in Tables 8.1, 8.6, and 9.4.

Chapter 10

Conclusions and Future Work

10.1 Conclusions

In this dissertation, we have introduced two new data representations, univariate and multivariate *spherical radial basis functions* (SRBFs), and two novel compression algorithms, *clustered tensor approximation* (CTA) and *K-clustered tensor approximation* (K-CTA), for real-time data-driven rendering. While SRBFs provide an intrinsic and efficient description of univariate or multivariate spherical functions, CTA and K-CTA are powerful compression algorithms that allow sophisticated data analysis, compact storage space, and fast data reconstruction. Furthermore, we have also described several important applications in computer graphics and vision. Experimental results reveal that the proposed representations and approximation algorithms can be seamlessly integrated to obtain practical solutions of photo-realistic rendering at real-time rates.

10.1.1 Spherical Radial Basis Functions

Based on univariate/multivariate SRBFs, illumination and radiance functions can be modeled in their intrinsic spherical domain to avoid artifacts that result from false boundaries, distortions, and unnecessary parameterization. Most existing methods were not originally developed to handle spherical functions and deficient in this important feature. Therefore, they frequently have to model a spherical function in an inappropriate domain rather than the unit hyper-sphere. By contrast, the proposed SRBFs naturally bear the spherical domain in mind and are indeed suitable for directional data analysis.

Furthermore, all-frequency signals can be efficiently modeled by adjusting the bandwidth parameter of a SRBF. Previous articles have demonstrated that some high-frequency features, such as sharp shadow boundaries and specular highlights, have significant contributions to human visual perception. Unfortunately, data representations based on the popular spherical harmonics frequently rely on discarding high-frequency parts to achieve compact results. This makes spherical harmonics difficult to adapt to various kinds of real-world illumination and radiance functions. On the contrary, the proposed SRBFs allow localized high-frequency signals to be easily described by their bandwidth parameters. This spatial localization property particularly distinguishes SRBFs from spherical harmonics. We thus expect that univariate and multivariate SRBFs will have a great impact on data representations for illumination and radiance functions in the future.

10.1.2 Tensor Approximation Algorithms

As for approximation algorithms, one major disadvantage of previous tensor-based approaches for real-time applications is that the decomposed results are not compact enough to be employed for efficient reconstruction on GPUs. We thus aim at overcoming this drawback by introducing the concept of sparse representation into multi-linear models, and effectively integrate clustering, sparse coding, and tensor approximation into a unified framework. As a result, the proposed CTA and K-CTA algorithms are especially appropriate for analyzing multi-dimensional data sets in real-time applications. When compared to traditional tensor approximation, the rendering performance of CTA and K-CTA is frequently improved by a substantial factor with only slight increases in approximation errors and amounts of compressed data. Moreover, K-CTA can also exploit inter-cluster coherence for smooth transitions across different physical conditions by mixing the decomposed results of multiple clusters.

At first glance, CTA and K-CTA may seem to have similarities with previous matrix factorization methods [90, 124]. Nayar et al. [124] introduced two-stage singular value decomposition to exploit inter-block coherence, but their approach does not allow sparse representations. Lawrence et al. [90] instead proposed to include sparsity penalty in the objective function so that the final results tend to be sparse. Nevertheless, the weighting parameter of sparsity penalty should be tuned for different data sets in order to obtain a pre-defined number of non-zero terms. By contrast, the proposed CTA and K-CTA algorithms permit sparse representations in which the number of non-zero terms is inherently enforced. For K-CTA, the sparsity can even be controlled by a user-defined threshold. In this way, an upper bound on run-time reconstruction costs is guaranteed. We believe that this property of CTA and K-CTA is significant for real-time applications, since run-time performance is predictable and totally under user control.

10.1.3 Summary

Additionally, we can further combine the advantages of SRBFs and CTA/K-CTA to provide more compact compressed data, while still achieving real-time rendering rates at run-time. The proposed all-frequency radiance transfer algorithms in Chapter 9 especially demonstrate that tight corporation between SRBFs and tensor approximation algorithms is possible and often favorable for practical real-time applications in computer graphics and vision. The successful radiance transfer frameworks also indicate that parametric representations, such as univariate and multivariate SRBFs, and non-parametric models, for example CTA and K-CTA, may not actually conflict with each other.

In brief, we suggest that it is often preferable to employ univariate or multivariate SRBFs for modeling spherical functions, as they usually lead to more compact storage space and faster rendering performance on GPUs. Nevertheless, a slight quality loss and long computation time are inevitable consequences. By contrast, CTA and K-CTA indeed provide a more general, flexible, and accurate data analysis tool for various kinds of multi-dimensional visual data sets. They are not restricted to a pre-defined functional form and only need efficient linear algebra computation. However, CTA and K-CTA have higher reconstruction costs and often require additional auxiliary data for efficient run-time rendering on GPUs.

10.2 Future Work

10.2.1 Flexible Data Representations

In this dissertation, we only consider univariate SRBFs that are circularly axis-symmetric. Such 'isotropic' univariate SRBFs are inefficient in modeling axis-asymmetric signals on the unit hyper-sphere. For example, we may need a lot of univariate SRBFs to accurately represent an elliptically shaped data set around an axis, but it is easy to identify that only an elliptically shaped univariate SRBF would be sufficient. In fact, this 'anisotropic' univariate SRBF can be constructed by introducing more parameters to the original isotropic univariate SRBF. Specifically, we can rely on an additional SRBF center to control the orientation and anisotropy of a univariate SRBF. Nevertheless, the impact of an extra SRBF center on approximation errors and the non-linear optimization process is still unknown. More model parameters also imply more computation time. We plan to verify and solve these practical issues of anisotropic univariate SRBFs in the future.

In the current implementation, the proposed univariate/multivariate SRBF representations only employ one type of SRBFs. An obvious question is that can we utilize more than one type of SRBFs to model observations on the unit hyper-sphere? The answer is certainly 'yes' to the question. We may achieve this simply by a weighted sum of different types of univariate/multivariate SRBFs. For a multivariate SRBF, it can even be constructed from the product of different types of univariate SRBFs. Nevertheless, the real question is that will this sophisticated approach outperform current SRBF representations? This may need intensive experiments to reach a final conclusion.

Last but not the least, although this dissertation introduces novel parametric representations to model observations on the unit hyper-sphere, our ultimate goal is to develop flexible data representations for various kinds of real-world visual data sets. The proposed univariate and multivariate SRBFs are only a small step toward this objective. Since a flexible data representation should be adaptive to real-world observations with different characteristics, we are particularly interested in an automatic model selection scheme based on univariate/multivariate SRBFs. Although no significant differences were found between the univariate Gaussian and Abel-Poisson SRBF kernels in our experiments, this outcome only exposes a small part of the real world. We intend to extensively investigate the effects of different types of univariate/multivariate SRBFs on more real-world visual data sets in the near future. As a result, one or more appropriate types of univariate/multivariate SRBFs can be automatically determined for a given visual data set.

10.2.2 Powerful Approximation Algorithms

Unlike principal component analysis, one major disadvantage of tensor approximation algorithms is that a globally optimal solution is not guaranteed. The alternating least-squares algorithm for tensor decomposition will converge to the global optimum only when the starting point is close enough to this optimum. Therefore, we aim to resolve an appropriate initial guess for approximating the globally optimal solution. Moreover, the reduced ranks are currently specified by users to control the approximation quality. We seek to develop a method to automatically determine the 'best' reduced ranks for a given multi-dimensional visual data set.

There are many similarities between linear and multi-linear models in dimensionality reduction. For example, N-mode singular value decomposition [33] can be regarded as a natural extension of singular value decomposition for multi-dimensional data sets. The similarities in fact imply that many techniques developed for linear models may be applied to multi-linear models with mathematical reformulation or some modifications. Thus, we are particularly interested in existing techniques of linear models that can be seamlessly integrated with CTA/K-CTA, or in extending CTA/K-CTA to analyze multi-dimensional visual data sets with more general or complicated structures.

Moreover, CTA and K-CTA are currently allowed to perform clustering along only one mode of an input tensor. If there are large variations in more than one mode, users should carefully choose an appropriate mode for partition, while leaving other modes unchanged. For example, a bidirectional texture function that comprises of complex meso-structures and highly specular materials would require high reduced ranks for the illumination and view modes. Therefore, both modes may have to be partitioned in order to reduce run-time reconstruction costs. This has inspired us with an interesting research direction in the future—generalizing CTA/K-CTA for clustering along multiple modes of an input tensor.

Except for Tucker models [180, 181] adopted in this dissertation, there are also other popular multi-linear models, such as parallel factor analysis [57] and canonical decomposition [18]. In the present, we have no idea about which model is the 'best' for a given multi-dimensional visual data set. Intensive experiments should be conducted in order to make a general suggestion.

In brief, the proposed CTA and K-CTA algorithms are not restricted to solve problems in computer graphics and vision. They are indeed general approximation algorithms that can be employed to analyze various multi-dimensional data sets. We believe that CTA/K-CTA can be applied to other data-driven models, such as the analysis of time-varying materials and volume data sets, and may have an impact on many fields outside computer graphics.

10.2.3 General Data-Driven Applications

In this dissertation, the proposed framework of optimized parameterization for multivariate SRBFs only supports directional variables. However, real-world visual data sets frequently depend on other types of physical factors. These non-directional factors should not be excluded from parameterization. For example, a bidirectional texture function is a complex effect under different spatial and directional conditions. We may additionally take the spatial variables into account for parameterization. In this way, the surface appearance and geometric details of meso-structures can be implicitly modeled in a unified framework by optimized parameterization. Furthermore, the proposed parameterization framework also relies on a pre-defined parametric model. Its optimality thus is built upon a specific functional form. Finding the *truly* optimal parameterizations for a given visual data set is still a challenging problem. We plan to investigate this issue using non-parametric models for multivariate SRBFs, or even extend tensor approximation algorithms to support this concept in the future.

Moreover, recall that the Legendre expansions particularly facilitate the spherical singular integral of two univariate SRBFs. An analytic solution to the convolution of two univariate Gaussian SRBFs was also derived in Section 3.5.1. This special property of univariate SRBFs is rather important to practical rendering applications in computer graphics. With this property, we can efficiently evaluate the ubiquitous rendering equation [73] (or other similar mathematical formulation for rendering) when both the appearance model and the incident illumination function are represented based on univariate SRBFs. However, many appearance models, such as bidirectional reflectance distribution functions and bidirectional texture functions, are intrinsically multivariate functions that can be represented more efficiently with multivariate SRBFs. It is unclear whether there is an efficient way to compute the spherical singular integral of a multivariate SRBF and a univariate SRBF (or two multivariate SRBFs). For the parameterized multivariate SRBF representation, this issue will become even more complicated. Therefore, we are interested in deriving an analytic or approximate solution to this problem in the future.

There are also many possible applications of the proposed data representations and approximation algorithms in computer graphics and vision. We especially aim at extending the proposed all-frequency precomputed radiance transfer algorithm to account for dynamic scenes. With some modifications, univariate SRBFs and CTA/K-CTA may be employed to achieve real-time rendering rates for dynamic radiance transfer data sets. In addition to radiance transfer, another possible research direction is to develop an importance sampling algorithm for advanced materials. Based on the proposed multivariate SRBF representations, the distributions of spatially-varying materials, such as bidirectional texture functions, may be efficiently analyzed for generating appropriate sampling directions.

In summary, general data-driven applications are one of the main purposes of this dissertation. For example, it is rather important whether artists could directly and intuitively edit the derived parameters of univariate/multivariate SRBFs or the decomposed factors of CTA/K-CTA. The results of the proposed methods also should be utilized as the input data to various statistical or machine learning problems, such as feature selection, classification, and inference. The overall effect actually corresponds to solving these problems in the parametric space of univariate/multivariate SRBFs or in the reduced tensor space. Besides data reconstruction, we thus seek to extend the proposed data representations and approximation algorithms to obtain interpretable results or support more advanced features for a variety of data-driven applications.

Bibliography

[1] E. Acar and B. Yener, "Unsupervised Multiway Data Analysis: A Literature Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 1, pp. 6–20, 2009.

[2] S. Agarwal, R. Ramamoorthi, S. J. Belongie, and H. W. Jensen, "Structured Importance Sampling of Environment Maps," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 605–612, 2003.

[3] M. Aharon, "Overcomplete Dictionaries for Sparse Representation of Signals," Technion - Israel Institute of Technology, Ph.D. dissertation, 2006.

[4] M. Aharon, M. Elad, and A. M. Bruckstein, "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[5] Z. Bai, J. W. Demmel, J. J. Dongarra, A. Ruhe, and H. A. Van Der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. Society for Industrial and Applied Mathematics, 2000.

[6] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra, "Clustering on the Unit Hypersphere Using Von Mises-Fisher Distributions," *Journal of Machine Learning Research*, vol. 6, pp. 1345–1382, 2005.

[7] D. Barber, "Machine Learning: A Probabilistic Approach," 2006. [Online]. Available: http://web4.cs.ucl.ac.uk/staff/D.Barber/courses/mlgm_epfl_book.pdf

[8] A. T. Basilevsky, *Statistical Factor Analysis and Related Methods: Theory and Applications*. Wiley Press, 1994.

[9] M. Belkin and P. Niyogi, "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering," in *Advances in Neural Information Processing Systems 14*, pp. 585–591. MIT Press, 2002.

[10] A. Ben-Artzi, K. Egan, R. Ramamoorthi, and F. Durand, "A Precomputed Polynomial Representation for Interactive BRDF Editing with Global Illumination," *ACM Transactions on Graphics*, vol. 27, no. 2, 2008.

[11] A. Ben-Artzi, R. Overbeck, and R. Ramamoorthi, “Real-Time BRDF Editing in Complex Lighting,” *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 945–954, 2006.

[12] D. Berman, J. T. Bartell, and D. Salesin, “Multiresolution Painting and Compositing,” in *Proceedings of ACM SIGGRAPH 1994*, pp. 85–90, 1994.

[13] C. M. Bishop, “Bayesian PCA,” in *Advances in Neural Information Processing Systems 11*, pp. 382–388. MIT Press, 1999.

[14] C. M. Bishop, M. Svensén, and C. K. I. Williams, “GTM: The Generative Topographic Mapping,” *Neural Computation*, vol. 10, no. 1, pp. 215–234, 1998.

[15] D. Blythe, “The Direct3D 10 System,” *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 724–734, 2006.

[16] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, “A Limited Memory Algorithm for Bound Constrained Optimization,” *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[17] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, “Reconstruction and Representation of 3D Objects with Radial Basis Functions,” in *Proceedings of ACM SIGGRAPH 2001*, pp. 67–76, 2001.

[18] J. D. Carroll and J.-J. Chang, “Analysis of Individual Differences in Multidimensional Scaling via an N-Way Generalization of ’Eckart-Younga’ Decomposition,” *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[19] S. S. Chen, D. L. Donoho, and M. A. Saunders, “Atomic Decomposition by Basis Pursuit,” *SIAM Review*, vol. 43, no. 1, pp. 129–159, 2001.

[20] W.-C. Chen, J.-Y. Bouguet, M. H. Chu, and R. Grzeszczuk, “Light Field Mapping: Efficient Representation and Hardware Rendering of Surface Light Fields,” *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 447–456, 2002.

[21] Y. Chen, X. Tong, J. Wang, S. Lin, B. Guo, and H.-Y. Shum, “Shell Texture Functions,” *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 343–353, 2004.

[22] P. Clarberg, W. Jarosz, T. Akenine-Möller, and H. W. Jensen, “Wavelet Importance Sampling: Efficiently Evaluating Products of Complex Functions,” *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 1166–1175, 2005.

[23] Computer Graphics Group, University of Bonn, “BTF Database Bonn,” 2006. [Online]. Available: http://btf.cs.uni-bonn.de/

[24] R. L. Cook, “Stochastic Sampling in Computer Graphics,” *ACM Transactions on Graphics*, vol. 5, no. 1, pp. 51–72, 1986.

[25] R. L. Cook and K. E. Torrance, “A Reflectance Model for Computer Graphics,” *ACM Transactions on Graphics*, vol. 1, no. 1, pp. 7–24, 1982.

[26] T. F. Cox and M. A. A. Cox, *Multidimensional Scaling*, 2nd ed. CRC Press, 2000.

[27] O. G. Cula and K. J. Dana, “Compact Representation of Bidirectional Texture Functions,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition 2001*, pp. 1041–1047, 2001.

[28] K. J. Dana, B. Van Ginneken, S. K. Nayar, and J. J. Koenderink, “Reflectance and Texture of Real-World Surfaces,” *ACM Transactions on Graphics*, vol. 18, no. 1, pp. 1–34, 1999.

[29] P.-E. Danielsson, “Euclidean Distance Mapping,” *Computer Graphics and Image Processing*, vol. 14, pp. 227–248, 1980.

[30] K. Daubert, J. Kautz, H.-P. Seidel, W. Heidrich, and J.-M. Dischler, “Efficient Light Transport Using Precomputed Visibility,” *IEEE Computer Graphics and Applications*, vol. 23, no. 3, pp. 28–37, 2003.

[31] G. M. Davis, S. G. Mallat, and M. Avellaneda, “Adaptive Greedy Approximations,” *Constructive Approximation*, vol. 13, no. 1, pp. 57–98, 1997.

[32] L. De Lathauwer, B. De Moor, and J. Vandewalle, “A Multilinear Singular Value Decomposition,” *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.

[33] L. De Lathauwer, B. De Moor, and J. Vandewalle, “On the Best Rank-1 and Rank-$(R_1, R_2, \ldots, R_n)$ Approximation of Higher-Order Tensors,” *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1324–1342, 2000.

[34] P. E. Debevec, “Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-based Graphics with Global Illumination and High Dynamic Range Photography,” in *Proceedings of ACM SIGGRAPH 1998*, pp. 189–198, 1998.

[35] P. E. Debevec, “Light Probe Image Gallery,” 2004. [Online]. Available: http://www.debevec.org/probes/

[36] P. E. Debevec and J. Malik, “Recovering High Dynamic Range Radiance Maps from Photographs,” in *Proceedings of ACM SIGGRAPH 1997*, pp. 369–378, 1997.

[37] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[38] H. Q. Dinh, G. Turk, and G. G. Slabaugh, “Reconstructing Surfaces by Volumetric Regularization Using Radial Basis Functions,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 10, pp. 1358–1371, 2002.

[39] M. Eck, T. D. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, “Multiresolution Analysis of Arbitrary Meshes,” in *Proceedings of ACM SIGGRAPH 1995*, pp. 173–182, 1995.

[40] K.-L. Fang, “Bidirectional Texture Function Modeling Using Optimized Reparameterization in the Lighting and Viewing Space,” National Chiao Tung University, Master’s thesis, 2008.

[41] M. J. Fengler, W. Freeden, and V. Michel, “The Kaiserslautern Multiscale Geopotential Model SWITCH-03 from Orbit Perturbations of the Satellite CHAMP and Its Comparison to the Models EGM96, UCPH2002_02_0.5, EIGEN-1s, and EIGEN-2,” *Geophysical Journal International*, vol. 157, no. 2, pp. 499–514, 2004.

[42] J. Filip, M. J. Chantler, P. R. Green, and M. Haindl, “A Psychophysically Validated Metric for Bidirectional Texture Data Reduction,” *ACM Transactions on Graphics*, vol. 27, no. 5, 2008.

[43] A. Finkelstein and D. Salesin, “Multiresolution Curves,” in *Proceedings of ACM SIGGRAPH 1994*, pp. 261–268, 1994.

[44] W. Freeden, T. Gervens, and M. Schreiner, *Constructive Approximation on the Sphere*. Oxford University Press, 1998.

[45] W. Freeden and U. Windheuser, “Combined Spherical Harmonic and Wavelet Expansion—A Future Concept in Earth’s Gravitational Determination,” *Applied and Computational Harmonic Analysis*, vol. 4, no. 1, pp. 1–37, 1997.

[46] D. B. Goldman, B. Curless, A. Hertzmann, and S. M. Seitz, “Shape and Spatially-Varying BRDFs from Photometric Stereo,” in *Proceedings of IEEE International Conference on Computer Vision 2005*, pp. 341–348, 2005.

[47] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, “The Lumigraph,” in *Proceedings of ACM SIGGRAPH 1996*, pp. 43–54, 1996.

[48] S. J. Gortler, P. Schröder, M. F. Cohen, and P. Hanrahan, “Wavelet Radiosity,” in *Proceedings of ACM SIGGRAPH 1993*, pp. 221–230, 1993.

[49] P. Green, J. Kautz, and F. Durand, “Efficient Reflectance and Visibility Approximations for Environment Map Rendering,” *Computer Graphics Forum*, vol. 26, no. 3, pp. 495–502, 2007.

[50] P. Green, J. Kautz, W. Matusik, and F. Durand, “View-Dependent Precomputed Light Transport Using Nonlinear Gaussian Function Approximations,” in *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2006*, pp. 7–14, 2006.

[51] J. Gu, C.-I. Tu, R. Ramamoorthi, P. Belhumeur, W. Matusik, and S. Nayar, “Time-Varying Surface Appearance: Acquisition, Modeling and Rendering,” *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 762–771, 2006.

[52] M. Haindl and J. Filip, “Fast BTF Texture Modelling,” in *Proceedings of the 3rd International Workshop on Texture Analysis and Synthesis*, pp. 47–52, 2003.

[53] M. Haindl and J. Filip, “Extreme Compression and Modeling of Bidirectional Texture Function,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 10, pp. 1859–1865, 2007.

[54] M. Haindl, J. Grim, P. Pudil, and M. Kudo, “A Hybrid BTF Model Based on Gaussian Mixtures,” in *Proceedings of the 4th International Workshop on Texture Analysis and Synthesis*, pp. 95–100, 2005.

[55] C. Han, B. Sun, R. Ramamoorthi, and E. Grinspun, “Frequency Domain Normal Map Filtering,” *ACM Transactions on Graphics*, vol. 26, no. 3, 2007.

[56] K. Hara, K. Nishino, and K. Ikeuchi, “Mixture of Spherical Distributions for Single-View Relighting,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 1, pp. 25–35, 2008.

[57] R. A. Harshman, “Foundations of the PARAFAC Procedure: Models and Conditions for an ’Explanatory’ Multimodal Factor Analysis,” *UCLA Working Papers in Phonetics*, vol. 16, pp. 1–84, 1970.

[58] T. Hawkins, P. Einarsson, and P. E. Debevec, “Acquisition of Time-Varying Participating Media,” *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 812–815, 2005.

[59] T. Hazan, S. Polak, and A. Shashua, “Sparse Image Coding Using a 3D Non-Negative Tensor Factorization,” in *Proceedings of IEEE International Conference on Computer Vision 2005*, pp. 50–57, 2005.

[60] X. He, D. Cai, H. Liu, and J. Han, “Image Clustering with Tensor Representation,” in *Proceedings of ACM Multimedia 2005*, pp. 132–140, 2005.

[61] W. Heidrich, K. Daubert, J. Kautz, and H.-P. Seidel, “Illuminating Micro Geometry Based on Precomputed Visibility,” in *Proceedings of ACM SIGGRAPH 2000*, pp. 455–464, 2000.

[62] W. Heidrich and H.-P. Seidel, “Realistic, Hardware-Accelerated Shading and Lighting,” in *Proceedings of ACM SIGGRAPH 1999*, pp. 171–178, 1999.

[63] P. O. Hoyer, “Non-Negative Matrix Factorization with Sparseness Constraints,” *Journal of Machine Learning Research*, vol. 5, pp. 1457–1469, 2004.

[64] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. Wiley Press, 2001.

[65] Industrial Light & Magic (ILM), *Technical Introduction to OpenEXR*, 2006, http://www.openexr.com/documentation.html.

[66] C. E. Jacobs, A. Finkelstein, and D. Salesin, “Fast Multiresolution Image Querying,” in *Proceedings of ACM SIGGRAPH 1995*, pp. 277–286, 1995.

[67] D. L. James and K. Fatahalian, “Precomputing Interactive Dynamic Deformable Scenes,” *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 879–887, 2003.

[68] T. S. Jebara, “Images as Bags of Pixels,” in *Proceedings of IEEE International Conference on Computer Vision 2003*, pp. 265–272, 2003.

[69] H. W. Jensen, “Importance Driven Path Tracing Using the Photon Map,” in *Proceedings of Eurographics Workshop on Rendering 1995*, pp. 326–335, 1995.

[70] H. W. Jensen, “Global Illumination Using Photon Maps,” in *Proceedings of Eurographics Workshop on Rendering 1996*, pp. 21–30, 1996.

[71] Q.-Z. Jiang, “Importance Sampling from Product of the BRDF and the Illumination Using Spherical Radial Basis Functions,” National Chiao Tung University, Master’s thesis, 2007.

[72] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. Springer-Verlag Press, 2002.

[73] J. T. Kajiya, “The Rendering Equation,” in *Proceedings of ACM SIGGRAPH 1986*, pp. 143–150, 1986.

[74] J. T. Kajiya and B. P. Von Herzen, “Ray Tracing Volume Densities,” in *Proceedings of ACM SIGGRAPH 1984*, pp. 165–174, 1984.

[75] N. Kambhatla and T. K. Leen, “Dimension Reduction by Local Principal Component Analysis,” *Neural Computation*, vol. 9, no. 7, pp. 1493–1516, 1997.

[76] J. Kautz, J. Lehtinen, and T. Aila, “Hemispherical Rasterization for Self-Shadowing of Dynamic Objects,” in *Proceedings of Eurographics Symposium on Rendering 2004*, pp. 179–184, 2004.

[77] J. Kautz and M. D. McCool, "Interactive Rendering with Arbitrary BRDFs Using Separable Approximations," in *Proceedings of Eurographics Workshop on Rendering 1999*, pp. 247–260, 1999.

[78] E. Kofidis and P. A. Regalia, "Tensor Approximation and Signal Processing Applications," in *Structured Matrices in Mathematics, Computer Science, and Engineering I*, pp. 103–134. American Mathematical Society, 2001.

[79] E. Kofidis and P. A. Regalia, "On the Best Rank-1 Approximation of Higher-Order Supersymmetric Tensors," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 3, pp. 863–884, 2002.

[80] T. Kohonen, *Self-Organizing Maps*, 3rd ed. Springer-Verlag Press, 2001.

[81] M. L. Koudelka, S. Magda, P. N. Belhumeur, and D. J. Kriegman, "Acquisition, Compression, and Synthesis of Bidirectional Texture Functions," in *Proceedings of the 3rd International Workshop on Texture Analysis and Synthesis*, pp. 59–64, 2003.

[82] M. A. Kramer, "Nonlinear Principal Component Analysis Using Autoassociative Neural Networks," *AIChE Journal*, vol. 37, no. 2, pp. 233–243, 1991.

[83] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski, "Dictionary Learning Algorithms for Sparse Representation," *Neural Computation*, vol. 15, no. 2, pp. 349–396, 2003.

[84] Kriegman Research Group, University of California at San Diego, "Volumetric Surface Texture Database," 2007. [Online]. Available: http://vision.ucsd.edu/kriegman-grp/research/vst/

[85] A. W. Kristensen, T. Akenine-Möller, and H. W. Jensen, "Precomputed Local Radiance Transfer for Real-Time Lighting Design," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 1208–1215, 2005.

[86] E. P. F. Lafortune, S.-C. Foo, K. E. Torrance, and D. P. Greenberg, "Non-Linear Approximation of Reflectance Functions," in *Proceedings of ACM SIGGRAPH 1997*, pp. 117–126, 1997.

[87] P. A. Lalonde and A. Fournier, "A Wavelet Representation of Reflectance Functions," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 4, pp. 329–336, 1997.

[88] P. A. Lalonde and A. Fournier, "Generating Reflected Directionsfrom BRDF Data," *Computer Graphics Forum*, vol. 16, no. 3, pp. 293–300, 1997.

[89] L. Latta and A. Kolb, “Homomorphic Factorization of BRDF-Based Lighting Computation,” *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 509–516, 2002.

[90] J. Lawrence, A. Ben-Artzi, C. DeCoro, W. Matusik, H. Pfister, R. Ramamoorthi, and S. Rusinkiewicz, “Inverse Shade Trees for Non-Parametric Material Representation and Editing,” *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 735–745, 2006.

[91] J. Lawrence, S. Rusinkiewicz, and R. Ramamoorthi, “Efficient BRDF Importance Sampling Using a Factored Representation,” *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 496–505, 2004.

[92] N. D. Lawrence, “Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models,” *Journal of Machine Learning Research*, vol. 6, pp. 1783–1816, 2005.

[93] D. D. Lee and H. S. Seung, “Learning the Parts of Objects by Non-Negative Matrix Factorization,” *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[94] D. D. Lee and H. S. Seung, “Algorithms for Non-Negative Matrix Factorization,” in *Advances in Neural Information Processing Systems 13*, pp. 556–562. MIT Press, 2001.

[95] J. A. Lee and M. Verleysen, *Nonlinear Dimensionality Reduction*. Springer-Verlag Press, 2007.

[96] S. Lefebvre and H. Hoppe, “Parallel Controllable Texture Synthesis,” *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 777–786, 2005.

[97] S. Lefebvre and H. Hoppe, “Appearance-Space Texture Synthesis,” *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 541–548, 2006.

[98] J. Lehtinen, “A Framework for Precomputed and Captured Light Transport,” *ACM Transactions on Graphics*, vol. 26, no. 4, 2007.

[99] J. Lehtinen and J. Kautz, “Matrix Radiance Transfer,” in *Proceedings of ACM Symposium on Interactive 3D Graphics 2003*, pp. 59–64, 2003.

[100] T. K.-H. Leung and J. Malik, “Representing and Recognizing the Visual Appearance of Materials Using Three-Dimensional Textons,” *International Journal of Computer Vision*, vol. 43, no. 1, pp. 29–44, 2001.

[101] M. Levoy and P. Hanrahan, “Light Field Rendering,” in *Proceedings of ACM SIGGRAPH 1996*, pp. 31–42, 1996.

[102] M. S. Lewicki and T. J. Sejnowski, “Learning Overcomplete Representations,” *Neural Computation*, vol. 12, no. 2, pp. 337–365, 2000.

[103] E. Lindholm, M. J. Kligard, and H. P. Moreton, “A User-Programmable Vertex Engine,” in *Proceedings of ACM SIGGRAPH 2001*, pp. 149–158, 2001.

[104] G. Liu, J. Zhang, W. Wang, and L. McMillan, “Human Motion Estimation from a Reduced Marker Set,” in *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2006*, pp. 35–42, 2006.

[105] X. Liu, P.-P. J. Sloan, H.-Y. Shum, and J. Snyder, “All-Frequency Precomputed Radiance Transfer for Glossy Objects,” in *Proceedings of Eurographics Symposium on Rendering 2004*, pp. 337–344, 2004.

[106] M. Lounsbery, T. D. DeRose, and J. D. Warren, “Multiresolution Analysis for Surfaces of Arbitrary Topological Type,” *ACM Transactions on Graphics*, vol. 16, no. 1, pp. 34–73, 1997.

[107] W.-C. Ma, S.-H. Chao, Y.-T. Tseng, Y.-Y. Chuang, C.-F. Chang, B.-Y. Chen, and M. Ouhyoung, “Level-of-Detail Representation of Bidirectional Texture Functions for Real-Time Rendering,” in *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2005*, pp. 187–194, 2005.

[108] W.-C. Ma, C.-T. Hsiao, K.-Y. Lee, Y.-Y. Chuang, and B.-Y. Chen, “Real-Time Triple Product Relighting Using Spherical Local-Frame Parameterization,” *The Visual Computer*, vol. 22, no. 9-11, pp. 682–692, 2006.

[109] D. Mahajan, I. K. Shlizerman, R. Ramamoorthi, and P. N. Belhumeur, “A Theory of Locally Low Dimensional Light Transport,” *ACM Transactions on Graphics*, vol. 26, no. 3, 2007.

[110] S. G. Mallat and Z. Zhang, “Matching Pursuits with Time-Frequency Dictionaries,” *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[111] T. Malzbender, D. Gelb, and H. J. Wolters, “Polynomial Texture Maps,” in *Proceedings of ACM SIGGRAPH 2001*, pp. 519–528, 2001.

[112] K. V. Mardia and P. E. Jupp, *Directional Statistics*. Wiley Press, 2000.

[113] W. Matusik, “A Data-Driven Reflectance Model,” Massachusetts Institute of Technology, Ph.D. dissertation, 2003.

[114] W. Matusik, H. Pfister, M. Brand, and L. McMillan, “A Data-Driven Feflectance Model,” *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 759–769, 2003.

[115] D. K. McAllister, A. Lastra, and W. Heidrich, “Efficient Rendering of Spatial Bi-Directional Reflectance Distribution Functions,” in *Proceedings of Graphics Hardware 2002*, pp. 79–88, 2002.

[116] M. D. McCool, J. Ang, and A. Ahmad, “Homomorphic Factorization of BRDFs for High-Performance Rendering,” in *Proceedings of ACM SIGGRAPH 2001*, pp. 171–178, 2001.

[117] T. E. McGraw, B. C. Vemuri, R. Yezierski, and T. Mareci, “Segmentation of High Angular Resolution Diffusion MRI Modeled as a Field of Von Mises-Fisher Mixtures,” in *Proceedings of European Conference on Computer Vision 2006*, pp. 463–475, 2006.

[118] J. Meseth, G. Müller, and R. Klein, “Reflectance Field Based Real-Time, High-Quality Rendering of Bidirectional Texture Functions,” *Computers & Graphics*, vol. 28, no. 1, pp. 105–112, 2004.

[119] G. S. P. Miller, S. M. Rubin, and D. B. Ponceleon, “Lazy Decompression of Surface Light Fields for Precomputed Global Illumination,” in *Proceedings of Eurographics Workshop on Rendering 1998*, pp. 281–292, 1998.

[120] Mitsubishi Electric Research Laboratories (MERL), “MERL BRDF Database,” 2006. [Online]. Available: http://www.merl.com/brdf/

[121] G. Müller, J. Meseth, and R. Klein, “Fast Environmental Lighting for Local-PCA Encoded BTFs,” in *Proceedings of Computer Graphics International 2004*, pp. 198–205, 2004.

[122] M. Müller, B. Heidelberger, M. Teschner, and M. H. Gross, “Meshless Deformations Based on Shape Matching,” *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 471–478, 2005.

[123] F. J. Narcowich and J. D. Ward, “Nonstationary Wavelets on the m-Sphere for Scattered Data,” *Applied and Computational Harmonic Analysis*, vol. 3, no. 4, pp. 324–336, 1996.

[124] S. K. Nayar, P. N. Belhumeur, and T. E. Boult, “Lighting Sensitive Display,” *ACM Transactions on Graphics*, vol. 23, no. 4, pp. 963–979, 2004.

[125] R. Ng, R. Ramamoorthi, and P. Hanrahan, “All-Frequency Shadows Using Non-Linear Wavelet Lighting Approximation,” *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 376–381, 2003.

[126] R. Ng, R. Ramamoorthi, and P. Hanrahan, “Triple Product Wavelet Integrals for All-Frequency Relighting,” *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 477–487, 2004.

[127] H. T. Nguyen, Q. Ji, and A. W. M. Smeulders, “Robust Multi-Target Tracking Using Spatio-Temporal Context,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition 2006*, pp. 578–585, 2006.

[128] NVIDIA Corporation, "NVIDIA CUDA: Compute Unified Device Architecture," 2008. [Online]. Available: http://www.nvidia.com/cuda/

[129] H.-S. Oh and T.-H. Li, "Estimation of Global Temperature Fields from Scattered Observations by a Spherical-Wavelet-Based Spatially Adaptive Method," *Journal of the Royal Statistical Society - Series B*, vol. 66, no. 1, pp. 221–238, 2004.

[130] M. M. Oliveira, G. Bishop, and D. K. McAllister, "Relief Texture Mapping," in *Proceedings of ACM SIGGRAPH 2000*, pp. 359–368, 2000.

[131] J. D. Owens, D. P. Luebke, N. K. Govindaraju, M. J. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, "A Survey of General-Purpose Computation on Graphics Hardware," *Computer Graphics Forum*, vol. 26, no. 1, pp. 80–113, 2007.

[132] P. Paatero and U. Tapper, "Positive Matrix Factorization: A Non-Negative Factor Model with Optimal Utilization of Error Estimates of Data Values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[133] G. Patanè and M. Russo, "The Enhanced LBG Algorithm," *Neural Networks*, vol. 14, no. 9, pp. 1219–1237, 2001.

[134] M. Pauly, R. Keiser, B. Adams, P. Dutré, M. H. Gross, and L. J. Guibas, "Meshless Animation of Fracturing Solids," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 957–964, 2005.

[135] P. Peers, K. Vom Berge, W. Matusik, R. Ramamoorthi, J. Lawrence, S. Rusinkiewicz, and P. Dutré, "A Compact Factored Representation of Heterogeneous Subsurface Scattering," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 746–753, 2006.

[136] H. Pfister, M. Zwicker, J. Van Baar, and M. H. Gross, "Surfels: Surface Elements as Rendering Primitives," in *Proceedings of ACM SIGGRAPH 2000*, pp. 335–342, 2000.

[137] M. M. Pharr and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Press, 1996.

[138] F. Policarpo and M. M. Oliveira, "Relief Mapping of Non-Height-Field Surface Details," in *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2006*, pp. 55–62, 2006.

[139] F. Policarpo, M. M. Oliveira, and J. L. D. Comba, "Real-Time Relief Mapping on Arbitrary Polygonal Surfaces," in *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2005*, pp. 155–162, 2005.

[140] Program of Computer Graphics, Cornell University, "Reflectance Data," 2002. [Online]. Available: http://www.graphics.cornell.edu/online/measurements/reflectance/

[141] K. Proudfoot, W. R. Mark, S. Tzvetkov, and P. Hanrahan, “A Real-Time Procedural Shading System for Programmable Graphics Hardware,” in *Proceedings of ACM SIGGRAPH 2001*, pp. 159–170, 2001.

[142] Z. Qin, M. D. McCool, and C. S. Kaplan, “Real-Time Texture-Mapped Vector Glyphs,” in *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2006*, pp. 125–132, 2006.

[143] R. Ramamoorthi and P. Hanrahan, “A Signal-Processing Framework for Inverse Rendering,” in *Proceedings of ACM SIGGRAPH 2001*, pp. 117–128, 2001.

[144] R. Ramamoorthi and P. Hanrahan, “An Efficient Representation for Irradiance Environment Maps,” in *Proceedings of ACM SIGGRAPH 2001*, pp. 497–500, 2001.

[145] R. Ramamoorthi and P. Hanrahan, “Frequency Space Environment Map Rendering,” *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 517–526, 2002.

[146] R. Ramamoorthi and P. Hanrahan, “A Signal-Processing Framework for Reflection,” *ACM Transactions on Graphics*, vol. 23, no. 4, pp. 1004–1042, 2004.

[147] J. O. Ramsay and B. W. Silverman, *Functional Data Analysis*, 2nd ed. Springer-Verlag Press, 2005.

[148] L. Rebollo-Neira and D. Lowe, “Optimized Orthogonal Matching Pursuit Approach,” *IEEE Signal Processing Letters*, vol. 9, no. 4, pp. 137–140, 2002.

[149] Z. Ren, R. Wang, J. Snyder, K. Zhou, X. Liu, B. Sun, P.-P. J. Sloan, H. Bao, Q. Peng, and B. Guo, “Real-Time Soft Shadows in Dynamic Scenes Using Spherical Harmonic Exponentiation,” *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 977–986, 2006.

[150] S. T. Roweis and L. K. Saul, “Nonlinear Dimensionality Reduction by Locally Linear Embedding,” *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[151] S. Rusinkiewicz, “A new change of variables for efficient brdf representation,” in *Proceedings of Eurographics Workshop on Rendering 1998*, pp. 11–22, 1998.

[152] S. Rusinkiewicz and M. Levoy, “QSplat: A Multiresolution Point Rendering System for Large Meshes,” in *Proceedings of ACM SIGGRAPH 2000*, pp. 343–352, 2000.

[153] M. Sattler, R. Sarlette, and R. Klein, “Efficient and Realistic Visualization of Cloth,” in *Proceedings of Eurographics Symposium on Rendering 2003*, pp. 167–178, 2003.

[154] L. K. Saul and S. T. Roweis, “Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifold,” *Journal of Machine Learning Research*, vol. 4, pp. 119–155, 2003.

[155] B. Schölkopf, A. J. Smola, and K.-R. Müller, “Nonlinear Component Analysis as a Kernel Eigenvalue Problem,” *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[156] P. Schröder and W. Sweldens, “Spherical Wavelets: Efficiently Representing Functions on the Sphere,” in *Proceedings of ACM SIGGRAPH 1995*, pp. 161–172, 1995.

[157] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. K. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan, and P. Hanrahan, “Larrabee: A Many-Core x86 Architecture for Visual Computing,” *ACM Transactions on Graphics*, vol. 27, no. 3, 2008.

[158] J. Shade, S. J. Gortler, L. wei He, and R. Szeliski, “Layered Depth Images,” in *Proceedings of ACM SIGGRAPH 1998*, pp. 231–242, 1998.

[159] A. Shashua and T. Hazan, “Non-Negative Tensor Factorization with Applications to Statistics and Computer Vision,” in *Proceedings of International Conference on Machine Learning 2005*, pp. 792–799, 2005.

[160] F. X. Sillion, J. Arvo, S. H. Westin, and D. P. Greenberg, “A Global Illumination Solution for General Reflectance Distributions,” in *Proceedings of ACM SIGGRAPH 1991*, pp. 187–196, 1991.

[161] P.-P. J. Sloan, J. Hall, J. C. Hart, and J. Snyder, “Clustered Principal Components for Precomputed Radiance Transfer,” *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 382–391, 2003.

[162] P.-P. J. Sloan, J. Kautz, and J. Snyder, “Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments,” *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 527–536, 2002.

[163] P.-P. J. Sloan, X. Liu, H.-Y. Shum, and J. Snyder, “Bi-Scale Radiance Transfer,” *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 370–375, 2003.

[164] P.-P. J. Sloan, B. Luna, and J. Snyder, “Local, Deformable Precomputed Radiance Transfer,” *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 1216–1224, 2005.

[165] A. K. Smilde, R. Bro, and P. Geladi, *Multi-way Analysis: Applications in the Chemical Sciences*. Wiley Press, 2004.

[166] Y. Song, Y. Chen, X. Tong, S. Lin, J. Shi, B. Guo, and H.-Y. Shum, “Shell Radiance Texture Functions,” *The Visual Computer*, vol. 21, no. 8-10, pp. 774–782, 2005.

[167] M. M. Stark, J. Arvo, and B. E. Smits, “Barycentric Parameterizations for Isotropic BRDFs,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 2, pp. 126–138, 2005.

[168] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin, *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann Press, 1996.

[169] B. Sun, K. Sunkavalli, R. Ramamoorthi, P. N. Belhumeur, and S. K. Nayar, "Time-Varying BRDFs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 3, pp. 595–609, 2007.

[170] W. Sun and A. Mukherjee, "Generalized Wavelet Product Integral for Rendering Dynamic Glossy Objects," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 955–966, 2006.

[171] X. Sun, K. Zhou, Y. Chen, S. Lin, J. Shi, and B. Guo, "Interactive Relighting with Dynamic BRDFs," *ACM Transactions on Graphics*, vol. 26, no. 3, 2007.

[172] F. Suykens, K. vom Berge, A. Lagae, and P. Dutré, "Interactive Rendering with Bidirectional Texture Functions," *Computer Graphics Forum*, vol. 22, no. 3, pp. 463–472, 2003.

[173] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[174] M. E. Tipping and C. M. Bishop, "Mixtures of Probabilistic Principal Component Analysers," *Neural Computation*, vol. 11, no. 2, pp. 443–482, 1999.

[175] M. E. Tipping and C. M. Bishop, "Probabilistic Principal Component Analysis," *Journal Of The Royal Statistical Society Series B*, vol. 61, no. 3, pp. 611–622, 1999.

[176] J. A. Tropp, "Greed is Good: Algorithmic Results for Sparse Approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[177] J. A. Tropp, "Topics in Sparse Approximation," The University of Texas at Austin, Ph.D. dissertation, 2004.

[178] Y.-T. Tsai, C.-C. Chang, Q.-Z. Jiang, and S.-C. Weng, "Importance Sampling of Products from Illumination and BRDF Using Spherical Radial Basis Functions," *The Visual Computer*, vol. 24, no. 7-9, pp. 817–826, 2008.

[179] Y.-T. Tsai and Z.-C. Shih, "All-Frequency Precomputed Radiance Transfer Using Spherical Radial Basis Functions and Clustered Tensor Approximation," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 967–976, 2006.

[180] L. R. Tucker, "The Extension of Factor Analysis to Three-Dimensional Matrices," in *Contributions to Mathematical Psychology*, pp. 110–127. Holt, Rinehart and Winston Press, 1964.

[181] L. R. Tucker, “Some Mathematical Notes on Three-Mode Factor Analysis,” *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[182] G. Turk and J. F. O’Brien, “Modelling with Implicit Surfaces that Interpolate,” *ACM Transactions on Graphics*, vol. 21, no. 4, pp. 855–873, 2002.

[183] M. A. O. Vasilescu and D. Terzopoulos, “Multilinear Analysis of Image Ensembles: TensorFaces,” in *Proceedings of European Conference on Computer Vision 2002*, pp. 447–460, 2002.

[184] M. A. O. Vasilescu and D. Terzopoulos, “Multilinear Subspace Analysis of Image Ensembles,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition 2003*, pp. 93–99, 2003.

[185] M. A. O. Vasilescu and D. Terzopoulos, “TensorTextures: Multilinear Image-Based Rendering,” *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 336–342, 2004.

[186] E. Veach, “Robust Monte Carlo Methods for Light Transport Simulation,” Stanford University, Ph.D. dissertation, 1997.

[187] E. Veach and L. J. Guibas, “Optimally Combining Sampling Techniques for Monte Carlo Rendering,” in *Proceedings of ACM SIGGRAPH 1995*, pp. 419–428, 1995.

[188] R. Vidal, Y. Ma, and S. Sastry, “Generalized Principal Component Analysis (GPCA),” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 12, pp. 1945–1959, 2005.

[189] D. Vlasic, M. Brand, H. Pfister, and J. Popović, “Face Transfer with Multilinear Models,” *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 426–433, 2005.

[190] H. Wang and N. Ahuja, “Facial Expression Decomposition,” in *Proceedings of IEEE International Conference on Computer Vision 2003*, pp. 958–965, 2003.

[191] H. Wang and N. Ahuja, “A Tensor Approximation Approach to Dimensionality Reduction,” *International Journal of Computer Vision*, vol. 76, no. 3, pp. 217–229, 2008.

[192] H. Wang, Q. Wu, L. Shi, Y. Yu, and N. Ahuja, “Out-of-Core Tensor Approximation of Multi-Dimensional Matrices of Visual Data,” *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 527–535, 2005.

[193] J. Wang, X. Tong, S. Lin, M. Pan, C. Wang, H. Bao, B. Guo, and H.-Y. Shum, “Appearance Manifolds for Modeling Time-Variant Appearance of Materials,” *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 754–761, 2006.

[194] L. Wang, X. Wang, X. Tong, S. Lin, S.-M. Hu, B. Guo, and H.-Y. Shum, "View-Dependent Displacement Mapping," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 334–339, 2003.

[195] R. Wang, J. Tran, and D. P. Luebke, "All-Frequency Relighting of Non-Diffuse Objects Using Separable BRDF Approximation," in *Proceedings of Eurographics Symposium on Rendering 2004*, pp. 345–354, 2004.

[196] R. Wang, J. Tran, and D. P. Luebke, "All-Frequency Relighting of Glossy Objects," *ACM Transactions on Graphics*, vol. 25, no. 2, pp. 293–318, 2006.

[197] X. Wang, X. Tong, S. Lin, S.-M. Hu, B. Guo, and H.-Y. Shum, "Generalized Displacement Maps," in *Proceedings of Eurographics Symposium on Rendering 2004*, pp. 227–234, 2004.

[198] G. J. Ward, "Measuring and Modeling Anisotropic Reflection," in *Proceedings of ACM SIGGRAPH 1992*, pp. 265–272, 1992.

[199] G. N. Watson, *A Treatise on the Theory of Bessel Functions*, 2nd ed. Cambridge University Press, 1944.

[200] S.-C. Weng, "An Efficient BRDF Importance Sampling Method for Global Illuimation," National Chiao Tung University, Master's thesis, 2006.

[201] S. H. Westin, J. Arvo, and K. E. Torrance, "Predicting Reflectance Functions from Complex Surfaces," in *Proceedings of ACM SIGGRAPH 1992*, pp. 255–264, 1992.

[202] T. Whitted, "An Improved Illumination Model for Shaded Display," *Communications of the ACM*, vol. 23, no. 6, pp. 343–349, 1980.

[203] T.-T. Wong, C.-W. Fu, P.-A. Heng, and C.-S. Leung, "The Plenoptic Illumination Function," *IEEE Transactions on Multimedia*, vol. 4, no. 3, pp. 361–371, 2002.

[204] Q. Wu, T. Xia, C. Chen, H.-Y. S. Lin, H. Wang, and Y. Yu, "Hierarchical Tensor Approximation of Multi-Dimensional Visual Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 1, pp. 186–199, 2008.

[205] D. Xu, S. Yan, L. Zhang, H. Zhang, Z. Liu, and H.-Y. Shum, "Concurrent Subspaces Analysis," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition 2005*, pp. 203–208, 2005.

[206] K. Xu, Y.-T. Jia, H. Fu, S.-M. Hu, and C.-L. Tai, "Spherical Piecewise Constant Basis Functions for All-Frequency Precomputed Radiance Transfer," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 454–467, 2008.

[207] S. Yan, D. Xu, S. Lin, T. S. Huang, and S.-F. Chang, “Element Rearrangement for Tensor-Based Subspace Learning,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition 2007*, 2007.

[208] J. Yang, D. Zhang, A. F. Frangi, and J.-Y. Yang, “Two-Dimensional PCA: A New Approach to Appearance-Based Face Representation and Recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 131–137, 2004.

[209] J. Ye, “Generalized Low Rank Approximations of Matrices,” *Machine Learning*, vol. 61, no. 1-3, pp. 167–191, 2005.

[210] R. Zass and A. Shashua, “Nonnegative Sparse PCA,” in *Advances in Neural Information Processing Systems 19*, pp. 1561–1568. MIT Press, 2007.

[211] K. Zhou, Y. Hu, S. Lin, B. Guo, and H.-Y. Shum, “Precomputed Shadow Fields for Dynamic Scenes,” *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 1196–1201, 2005.

[212] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, “Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization,” *ACM Transactions on Mathematical Software*, vol. 23, no. 4, pp. 550–560, 1997.

[213] T. Zickler, S. Enrique, R. Ramamoorthi, and P. N. Belhumeur, “Reflectance Sharing: Image-Based Rendering from a Sparse Set of Images,” in *Proceedings of Eurographics Symposium on Rendering 2005*, pp. 253–264, 2005.

[214] T. Zickler, R. Ramamoorthi, S. Enrique, and P. N. Belhumeur, “Reflectance Sharing: Predicting Appearance from a Sparse Set of Images of a Known Shape,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1287–1302, 2006.

[215] M. Zwicker, H. Pfister, J. Van Baar, and M. H. Gross, “Surface Splatting,” in *Proceedings of ACM SIGGRAPH 2001*, pp. 371–378, 2001.

Vita

Yu-Ting Tsai was born in January, 1978, in Taichung, Taiwan, Republic of China. He received the B.S. and M.S. degrees in electronics engineering from National Chiao Tung University in 2000 and 2002, respectively. In 2002, he decided to study a Ph.D. degree and joined the Computer Graphics Laboratory at the Department of Computer Science, National Chiao Tung University. During the Ph.D. program, he has enthusiastically pursued research quality and published one technical paper on real-time rendering at SIGGRAPH 2006. After several years of hard work, he finally received the Ph.D. degree in computer science from National Chiao Tung University in 2009. His research interests mainly include real-time rendering, material modeling, and global illumination. He is also highly interested in research fields other than computer graphics, such as computer hardware, computer vision, machine learning, and signal processing.