Rendering Algorithms for Natural Scenes

Weiming Dong

Département de formation doctorale en informatique
UFR STMIA

École doctorale IAEM Lorraine

Algorithmes pour la Simulation Visuelle de Scènes Naturelles

THÈSE

présentée et soutenue publiquement le 4 mai 2007

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1

(spécialité informatique)

par

DONG Weiming

Composition du jury

Président :	Andre Gagalowicz	Directeur de recherche, INRIA, Paris, France
Rapporteurs :	Jiaxin Wang	Professeur, Tsinghua University, Pékin, Chine
	Pascal Guitton	Professeur, Université Bordeaux 1, Bordeaux, France
Examinateurs :	Jean-Paul Haton	Professeur, Université Henri Poincaré, Nancy, France
	Jean-Claude Paul	Directeur de recherche, INRIA, Nancy, France
	Bruno Lévy	Chargé de recherche, INRIA, Nancy, France

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thloria.

Remerciements

It takes a tremendous amount of dedication and perseverance to complete a doctoral dissertation. At least that's what I have been told. I'm not sure I possess either of those qualities in great enough quantity to complete a dissertation, but I am fortunate in that I have been constantly surrounded by many great and motivated people whose encouragement and assistance throughout this process has made up for any attributes I lack. I would like to sincerely extend my gratitude to those individuals here.

First and foremost I would like to thank my advisor, Prof. Jean-Claude Paul, for everything I have learned from him and for his infinite kindness and patience over the past three years. I also thank my co-advisor Dr. Bruno Lévy for the helpful discussions we had about my thesis and many other graphics topics, and Dr. Laurent Alonso, Dr. Wan-Chiu Li, Dr. Nicolas Ray and Gregory Lecot for helping shape my rendering thinking. I am grateful to all the INRIA Lorraine/LORIA faculty, fellow graduate students, and staff for creating a great research environment and a fun place to spend a few years of one's life. Specially, I would like to thank the administrative assistant of ALICE team Mrs. Isabelle Herlich, for her kindly help on all the tedious administrative things before and during my career in France.

I would like to thank my jury, consisting of Prof. Jiaxin Wang, Prof. Pascal Guitton, Prof. André Gagalowicz, Prof. Jean-Paul Haton, Prof. Jean-Claude Paul and Dr. Bruno Lévy, for their encouragement and discussions about my thesis.

My friends in Nancy have helped a lot at various stage of my graduate career. I would like to thank Dr. Xiaopeng Zhang, Dr. Huaizhong Kou, Joseph Razik, Dong Cheng, Dr. Bin Wang, Dr. Jian Li, Dr. Liping Lu, Ning Jia, Dr. Jingwei Yang, Yun Wang, Lei Zheng and Dr. Calvin Chen, for helping me to solve many problems for my study or for my life.

Thanks to Prof. Baogang Hu, Dr. Xiaopeng Zhang and Bizhen Hong, for providing me a pleasant research environment when I was visiting LIAMA (the Chinese brunch of INRIA). Thanks also to Prof. Junhai Yong and Dr. Hui Zhang, for their kindly help when I was in Tsinghua University. Specially thank to Ning Zhou and Xing Mei, they have been working with me for the last two years and I appreciate very much for all their help in my papers and thesis. The programs of the color patterns part in my thesis are all prepared by Ning Zhou.

I am immensely grateful to my parents and my wife for everything they have always been doing for me, and especially for all their help in the last few months. They took care of my my life when I was preparing my thesis in Beijing and in Nancy, without them, it would not have been finished on time.

At last, I thank Li-Yi Wei, Steve Zelinka, Vivek Kwatra, Tuen-Young Ng, Jiaya Jia for sharing their results and texture samples on the web sites. Thank Professor Ramin Zabih and his students at Cornell University for sharing their code for computing graph min-cut.

"When the fans blow a whistle to me, I never mind and I'm not sad, because I know they are encouraging me and they love me. When I have overcome a difficulty, I'm not entranced with it, because I know there are still more difficulties waiting for me."
Alessandro del Piero, the Captain of Juventus Football Club

Résumé

L'objet de cette thèse est de proposer des algorithmes permettant de synthétiser des scènes naturelles complexes, par le nombre des objets qui les composent, par leur forme et par les phénomènes optiques qui peuvent caractériser certains objets ou phénomènes naturels, dans l'eau ou l'atmosphère. Notre objectif est de synthètiser, avec une certaine interactivitl'e, des images simulant leur apparence.

Pour atteindre cet objectif, nous proposons des algorithmes permettant de résoudre divers problèmes. Nous proposons tout d'abord un algorithme qui permet de combiner, pour une image de donnée, des systèmes de visualisation conventionnels et une représentation spectrale. Cette technique permet de simuler visuellement une scène où se produisent partiellement des effets optiques complexes (diffraction, interférence, etc.), en quelques secondes. Nous introduisons ensuite un algorithme permettant de modéliser et visualiser des motifs de couleur, des motifs de fleurs, par exemple. Une application de cet algorithme aux variations saisonnières des feuilles est également présente.

Pour les scènes d'une complexité qui n'est pas à la portée des techniques de simulation et de visualisation, nous présentons un algorithme qui permet de synthétiser des scènes naturelles à l'aide d'échantillons d'images. Nous présentons ainsi d'abord une approche dite "Tile-based texture synthesis", fondée sur l'optimisation de la qualité de l'échantillon sélectionné à l'aide de techniques d'intelligence artificielle. Ensuite, nous présentons une technique de génération de textures guidée par des cartes présentant les caractéristiques perceptuelles globales de l'échantillon retenu.

Mots-clés: Couleurs structurelles, motifs de couleur, texture synthétisée, synthétiser des écosystèmes

Abstract

Aiming at developing feasible rendering algorithms for natural scenes, this thesis focus on complex optical phenomena, i.e. generate realistic images of natural objects in outdoor environments. What we mean by "realistic" is that the interactively generated appearance of the natural objects should well catch their optical characteristics. This property makes the rendering algorithms optimum in a simulation point of view.

To solve this problem, we propose several new algorithms from different angles. We first propose a framework that combines the traditional rendering system with spectra representation. With our framework, computing a scene with an object producing wave-based optics effects costs only a few additional seconds and is more efficient than previous methods for the same

quality images. Then we introduce a system for modeling and visualization of flower color patterns. We build a connection between the user-controllable image synthesis and real botanical patterns. An additional application on the simulation of seasonal maples is also presented.

For the natural phenomena whose physical complexities are out of current rendering capability, we employ sample images especially natural textures, and decorate the natural scenes with texture synthesis. In this part, we first present approaches for tile-based texture synthesis that are based on the optimization of tile set quality within Genetic Algorithm-based framework. Then we present an interactive tool for anisotropic 2D texture synthesis guided by feature maps, which preserves some global perceptual characteristics of the example.

Keywords: Structural colors, color patterns, texture synthesis, ecosystem generation

Contents

Résumé Étendu

1 Algorithmes pour la synthèse d'images d'environnements naturels

Nous présentons dans cette thèse des algorithmes permettant de générer des images de synthèse d'environnements naturels.

Les environnements naturels sont très compliqués à simuler. Ils sont généralement composés d'une multitude d'objets de formes géométriques très variées. L'abondance des couleurs et leur subtile variation sont impressionnantes. Les phénomènes optiques qui s'y déroulent sont si complexes que l'on en découvre seulement aujourd'hui peu à peu toute la richesse. Enfin, ce monde est dynamique, vivant, en perpétuelle évolution.

La modélisation géométrique des objets naturels a fait l'objet de nombreux travaux. Les uns sont fondés sur des modèles fondés sur la connaissance que nous avons des phénomènes physiques, biologiques. Par exemple, la simulation des plantes peut être simulée grâce à des modèles d'architecture des plantes et de leur croissance. La modélisation géométrique des objets naturels peut aussi être simulée grâce à des interfaces appropriées permettant de décrire rapidement, des types d'objets : des fleurs, des insectes, etc. Les deux approches requièrent l'usage de primitives géométriques classiques planes ou courbes.

FIG. 1: Cet arbre est généré à partir d'un modèle botanique (GreenLab China).

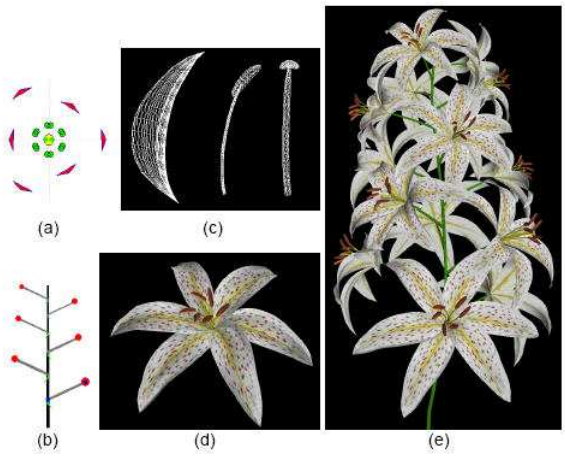


FIG. 2: Lily model [IOOI05]. L'information structurelle est donnée par un diagramme floral (a) et une inflorescence (b).

A une échelle plus petite, les objets naturels présentent des caractéristiques particulières : les animaux, mais aussi certaines fleurs ou insectes présentent une sorte de fourrure, les poissons présentent des écailles, l'écorce d'un arbre présente des irrégularités singulières. Leur microstructure géométrique détermine aussi largement les propriétés de réflectance de ces surfaces. Toutes ces questions ont fait l'objet de nombreuses publications.

Notre travail porte sur l'apparence que prennent les objets dans la lumière, ou sous l'impact de phénomènes biologiques ou encore lorsqu'ils sont si nombreux que leur représentation en 3 Dimensions est hors de portée. Nous présenterons ainsi nos contributions dans trois autres domaines : la génération d'images basées sur la simulation des interactions lumière-matière, la génération des motifs de couleur, et la génération d'images simulant de vastes écosystèmes.

2 Motivation

Considérons la FIG. 3. Elle représente quelques insectes de type Cetonia Aurata, butinant sur une fleur des champs. Considérons que la modélisation géométrique de ces objets est acquise. Comment donner à ces objets flerus et insects l'aspect naturel du monde vivant ? Notre objectif initial était de permettre la synthèse d'images analogues é cette photographie : prédire la couleur des insectes et de ses variations selon leur exposition au soleil, décrire le dégradé de la couleur des fleurs, enfin, mettre en situation ces êtres vivants dans leur écosystème.

2.1 Simuler l'apparence des objets naturels

Prédire l'apparence des objets sous l'effet de la lumière. L'état de l'Art dans le domaine des interactions lumière matière est immense. Beaucoup de travaux sont fondés sur des modèles prédictifs, qui simulent le comportement de la lumière, avec des approximations et des simplifications plus ou moins grandes.

FIG. 3: Une famille de Cetonia Aurata sur une fleur des Champs.

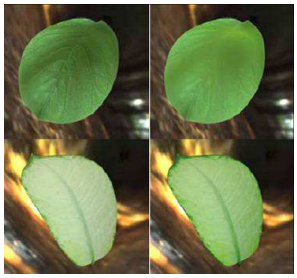


FIG. 4: Le modèle présenté par Wang [WWD+05].

– Dans le monde réel, la réflectance peut être exprimée par une fonction 12D. Les paramètres de cette fonction sont la position de l'entrée et de la sortie de la lumière ; l'angle d'incidence et de réflexion, la longueur d'ondes, le temps.
– On considère généralement que l'entrée et la sortie de la lumière se font au même moment et que la longueur d'ondes est invariable. Ainsi, nous pouvons omettre deux paramètres et obtenir une fonction 8D.
– Ce modèle est communément appelé : 8D BSSRDF (Bidirectional Subsurface Scattering Reflectance Distribution Function) et il est formalisé ainsi :
– Dans ce modèle, la lumière est supposée entrer dans la surface, traverser la surface être réfléchie au sein du matériau avant d'être réfléchie hors de celui-ci. Ainsi le point d'entrée de la lumière est différent de son point de sortie. Si nous considérons que l'entrée et la sortie de la lumière s'effectuent au même point, alors nous obtenons une fonction 6D. En outre, si nous considérons que la réflectance de la lumière est indépendante de la position, nous obtenons le modèle 4D le plus populaire. La fonction est appelée Bidirectional Reflectance Distribution Function (BRDF). Enfin, en ignorant l'effet de l'angle d'incidence et de réflexion, on obtient la fonction 2D qui caractérise une texture.

Considérons maintenant un objet naturel très commun : une feuille d'arbre. En fait, une feuile d'arbre est un matériau très fin, dans lequel la lumière transmise n'est pas réfléchie entièrement. La radiance diffuse réfléchie sur l'une des faces de la feuille est égale à la radiance réfléchie sur l'autre face de la feuille. A notre connaissance, le modèle qui a décrit le mieux ce phénomène est aujourd'hui, le modèle proposé par Jensen [DJ05] dans.

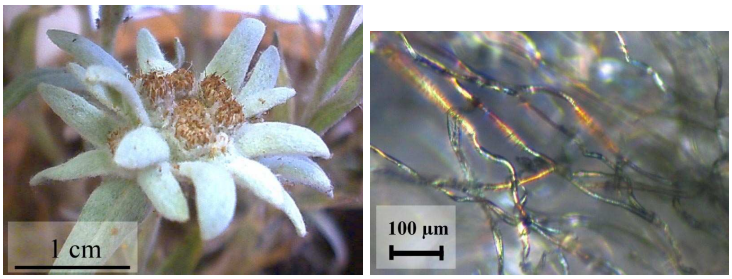


FIG. 5: Les Edelweis sont des plantes herbeuses qui développent des bractes blanche-argentées en étoile, particulièrement denses sur la face exposée au soleil [VRV+05]. L'image microscopique des filaments, qui en constituent les fibres cotonneuses, montre qu'ils diffractent la lumière protégeant la plante des rayons ultra-violet.

Il serait possible d'aller plus loin encore. Govaert dans [GJVU] a modélisé la structure cellulaire tridimensionnelle interne des tissus d'une feuille de type decotelydon. L'épiderme, le parenchyma et la mésophile spongieuse est explicitement décrite. Chaque cellule est caractérisée par un indice de réfraction et un coefficient d'absorbsion. Des techniques de type Monte Carlo sont ensuite mises en œuvre pour valider les propriétés optiques du modèle et les comparer avec les observations réalisées en laboratoire. De telles méthodes sont cependant trop coûteuses aujourd'hui pour être utilisées dans un contexte de synthèse d'images.

2.2 Simuler ou Acquérir ?

Une voie différente a été proposée par Wang dans [WWD+05]. Ils proposent un modèle paramétrique pour simuler le comportement de la lumière sur des surfaces très fines comme les feuilles d'un arbre. Leur modèle est décrit en terme de 6D BRDF et 6D fonction BRTF (Bidirectional Reflection Transmittance Function). Ces fonctions prennent en compte les principaux comportements de la lumière et elles sont définies de façon à permettre de définir la valeur des paramètres à partir de données mesurées. "Il est inutile de modéliser l'intérieur d'un matériau pour décrire une apparence qui peut être mesurée" écrivent les auteurs. C'est vrai. Mais encore faut-il disposer des objets que l'on veut simuler et les techniques de mesures adaptées.

Simuler ou acquérir ? Cette question reste une question ouverte. Nous avons exploré ces deux voies dans cette thèse.

Nous verrons qu'elle peut être posée dans les mêmes termes lorsque l'on souhaite générer des motifs de couleur. La synthèse de grands eco-systèmes également peut être réalisée par simulation ou à partir d'images.

2.3 Simuler des couleurs structurelles

De nombreux êtres naturels, poissons, insectes, fleurs, ont une couleur dite "structurelle". La découverte de ce phénomène ne fait que s'amplifier depuis l'introduction des nouvelles techniques, comme le microscope électronique chez les botanistes. L'étude de ces phénomènes est

un champ de recherche actif chez les les physiciens (photonique), principalement en fonction du champ d'applications nouvelles qu'ouvre ces découvertes dans le domaine des nanotechnologies.

Les couleurs dites "structurelles" résultent des effets des phénomènes d'interférence ou de diffraction de la lumière. L'interférence produit des effets de couleur qui sont dus aux différences de phase causée par une onde qui traverse un milieu très fin ayant différents indices de réfraction. On observe un phénomène de diffraction lorsque la lumière rencontre une surface dont le détail est comparable à la longueur d'onde de la lumière. Ainsi, dans la FIG. 5, la blancheur argentée de l'Edelweis est due à la diffraction de la lumière au sein des ses filaments.

Les phénomènes d'interférence ont fait l'objet de plusieurs publications dans le domaine de la synthèse d'images. Ils restent très compliqués à représenter. Cependant, ils peuvent être simulés à partir des lois de la géométrie optique. Les phénomènes de diffraction, en revanche, ne peuvent être décrits qu'à partir de la théorie ondulatoire de la lumière. Par ailleurs, pour simuler les effets de la diffraction, la théorie de Kirchoff est communément utilisée. Mais cette théorie n'est pas assez précise pour simuler les effets de diffraction dans les structures volumiques complexes que l'on découvre aujourd'hui chez de nombreux êtres vivants et qui sont à l'origine de leurs fascinantes couleurs. Simuler de tels effets à l'aide d'un modèle analytique calculable dans un espace raisonnable nous est apparu un défi stimulant.

La représentation spectrale de ces phénomènes dans le contexte des systèmes graphiques standards posait également un intéressant problème. Nous avons proposé un système permettant de synthétiser des images qui incluent des phénomènes représentés par une distribution spectrale dans un environnement graphique standard.

2.4 Générer des motifs de couleur

Revenons à un problème apparemment résolu, la simulation visuelle des feuilles d'un arbre. Dans le modèle prédictif de Jensen, des cartes sont utilisées pour en simuler les veines et la couleur. Dans le modèle présenté par Wang, l'apparence visuelle des feuilles d'arbre dépend également largement des cartes qui sont générées à partir de mesure (FIG. 4).

Récemment, [RFL+05] a présenté une technique permettant de simuler l'apparition et l'évolution des veines sur une feuille. Dans [RFL+05], la génération des veines d'une feuille peut être générée à partir d'un modèle biologique. Des algorithmes simulent les relations entre trois processus : le développement des veines qui dépend des sources d'hormone (auxine) qui sont dans la feuille, la modification de la distribution de cette hormone et la modification du motif et de la source de cette distribution liée à la croissance de la feuille.

Décrire la subtile variation de la couleur d'une feuille reste cependant un problème ouvert. Nous avons ainsi étudié une technique permettant de décrire de façon concomitante à leur description géométrique des motifs de couleur.

Les papillons, les poissons, les mamiphères, les fleurs, les coquillages, les dynosaures... présentent des motifs particuliers, qui évoluent d'ailleurs de façon très rationnelle en fonction de

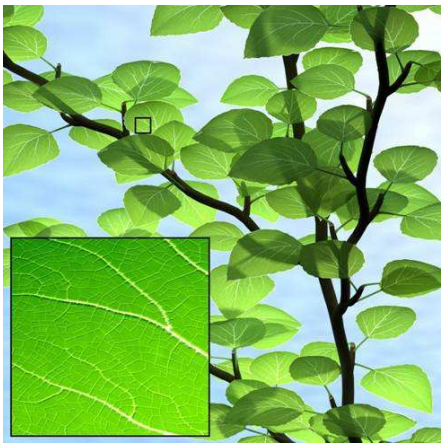


FIG. 6: Une branche de peuplier. Le modèle de nervures des kevilles e été pèné ù en utilisant un modèle de voisinage et avec deux heures avec une approximation en deux minutes [RFL$^+$05].

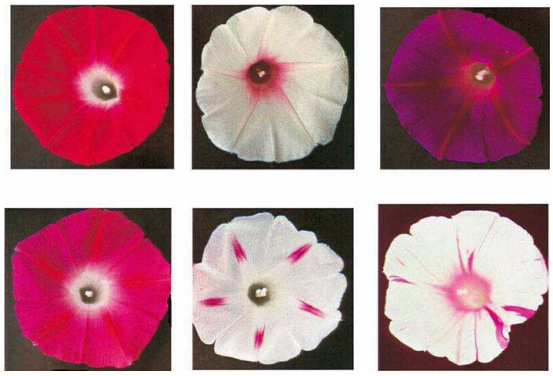


FIG. 7: Variation de couleurs dues aux génotypes et phénotypes d'une même espèce de fleur [CD00].

leur croissance. Ce phénomène a été beaucoup étudié par les zoologistes et des certains mathématiciens. La giraffe, c'est connu a une fourrure qui semble pouvoir être décrite par un diagramme de Voronoi, la fleur de tournesol semble inspirée par la suite de Fibonnacci, etc. Des modèles procéduraux ont ainsi été développés depuis plusieurs pour synthétiser des images.

Nous avons cherché à modéliser sur la base de connaissances botaniques la génération des motifs de couleur des pétales de fleurs et leur variation. Les facteurs qui gouvernent le couleurs d'un pétale d'une fleur sont très compliqués. Ils incluent les gènes, l'environnement, les interventions horticoles, les infections virales. Pour certaines fleurs, le motif de couleur est essentiellement déterminé par les gènes, mais l'environnement peut modifier ce motif. Nous avons essayé de simuler ici comment une fleur réalise le motif définit par ses gènes par différents pigments. Selon les botanistes, la pigmentation des fleurs est due aux flavonoïdes, aux carotenoïdes et aux alkaloïdes.

Le mécanisme chimique qui permet l'expression de la couleur d'un pétale est le ratio de différents types de pigments qui la composent ; le mécanisme anatomique et optique de l'expression de cette couleur est une distribution des pigments dans le pétale. La structure d'un pétale est

analogue à celle d'une feuille : elle est composée d'un épiderme supérieur, de cellules palissades, de cellules spongieuses et d'un épiderme inférieur. Beaucoup de pigments se trouvent dans l'épiderme supérieur et dans les cellules palissades. La densité des cellules spongieuses peut affecter la brillance de pétale. Plus elle est épaisse, plus le pétale est brillant. Si l'épiderme supérieur a une certaine protubérance, le pétale aura un aspect soyeux. Nous avons donc considéré qu'il était intéressant de construire un système de distribution de la pigmentation sur une carte d'épaisseur.

Beaucoup de pétales de fleurs ont seulement une couleur. Cependant, cette couleur change pendant la floraison. La couleur aura une transformation douce. D'autres pétales ont plusieurs couleurs. Ces couleurs sont combinées avec une certaine régularité ou de façon stochastique. Ce phénomène est appelé "variegation". Nous avons voulu simuler ces deux aspects. Pour les changements doux de couleur, nous avons utilisé une méthode d'interpolation. Pour les motifs de couleur très singuliers, nous avons cherché à en établir les règles. Nous avons également défini des règles précises pour la nervure, qui a une structure différente de celle des feuilles.

Le système que nous avons voulu construire permet à l'utilisateur de synthétiser librement une image de fleur tout en considérant de façon plausible les fondements botaniques de la formation des motifs de couleur dans un pétale. Cette approche est très complémentaire de celle utilisée dans [IOOI05] pour construire l'architecture d'une fleur. Nous avons voulu également que notre système puisse générer des couleurs correspondant aux variations saisonnières.

2.5 Synthétiser des écosystèmes

Synthétiser une fleur, une feuille, un insecte. Mais comment synthétiser un environnement naturel composé d'une multitude de fleurs, d'arbres et de feuilles, de papillons d'oiseaux. Comment synthétiser des écosystèmes d'une telle complexité ?

Une technique peut consister à modéliser un terrain, puis à générer des plantes modélisées à partir de modèles botaniques, à calculer les interactions lumineuses entre elles et avec le reste de l'environnement. Deussen et al. ont développé un système qui réalise ces différentes opérations. Le terrain est conçu en utilisant un éditeur graphique. La distribution des plantes est réalisée à la main, éventuellement par un simulateur d'écosystème ou par la combinaison de ces deux techniques.

Des modèles procéduraux permettant de générer différentes variétés de plantes. La complexité géométrique de la scène est réduite par une approximation dans laquelle des plantes similaires ou des groupes de plantes sont remplacés des instances d'objets représentatifs, avant que l'image de la scène soit calculée.

Les images réalisées sont impressionnantes. Mais la synthèse de telles images reste très difficile et exigeant en terme de temps de calcul et de savoir-faire. Par ailleurs, l'acquisition de nombreuses données biologiques et la construction de nombreux modèles 3D sont nécessaires.

Aussi, nous avons pensé explorer un autre domaine de techniques possibles : synthétiser des structures à partir d'images.

FIG. 8: Des plantes diverses, générées par des modèles procéduraux sont distribués dans la scène manuellement ou en simulant un éco-système.

Les textures sont abondamment utilisées en Informatique Graphique. Elles sont réalisées à l'aide de dessins ou de photographies scannées. La synthèse de textures est une méthode alternative. Etant donné un échantillon de texture, synthétiser une nouvelle texture c'est faire en sorte que l'image nouvelle qui sera générée sera perçue par l'observateur comme procédant du même processus que l'échantillon. Les défis principaux sont : comment estimer le processus de génération d'une texture à partir d'un échantillon fini ? Comment développer une procédure efficace d'échantillonnage pour produire de nouvelles textures à partir de cet échantillon ?

Nous avons orienté notre travail dans deux directions. Nous avons tout d'abord cherché une technique qui permette la synthèse de textures en temps réel. Ensuite, nous avons cherché comment préserver les apparences globales de l'image initiale dans les nouvelles textures que l'on souhaite synthétiser.

3 Etat de l'Art

3.1 Les interactions lumière-matière

Notre travail a porté sur la simulation de phénomènes optiques complexes et notre contribution a porté essentiellement sur la visualisation de scènes comportant à la fois des interactions lumineuses représentées par des spectres ou une codification RGB. Contrairement aux systèmes conventionnels qui calculent les couleurs à partir de valeurs tri-chromatiques, le système permet d'intégrer toute sorte de représentation spectrale pour calculer des effets optiques complexes, en particulier les magnifiques phénomènes d'irridescence que l'on rencontre dans la nature et qui résultent de phénomènes optiques comme la diffraction ou l'interférence. Au contraire des systèmes de représentation spectrale présentés antérieurement [DCWP02], notre système permet à l'utilisateur de calculer des scènes complexes seulement quand le besoin est nécessaire, seulement lorsqu'ils ont un un impact significatif dans le transport de la lumière et sur l'image finale.

Un grand nombre de modèles décrivent les effets optiques spéculaires qui peuvent être visua-

lisés avec des représentations spectrales. Hall [HG83] ont développé un système qui donne les composants de la lumière à partir d'un certain nombre d'échantillons aussi bien dans une représentation RGB que dans une représentation CIE. Meyer [Mey88] a proposé une approche utilisant 4 longueurs d'onde. Les chercheurs de l'Utah ont proposé une approche similaire fondée sur un échantillonnage des longueurs d'onde. Dans, Percy présente une alternative à la technique de représentation spectrale directe. Il propose une représentation fondée sur un ensemble de fonctions de base. Iehl [IP00] and Rougeron [RP97] roposent un système adaptatif. Sun [SFCD99] a proposé une méthode qui permet d'éviter la technique fixe d'échantillonnage et la représentation linéaire des couleurs proposée dans Percy. Nous avons utilisé cette méthode dans notre système.

Sun ont proposé un système qui prend en compte la représentation spectrale en entrée, prend en compte cette représentation dans le calcul des interactions lumineuses et génère une représentation spectrale convertible en RGB lors de l'affichage de l'image.

Ce système permet de gérer des phénomènes optiques comme la dispersion, l'interférence, la diffraction et la fluorescence [Sun06]. Mais il est très coûteux et peu efficace, lorsque seulement ces phénomènes occupent seulement une partie de la scène. Une autre difficulté est d'obtenir la composante spectrale de tous les objets présents dans la scène.

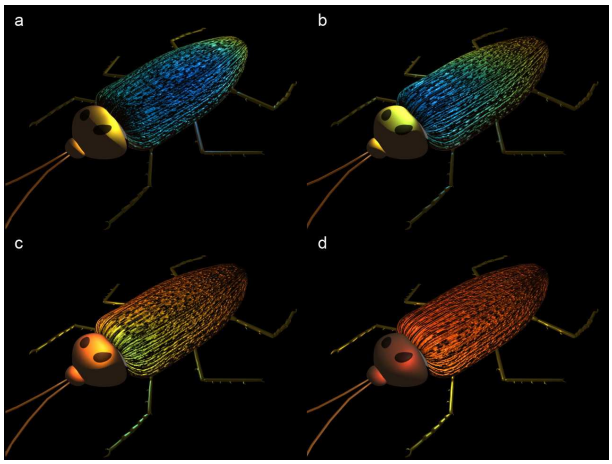


FIG. 9: Les interactions lumière-matière [Sun06].

Avec notre méthode, comme tous les calculs utilisent une représentation RGB le calcul est très rapide. En outre, comme le processus est d'autant plus efficace que la présence de phénomènes optiques spéciaux est faible dans la scène.

Les modèles que nous avons utilisés est un modèle d'interférence. La méthode est fondée sur la théorie ondulatoire et elle est capable de simuler les interactions de la lumière sur des matières multi-films. Dans Sun [Sun06], un système multi-couches est proposé,. Ce modèle empirique permet d'obtenir d'intéressants phénomènes d'irridescence, mais il est abusif de prétendre qu'il permet de simuler les effets optiques complexes produits par des structures micro-volumiques.

3.2 Générer des motifs de couleur

La simulation des motifs de couleur des organismes vivants a été étudiée par de nombreux chercheurs en biologie, mathématiques et aussi en informatique graphique. Les motifs de couleur sont très importants pour identifier l'apparence des êtres vivants : poissons, plantes, insectes. En informatique graphique des travaux ont été réalisés pour simuler les motifs de couleur de mammifères, de poissons, de coquillages, etc.

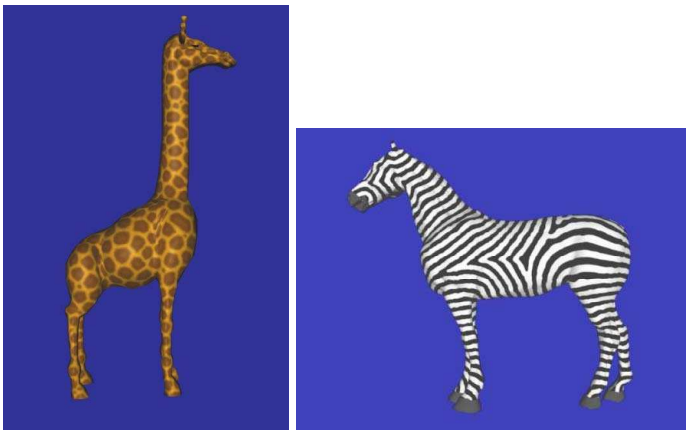


FIG. 10: Générer des motifs de couleur [Tur91].

Au lieu de tenter de simuler des motifs à partir d'une analyse strictement fondée sur des connaissances biologiques, d'ailleurs mal connues, des modèles mathématiques ont été utilisés pour simuler de façon approximative la formation de certains motifs de couleur dans la nature. Bien qu'il n'y ait pas de certitude sur leur validité, ces modèles peuvent produire des résultats visuels plausibles avec une bonne performance. Les travaux entrepris dans ce domaine s'appuient sur des équations de réaction-diffusion [Tur91]. Ils ont porté sur la simulation des motifs d'animaux à fourrure, de coquillages, de poissons ; un modèle "Clonal Mosaic" a été utilisé pour simuler la formation des motifs de couleur de certains mammifères. Un modèle de simulation fondé sur la botanique a été également proposé pour la simulation de la nervure des feuilles.

Pour synthétiser des images de plantes, des systèmes de modélisation ont été proposés. Linderman ont présenté par exemple une méthode pour modéliser des formes de feuille. Ijri ont présenté une méthode pour modéliser la géométrie des fleurs en préservant leur structure, au sens botanique. La simulation de l'apparence des feuilles a également fait l'objet de nombreux travaux portant sur la simulation des interactions lumière-matière. La formation des motifs de couleur a également un rôle crucial dans l'apparence des êtres vivants, en particulier des fleurs ou des feuilles. Notre travail a été centrée sur cette application, mais il peut être généré à d'autres espèces du vivant, comme les papillons par exemple.

3.3 Synthétiser des écosystèmes

Texture Tiling

Notre contribution dans le domaine de la génération de texture s'appuie sur un concpt connu, dit Texture-tiling. Cohen et al ont développé un algorithme stochastique pour tuilé de faàon non périodique un plan avec un petit nombre de Wang-Tiles en temps réel. Wei et al ont étendu ce travail en utilisant un GPU pour améliorer le mapping de texture basé sur ce concept [Wei04]. Ng et al ont présenté une autre approche pour générer un ensemble de petites textures à partir d'un exemple en entrée [NWT$^+$05]. Ces tuiles pourraient aussi être réunies pour synthétiser des textures plus larges. Notre méthode utilise le la technique dite ω-Tile pour définir un échantillon de texture en entrée.

Toutes les techniques présentées requièrent un ensemble de carreaux extraits d'un exemple d'entrée pour générer les motifs primaires. Aussi la qualité des textures-tiling n'est pas stable.

FIG. 11: Synthétiser des écosystèmes [WL00].

Algorithme génétique

Les algorithmes génétiques sont fondés sur une méthode stochastique pour résoudere des problèmes d'optimisation. Ces méthodes ont été appliquées avec succès en prouvant de bonnes solutions à de nombreux problèmes [LL02] ? Son avantage est du à sa capacité d'obtenir une optimisation dans un champ de recherche qui peut avoir de multiples dimensions, avec des solutions optimales locales.

Nous avons appliqué cette technique pour optimiser notre méthode de génération de texture.

Applications du Tiling

Une première application de la technique dite Texture-tiling a été décrite en [Sta97]. Glassner a présenté également des motifs intéressants de tiling : réguliers, semi-réguliers, périodiques. Des tuiles triangulaires ont été utilisées dans [NC99] pour générer une texture non-périodique sur un maillage tandis que l'approche développée dans [FL05] reformule le mécanisme Wang-Tile pour obtenir des Texture-Tiles sur des surfaces de topologie arbitraire.

Decaudin et Neyret [DN04] appliquent ce concept pour générer et visualiser des forêts en temps réel. Channey emploie la technique des Wang-tiles pour générer et animer des modèles de flux. Dans [LD99], une fonction de distribution est présentée. Une fonction de baae distribue de manière procédurale des objets sur une texture.

La composition d'images consiste à extraire un objet d'une image source et ensuite de la coller sur une cible. La technique du Graph cut [KSE+03] consiste à prélever un ensemble d'images sources et de les combiner pour former une seule image résultante. Perez et al. Utilisent une technique d'interpolation fondée sur la résolutions des équations de Poisson pour mélanger deux images avec une frontière spécifiée par l'utilisateur. Une méthode pour éditer une image de cette nature a été présentée dans [PGB03] et [JSTS06]. Nous utilisons une méthode similaire que [PGB03] et [JSTS06] pour améliorer le motif des tiles. Pour extraire efficacement un objet d'une scène complexe, nous utilisons la technique du Graph-cut décrite dans [RKB04], qui permet de segmenter une image grâce à une technique de minimisation d'une fonction énergie.

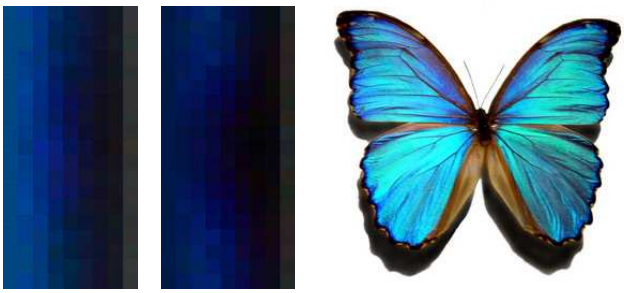
4 Contributions

4.1 Simuler des couleurs structurelles

Nous avons cherché à développer un modèle analytique qui simule la diffraction dans des structures microscopiques volumiques. Notre objectif fut motivé par l'étude de la microstructure observées dans les écailles des ailes de papillons de type Morpho. Cette espèce est assidûment étudiée par les zoologistes et les physiciens.

La microstructure volumique qui constitue l'arrangement des écailles produit une lumière d'intensité extrême dans le bleu et d'autres effets optiques spécifiques. Comme la théorie de Kirchoff n'est pas assez précise pour rendre compte des effets de la lumière dans des structures aussi complexes. Une méthode peut consister à résoudre les équations de Maxwell à l'aide de transformées de Fourrier dans une structure qui représente schématiquement l'arrangement intriqué observé au microscope électronique [Pla03]. Ce travail a été réalisé par des photoniciens, sur des microstructures simplifiées, dans le but de réaliser de telles nano-structures expérimentalement.

Nous avons testé un modèle expérimental sur une microstructure irrégulière très similaire aux microstructures observées. Ce modèle s'appuie sur les travaux de [BCM+81]. Les résultats ont confirmé que l'on pouvait obtenir des variations de couleur comme une fonction de l'incidence de la lumière. Ils sont une simplification extrême cependant des mécanismes responsables de la couleur structurelle. En réalité, la couleur structurelle du Morpho Rhetenor est le résultat combiné de plusieurs phénomènes optiques au sein des rangées de microstructures en forme de pin qui composent une écaille [KYK02]. La structure en lamelle observée dans chaque strate horizontale crée une forte interférence et des effets de diffraction, qui produisent une forte réflectivité dans une gamme de longueur d'onde, qui expliquent en partie la couleur bleue, métallique de cette espèce. Une réflectivité très élevée due aux multiples couches qui constituent chaque lamelle et le très petit espace qui sépare des lamelles adjacentes, et l'irrégularité de leur



(a) (b) (c)

FIG. 12: Couleurs structurelles obtenues à partir de simulations numériques dans une microstructure volumique (à gauche). Photographie d'un *Morpho rhetenor* (à droite).

FIG. 13: Simuler des couleurs structurelles [Don06].

hauteur produisent les pics de lumière observable pour certains angles d'incidence de la lumière. La pigmentation des écailles enfin, qui absorbe les longueurs d'ondes du vert au rouge, accentue la coloration bleue de cette espèce.

Clairement, la simulation visuelle de phénomènes aussi est apparue hors de notre portée. Nous avons donc cherché à simuler uniquement des phénomènes d'interférence, produits par exemple, lorsque la lumière traverse des couches très fines avant d'être réfléchie. Nous avons pu ainsi, à partir du modèle proposé dans [Don06] et des propriétés connues de différents films métalliques simuler des phénomènes d'irridescence.

Ces simulations sont cependant très coûteuses en tant de calcul. En outre, elles sont effectuées dans un espace, la distribution spectrale de la lumière, qui n'est pas celui de la représentation numérique standard des couleurs (RGB). Ainsi, si les phénomènes d'interférence ne se produisent que dans une infime partie de la scène, un temps infini est inutilement consacré à calculer les autres parties qui pourraient être calculées plus directement dans cet espace standard.

Nous avons proposé un système qui permet de synthétiser des images directement dans un système RGB à l'aide de ces deux représentations.

(a) (b)

FIG. 14: Générer des motifs de couleur [ZDP07].

4.2 Générer des motifs de couleur

Notre objectif était de développer une technique permettant à un concepteur de générer, en même temps que la structure et la géométrie d'un papillon, d'une fleur, ses motifs de couleur, de façon plausible par rapport aux phénomènes biologiques qui génèrent ces motifs dans la nature.

Le système que nous avons construit a les caractéristiques suivantes. L'état initial comprend, dans le cas d'une fleur, sa structure et sa géométrie, ses veines, et une distribution initiale de sa pigmentation. Ensuite, des paramètres caractérisent comment se dispersent et se combinent ces pigments. Enfin, des paramètres définissent la couleur de la fleur. Par souci de simplification, la forme est spécifiée par une image noir et blanc. La veinure est donnée comme un ensemble de nœuds dans un réseau de veines, qui inclut une veine racine ; une veine est composée de cellules capables de transportée une grande capacité d'auxine.

La distribution normalisée des pigments est représentée par une carte dans l'échelle du gris appelée carte de distribution des pigments. Chaque sorte de pigments correspond à une carte. Sa concentration en chaque point est représentée par le degré de gris du pixel. Plus sombre est le pixel, plus haute est la concentration du pigment. Des équations de réaction-diffusion décrivent l'interaction parmi différents pigments. Comme la composition du pigment de chaque cellule peut être calculée à partir de ces voisins, nous pouvons simuler la pigmentation selon un processus de croissance si un état initial et une cinématique du transport sont donnés . L'état initial peut être assigné librement par l'utilisateur.

Pour peindre la carte de distribution des pigments dans l'échelle de gris, nous construisons d'abord une base de données pour stocker les données de couleur absorbées par tous les pigments de différentes concentrations, ainsi qu'un certain nombre d'informations sur l'exposition de la fleur à la lumière. Après ces trois pré-calculs, nous calculons le motif de couleur de chaque pétale à partir de la care de distribution des pigments en temps réel.

Nous avons expérimenté ce système pour simuler de larges variétés de couleurs au sein de pétales de fleurs de même génotype, en en particulier de feuilles. Notre système peut simuler, par exemple, l'évolution saisonnière de feuilles.

Nous identifions tout d'aborsd les éléments significatifs responsables de la couleur des feuilles. La couleur d'une feuille est principalement composée par les pigments qu'elle contient, plus

FIG. 15: Générer des motifs de couleur [ZDM06].

précisément par le comportement d'absorbsion de la lumière par ces pigments. L'influence des pigments liés à la chlorophyle et de la carotine est bien connue aujourd'hui.

Il y a cependant d'autres pigments qui affectent la couleur d'une feuille en particulier en automne. Pour réduire la complexité des calculs, nous avons décidé de ne considérerque les plus importants pigments. Outre ceux mentionnés précédemment, citons l'autocyanine et le tannin. La chlorophyle est responsable de la couleur verte, tandis que la carotine crée une couleur jaune-orange. Les pigments autocyanes causent principalement la couleur rouge tandis que le tannin est la base de la pigmentation qui existe tout au long de l'année et est responsable de sa couleur brune.

Notre système simule le changement de couleur d'une feuille en trois étapes : (1) on configure l'environnement, (2) on simule l'influence du climat, enfin (3) on considère la texture de la feuille. La première étape (1) décide le temps mis par la feuille pour changer de couleur en fonction de son environnement local. La seconde (2) permet de faire les ajustements nécessaires dans l'étape (1) en considérant l'influence du climat sur le cas spécifique considéré, puis on calcule la concentration du pigment sur chaque feuille. L'étape (3) permet de générer la texture à partir des données relatives à la concentration des pigments.

5 Synthétiser des écosystèmes

Générer des scènes naturelles comme nous l'avons fait dans l'application précédent un travail difficile et extrêmement long. Dans une autre perspective, nous avons dinc travaillé sur une technique qui permet de générer des textures à partir d'un échantillon d'image. Cette technique est fondée sur l'optimisation de la qualité d'un Tile-set à l'aide d'un algorithme génétique. Notre principale contribution a été de mélanger une certaine localité définie par des mesures

d'optimisation dans une fonction globale qui peut optimiser la qualité de l'ensemble du Tileset.

Cette fonction d'évaluation équilibre les qualités parmi les Tiles et peut être optimisée par un algorithme génétique avec un temps de calcul raisonnable. Un algorithme génétique est une métode efficace de recherche d'un optimum global, qui peut automatiquement trouver et accumuler une connaissance de l'espace de recherche et contrôler de façon adaptative le processus de recherche vers la solution optimale globale.

FIG. 16: Synthétiser des écosystèmes [DZP07].

Notre approche commence avec une 8-Tiles ω-tile, telle que celle utilisée dans [NWT$^+$05]. L'algorithme peut être décrit ainsi : en précalcul, on initialise au hasard un nombre considérable d'ensembles d'échantillons à partir de l'input choisi, ensuite l'algorithme génétique trouve le meilleur. Finalement un "Graph-cut" est utilisé pour l'élimination des joints.

Dans une image, l'apparence visuelle peut être analysée par la continuité locale des Texels (Texture Elements) et les éléments perceptuels globaux tels que la variation de l'échelle d'un Texel. Ces éléments sont dus aux changements environnementaux, point de vue de la perspective, luminance, distribution des objets et géométrie de la surface. Les approches traditionnelles analysent les propriétés locales d'un échantillon donné et créent des images visuellement similaires en comparant des voisinnages locaux. Ainsi, il est possible de préserver les continuités locales des Texels. Ces approches ne sont pas sensibles cependant aux éléments de perception globale et sont limitées à des échantillons isotropiques.

Nous avons proposé une technique permettant de prendre en compte ces éléments de perception globale. L'idée est de considérer la texture désirée comme la conséquence des changements de l'environnement sur la texture isotropique. Nous avons développé plusieurs techniques interactives pour définir et extraire ces éléments à partir d'un modèle mult-modal (échelle, couleur, type), une projection multi-dimensionnelle, que nous appelons "champ d'éléments". Chaque élément est quantifié en une carte d'éléments masquée par des valeurs de couleur. De multiples cartes pourraient être intégrées ensemble comme la contrainte pour la synthèse désirée.

Nous avons démontré également plusieurs applications des cartes d'éléments que nous pouvons créés dans une texture synthétisée. Dans la figure ci-dessous, des régions de la texture doivent

(a) Image d'entrée (b) Région inconnue (c) Carte récupérée de balance

(d) Notre résultat (e) Résultat d'EBI

FIG. 17: Accomplissement d'image [DZP].

être complétées. L'image finale a été réalisée en reconstruisant les régions de la texture qui ne sont pas visibles dans l'image initiale.

6 Conclusion

Nous avons apporté plusieurs contributions originales à la résolution de différents problèmes :

– Simuler des couleurs structurelles,
– Générer des motifs de couleur,
– Synthétiser des écosystèmes.

Ces contributions ont une portée très limitée si l'on se souvient de la motivation première de notre recherche. Ainsi, au terme de ce travail de thèse, nous pourrions par exemple montrer une image réalisée avec un algorithme de simulation du phénomène d'interférence (Les scarabées seraient des scarabées 3D virtuels) et notre technique de génération de textures (projetés sur une texture synthétisée à partir d'un échantillon d'image). Nous serions loin cependant des résultats que nous avions escomptés.

L'écosystème synthétisé serait dépendant de l'échantillon de textures disponible. Les scarabées pourraient sembler s'être échappés d'une nouvelle d'Edgar Poe, mais ils auraient peu de rapport avec ceux de nos contrées. Ils seraient dispersés au hasard, et leur couleur ne dépendrait pas (plus) de leur exposition au soleil. Leur réflectance, si compliquée à calculer, serait réduite

une fonction 2D dans l'environnement texturé. En outre, nous ne savons que simuler des couleurs structurelles qu'à partir d'un modèle d'interférence, à l'aide d'une technique classique de l'optique géométrique, et pour les seuls paramètres dont nous disposons, par exemple, l'or, ou l'argent. En réalité, la simulation des effets optiques de tels objets naturels, lorsqu'elle a été élucidée par les physiciens, reste un problème ouvert pour les ingénieurs, et pour l'instant hors de notre portée.

Le parterre de fleurs pourrait être simulé en utilisant notre technique de "génération de motif". Pour générer les feuilles d'un arbre, un bouquet ou fonction de un tapis de fleurs, par exemple, avec de subtiles nuances de couleur correspondant à un génotype, un phénotype, ou fonction de leur maturité, nous pouvons par exemple créer un Atlas de textures, et y prélever les composants de manière stochastique. Nous avons démontré l'utilisation possible de notre technique pour construire un paysage. Mais la génération de ce paysage par des méthodes procédurales reste très longue à calculer. Notre principale contribution a porté finalement sur la synthèse de textures. La technique que nous avons proposée n'est pas une approche nouvelle, mais elle optimise sensiblement les techniques qui ont été précédemment présentées.

On peut cependant considérer les résultats que nous avons obtenu par rapport à leur portée plus générale.

– Le système que nous avons présenté dans [Don06] est basée sur une technique simple, mais c'est aujourd'hui la plus efficace, à notre connaissance pour générer des images dans le contexte du standard RGB, où interviennent des phénomènes qui nécessitent le calcul de distributions spectrales.
– Le modèle de génération de motifs de couleur publié dans et actuellement implanté dans le système de génération et de croissance des plantes (Green Lab) développé par le CIRAD et l'INRIA.
– Enfin, la supériorité de notre technique de synthèse de texture a été validée par plusieurs publications ; elle est une alternative à la technique présentée dans [ZZV+03]. Ses applications dans le domaine de la photographie offrent de nombreuses possibilités.

Publications

Simuler des couleurs structurelles

[1] Weiming Dong. Rendering optical effects based on spectra representation in complex scenes. In Computer Graphics International, pages 719-726, Hangzhou, China, 2006.

Générer des motifs de couleur

[2] Ning Zhou, Weiming Dong, and Xing Mei. Realistic simulation of seasonal variant maples. In PMA '06 : Proceedings of the Second International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications, Beijing, China, 2006.

Synthétiser des écosystèmes

[3] Weiming Dong, Ning Zhou, and Jean-Claude Paul. Optimize tile-based texture synthesis. In Proceedings of Graphics Interface 2007, Montréal, Canada, 2007.

[4] Weiming Dong, Shuangxian Sun, and Jean-Claude Paul. Optimal sample patches selection for tile-based texture synthesis. In CAD-CG '05 : Proceedings of the Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05), pages 503-508, Washington, DC, USA, 2005. IEEE Computer Society.

[5] Hongwei Sun, Kwok-Yan Lam, Siu-Leung Chung, Weiming Dong, Ming Gu, and Jiaguang Sun. Efficient vector quantization using genetic algorithm. Neural Comput. Appl., 14(3) :203-211, 2005.

Introduction

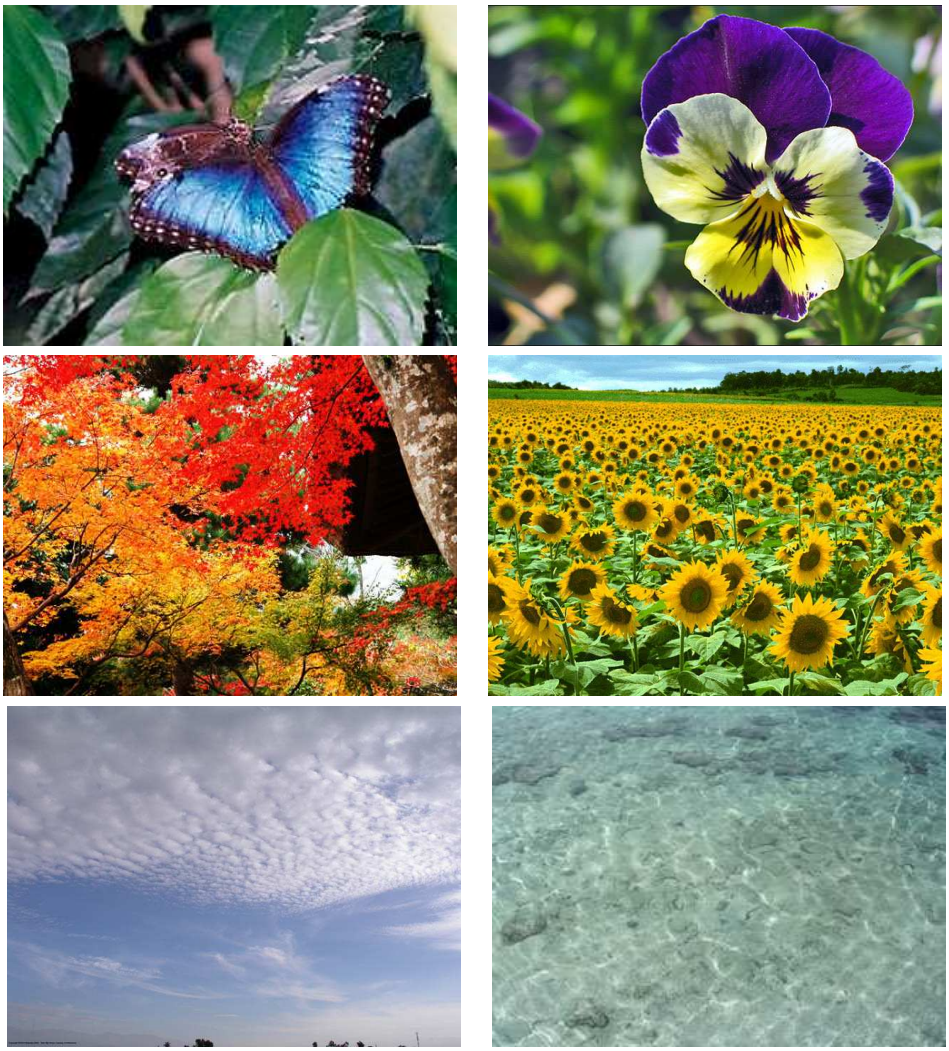


Figure 1: Richness and complexity of objects and phenomena of illumination, geometric complexity, patterns and reflectance found in natural scenes.

Image synthesis of natural scenes is an important and challenging problem. The appearance of natural phenomena has always been intriguing. The fascination is evident from the vast amount of depictions of natural phenomena created, ranging from early cave painting to impressionistic masterworks and photography.

The realistic rendering of natural scenes has many practical applications. Many industries, from entertainment to architectural design, are using computer generated imagery of natural scenes for their purposes. Therefore, it is important to design convincing visual simulation for natural scenes. For example, in drive simulators, it is critical that the virtual objects such as plants, clouds, terrain be carefully depicted because they serve as vital visual cues to a driver. On the other hand, the entertainment industry requires control in order to create desired effects. Therefore, the simulation of natural phenomena is subjected to two constraints: visual consistency and user control.

While scientists need to predict and simulate the behavior of certain phenomena, artists want to choreograph it in order to create a desired mood or effect. Movies require special effects not commonly found in nature or occurring very rarely. Even if the real phenomena can be found, controlling it can be difficult and cumbersome task. This makes modeling and rendering even more challenging problem. While purely physically-based simulation would yield a predictable result, the number of parameters controlling the simulation is unwieldy and unintuitive for non-expert users.

Significant progress has been made in the last decade in understanding how to generate realistic rendering of indoor scenes. The general approach is to analyze the physics of light transport in such environments and then to embody approximations to the physics in computational algorithms. Correct modeling of illumination and material properties is vital. It is now known that a sense of realism depends critically on accounting for shadows, secondary illumination, and non-uniform reflectance functions. Accurately approximating the effect of these properties involves great computational expense. As a result, methods for rendering realistic imagery almost always exploit assumptions about the nature of the geometric structure and illumination and materials properties likely to be encountered. Most of these assumptions derive from a presumption of indoor environments.

Natural scenes, especially outdoor scenes, present very different computational characteristics than indoor scenes. While the physics is the same, geometry, illumination, and reflectance properties are all distinctly different. Many of the techniques developed to support realistic rendering of indoor scenes require substantial modifications for natural, outdoor environments. The most difficult computational problem to overcome is the need to be able to aggregate the effects of micro-structures into large enough units that they can be rendered effectively, while at the same time preserving key aspects of visual appearance. This problem exists across a wide range of scales, ranging from foliage, in which a collection of individual leaves generates a collective appearance that is quite different than that of the constituent members, to distant landmarks, where detail must be suppressed without removing those properties that make landmarks distinctive and thus useful. In the microcosmic world, various pigmentations and tissue structures cooperate together to generate different flower patterns or the brilliant structural colors of some insects.

1 Why natural scenes?

There are many reasons why study appearance, illumination and light transport in natural environments:

- **Great beauty.**
- **Geometric, material, pattern and illumination richness and complexity.**
- **Many modes of light transport.** Light transport in natural environments is extremely complex as it varies from simple modes that can be described by traditional shading paradigms to very complicated modes that require correct physical description and full simulation.
- **Challenging to many classic algorithms.** Due to complex interactions between geometry, illumination and material properties, many "traditional" algorithms cannot handle natural environments adequately. Many existing models merely apply methods devised for "indoor" environments to natural scenes. Most often visual appearance of rendered natural scenes is not adequate. Furthermore, computational demands of natural scenes are overwhelming and "smarter" methods are needed to cope with complexity and computational demands.
- **There has been lots of research focusing on man-made materials and indoor environments, but not that much on natural and outdoor environments.**

Figure 1 shows the richness of illumination, geometric complexity, patterns and reflectance.

2 Motivation

Renderings of natural outdoor scenes have had a cartoon-like quality that significantly distracts from a sense of realism. Partially, this is due to computational and source data constraints that limit the geometric complexity of scenes to be rendered. On the other hand, illumination variety, complex reflectance functions and complex and not fully understood interactions between geometric complexity, illumination, pattern and material properties has also severely limited realism of natural scenes.

Why is the quality of computer generated images of the natural scenes inadequate?

- Source data and current computational constraints limit the geometric complexity of the environment.
- Illumination plays an equally important role in creating a sense of realism.
- We do not yet fully understand interactions between geometry, illumination, patterns and material properties.
- Some visual features of natural objects are difficult to be feasibly defined and duplicated.

Figure 2: A photo with several kinds of natural objects.

3 Problems

As shown in Figure 2, usually there are many kinds of natural objects in an outdoor scene. We research on how to synthesize similar realistic images from different aspects. The objective of this thesis is to develop algorithms which contain two properties: natural appearance and interactivity. The property of natural appearance assures that the objects rendered by our algorithms appear their real natural characteristics in the images. On the other hand, the interactivity means that with our systems, the users can conveniently adjust the parameters and generate results according to different requirements at interactive or real-time rate.

3.1 Natural Appearance

The appearance of natural scenes addresses many possible directions for researchers, such as object shapes and small-scale geometries, complex lighting effects, color patterns, large-scale environments, and so on. We start from the natural appearance of Figure 2 and mainly concentrate our work on three parts: light and matter (the lighting effects on the beetles), object color patterns (flower patterns) and the generation of large-scale ecosystems (the background with cluster of flowers and leaves).

Light and Matter

The human observer routinely has to deal with objects that are far from Lambertian. Many objects ubiquitous in the daily environment strongly deviate from Lambertian or Phong. Natural materials such as biological objects (leaves, flowers, insects), food (milk, fruits), or inorganic objects (sky, water, snow, clouds, weathered materials, rocks) exhibit significant subsurface or volumetric light transport and even more complex optical effects, namely interference, diffraction, fluorescence, dispersion, phosphorescence, etc.. These physical effects are very important

for realistic depiction of materials [Sta99, Sun06] and scenes [Don06]. Many applications ranging from flight simulators and artistic design rely on subtle lighting effects and cues that often cannot be described by simplifying the light transport equation and without including effects of spectrum-driven interactions and global illumination for some natural objects.

For example, iridescent material frequently occur in nature, such as soap bubbles, film-coated articles, and hummingbird feathers [WC83,Nas01]. Such colors arise from light interference, an optical phenomenon resulting from coherent superposition between lights. Since interference colors depend not only on illuminations and objects but also on viewing directions, they vary subtly and continuously over objects and usually create sensational visual effects. Because light interference relies on full spectral information, a purely color-based approach (such as in the red, green, blue-RGB-models) will not suffice to render this phenomenon.

Illumination and material appearance are at the heart of computer graphics. At the lowest level they are controlled by physically optical events that are computationally expensive to model, hard to understand and cumbersome to control in practical applications. This is especially true for illumination and appearance in the natural outdoor environments. As discussed in [SFCD99], a natural solution is to adopt a spectrum-based approach which describes lights, objects, and their interactions in terms of spectra. However, because color-based techniques is a standard in industrial rendering software and graphics platforms, we need to retain the color-based approach and add the effects of physically optical effects. This goal motivated our research work, a feasible rendering framework which can integrate both RGB-based and spectrum-based materials will be useful to efficiently generate realistic images.

Accurate computation of light-material interaction is therefore very complex, computationally expensive and sometimes hard to control and understand for an inexperienced users. For image synthesis purpose, approximations with intuitive parameters may often be enough to capture the appearance of almost any material.

Color Patterns

Natural color patterns, consisting of various pigmentation and venation influence, is one of the most beautiful figures in the natural world. It poses an interesting and important challenge for modeling and realistic rendering in computer graphics. For example, when light interacts with flower petals, several optical effects occur, namely reflection, transmission and absorption. The color we observe on the flowers is formed by the reflective light from the petals. However, it is not reflection but absorption which is mainly responsible for the appearances of the flowers [Goo76]. Pigments in the flower petals absorb light of certain wavelengthes. Different pigment has specific light absorption spectrum, which is mainly determined by its molecule structure. Their absorbance and distribution decide flower color patterns. Apparently, to accurately simulate the flower color patterns, the key point is the modeling of pigment distributions in the flower petals. Our work in this area is concentrated on the generation of pigment distribution maps. On the other hand, the simulation of the seasonal plant appearance variation is also a challenging problem, we will also explore it in this thesis.

Large-Scale Ecosystems

The large-scale ecosystem is very important in outdoor scenes. It performs as the background of a scene and generates grand visual appearance in the simulating results.

For some natural materials, the physical complexities are out of current simulating capabilities. On the other hand, because of the high complexity of the real world, realistic simulation of natural scenes is very costly in computation. The topographical subtlety of common natural features such as trees and clouds remains a stumbling block to cost-effective computer modeling. To solve these problems, one way is to use gathered data especially natural textures, to decorate the scenes with synthesized patterns.

Texture is a ubiquitous experience. It can describe a variety of natural phenomena with repetition, such as sound (background noise in a machine room), motion (animal running), visual appearance (surface color and geometry), and human activities (our daily lives). Since reproducing the realism of the physical world is a major goal for computer graphics, textures are important for rendering synthetic images and animations. We will use textures to generate large-scale ecosystems in rendering natural scenes. However, because textures are so diverse it is difficult to describe and reproduce them under a common framework.

3.2 Interactivity

In virtual environment construction of natural scenes, interactive or real time control to the objects in the scene are needed. The interfaces of rendering systems for natural scenes should be friendly to users in generating different results. Users could adjust the parameters to simulate the abundant appearances of natural objects. At the same time, the efficiency of the system is also very important.

For realistic rendering of complex lighting effects, usually it is not easy to adjust the parameters of the lighting functions to get a satisfactory results, the efficiency of the rendering framework is very important. Users should be allowed to get a rendering results at interactive rate for a scene which is not very complicated. Our work in this area is concentrated on this point. We tries to develop a more efficient system for realistic rendering, especially for the scenes where spectrum-based calculations are needed.

For color patterns simulation, the interactivity of the system is also important. The patterns of natural flowers and leaves are very abundant and vary rapidly among different kinds of them. Our goal is to develop a user-friendly system which allows users to interactively control the simulating process to generate various color patterns.

We use natural textures to synthesize the large-scale ecosystems. For this work, we also pay attention to the interactivity of the algorithms. We presents texture synthesis systems which uses interactive optimization framework to increase the results quality. We also provide methods which could interactively help to preserve the sample features in the results or generate more vivid appearances.

4 Contributions

We study the problem of realistic rendering of natural scenes from two ways: biology-based simulation and image-based synthesis.

First, we use biology-based methods to simulate the complex lighting effects and color patterns on natural objects. We build a rendering framework which allows computing full spectra light object interactions only when it is needed, i.e. for the part of the scene that requires simulating special spectra sensitive phenomena. An interference-based "Multilayer Film" model is implemented to prove the efficiency of our system [Don06].

For color patterns, we focus our work on the simulation of patterns on plant. We present a biologically-motivated algorithm for modeling and visualization of flower color patterns [ZDP07]. It is able to produce various flower color patterns with little user interaction. To make our algorithm to be widely used, we also develop a biologically-motivated system of seasonal variant scenes generation for maples, which has a obvious leaf color transformation during the time [ZDM06].

On the other hand, for large scale ecosystems, we use image-based methods to realistically decorating them. Natural texture is the key point in this field. First, for real-time rendering applications, we have developed optimized tile-based texture synthesis methods for textured background generation. ω-tiles [NWT$^+$05] are selected as the basic tiling patterns, then we present *Genetic Algorithm*-based optimization algorithms to assure the synthesis qualities [DSP05, DZP07].

To protect some global appearances of natural textures, we present an interactive tool for anisotropic 2D texture synthesis guided by feature maps, that preserves some global perceptual characteristics of the example [DZP]. With our system, the local continuities and global properties of the samples could both be protected in the synthesis results.

5 Organization

The rest of the thesis is organized as follows. In Chapter 1, we introduction some background knowledge of our work and explore each part of the thesis. In Chapter 2, we present an efficient framework for rendering natural scenes with spectral objects. In Chapter 3, we develop an efficient system for modeling and visualization of flower color patterns. In Chapter 4, we extend the algorithm in Chapter 3 and apply it to the seasonal leaves color variation. In Chapter 5, we present an approach for tile-based texture synthesis that is based on the optimization of tile set quality within a *Genetic Algorithm* (GA)-based framework. In Chapter 6, we provide a framework for perceptual-feature-aware texture synthesis, which can preserve the global appearances of input samples in the synthesis results. Finally, we conclude this thesis and describe future work.

Chapter 1

Background and Work Exploration

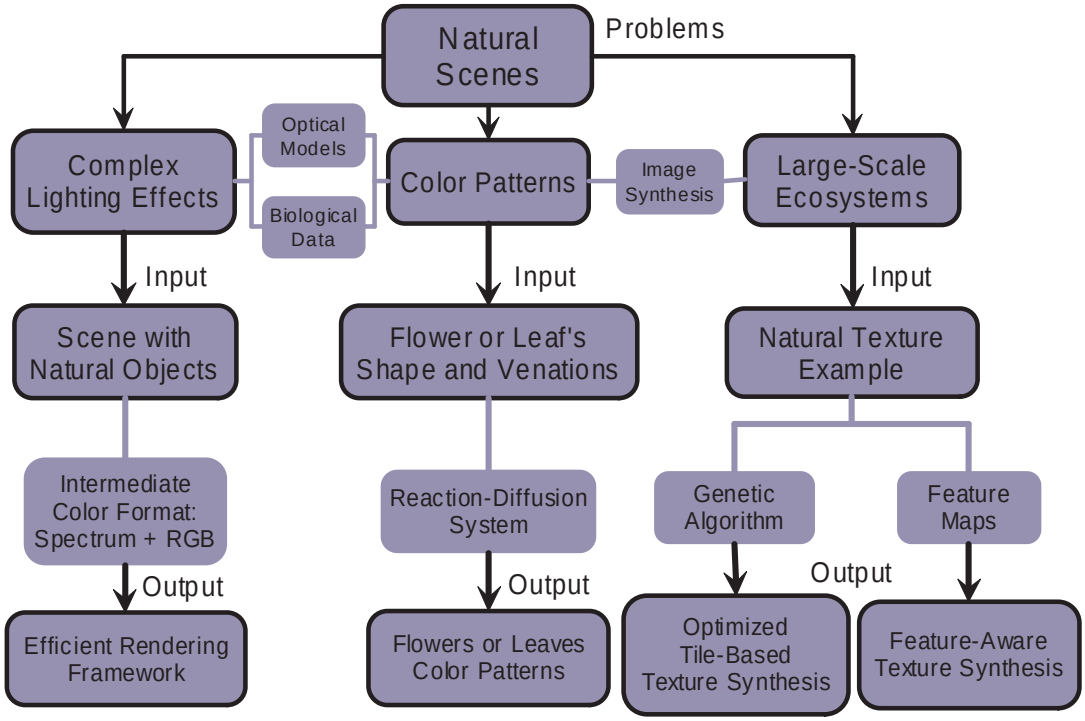


Figure 1.1: Thesis overview.

In order to improve the appearances of objects, complex lighting models and texture functions now replace the simple diffuse Lambert law implemented in early systems. These models take complex light-matter interaction schemes. Simulating the appearances of natural objects is still difficult today, however. There is no need to model the interior of a material in order to predict an appearance that can be captured. However, these measured data need to be available. On the other hand, adapting biologically complex multi-layers structures and computing the optical light and matter interaction in such structures is highly time consuming, but we have tried to follow that way. We tested our lighting models in a new efficient rendering framework. Our framework is more accurate and more efficient than previous concurrent frameworks. However, when the natural scenes become very complex with the increasing of the objects number, it will be difficult to model and arrange them in the scene to achieve a brilliant appearance. At the same time the rendering process will also be very time-consuming. For these scenes especially large scale ecosystems, we employ image based methods instead of biological models to synthesize the parts with many repetitive objects.

In this chapter, we introduce the background information of both biology based methods and image based synthesis. Then we briefly explore the new frameworks and algorithms presented in the thesis.

1.1 Light and Matter

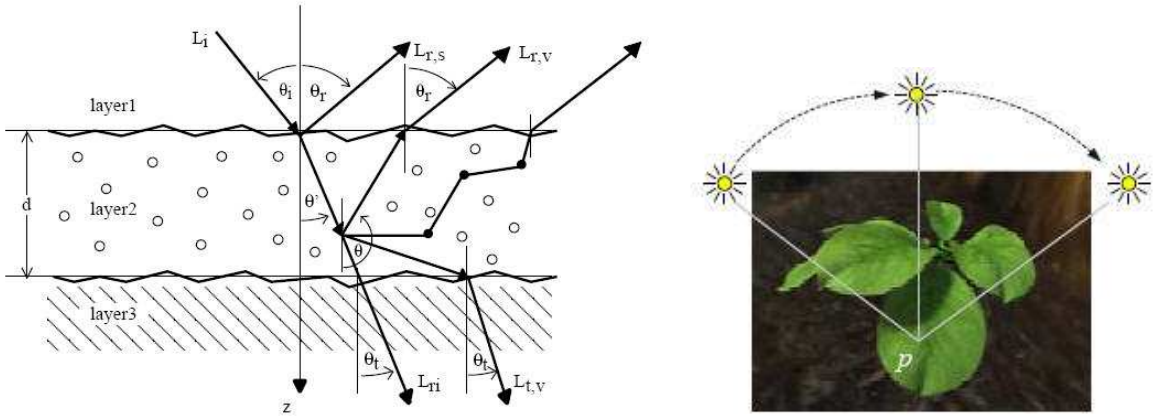
Appearances of objects are widely related to light and matter interactions [Gla95]. It is important to understand the specific physical foundation when we want to build a model for a lighting effect.

1.1.1 Scientific Foundations

In the real world, the reflectance can be expressed by a 12D function. Its parameters include x: the entering or exiting position of light; $\vec{\omega}$: incident or reflect angle; λ : wavelength; t: the time. We usually consider enter and exit happen at the exactly same moment t, and the wavelength is invariable, so we can omit these two parameters to get the 8D BSSRDF (Bidirectional Surface Scattering Distribution Function) model, which can be written as

$$S(x_i, \vec{\omega}_i, x_o, \vec{\omega}_o) = \frac{dL(x_o, \vec{\omega}_o)}{d\Phi(x_i, \vec{\omega}_i)} \tag{1.1}$$

One example which uses BSSRDF is the rendering of leaves. As shown in Figure 1.2, Light enters into the leaf, passes through the surface and reflects into the material before it comes back out, so the entering point and exiting point are different from each other. Then if we consider that light entering and exiting occur at the same point, $x_i = x_o$, we can get the 6D BTF (Bidirectional Texture Functions) or SVBRDF (Spatially-Varying Bidirectional Reflectance Distribution Function) model by omitting the exiting position x_o. If we only consider the relative position of light entering and exiting, we can use $(x_i - x_o)$ instead of x_i and x_o to get the 6D



(a) The geometry of scattering from a layered surface [HK93]. (b) Realistic rendering of leaves [WWD+05].

Figure 1.2: Rendering leaves with reflection functions.

BSSDF (Bidirectional Surface-Scattering Distribution Function). Moreover, if we suppose that the reflectance is independent of the position, we get the most popular 4D BRDF (Bidirectional Reflectance Distribution Function) model. In another thought, from the BTF or SVBRDF, we fix the incident angle of light, and then the reflectance is determined by the location of the object. It is the 4D light field model. By ignoring the effect of incident and reflect angle, we will finally get the 2D texture mapping method. Besides, there are many other reflectance models. Although these models consider different factors, however, they are all simplifications of the original 12D model.

1.1.2 Spectral Functions

To accurately render the physically-based optical effects like interference, diffraction, etc., we need to represent the light sources and materials of the objects with spectra rather than simple 3D color models (for example RGB model). A physical property that varies with wavelength is called a *spectral function* or simply a *spectrum*. The graph of a spectral function is called a *spectral curve*. Usually, spectral functions in the visible range are sufficient for realistic image synthesis. But sometimes a wider range such as the ultraviolet region may be involved when fluorescence occurs.

Spectral Power Distribution

A *spectral power distribution* (SPD) is defined as the power (energy per second) of a light ray in unit wavelength in a unit area perpendicular to the propagating direction. Thus, a SPD is actually the light intensity that is defined as the amplitude of the Poynting vector, except that the wavelength dependency is more emphasized in SPDs. In this thesis, we will use SPDs and light intensities as equivalent terms.

SPDs are essential for both computation and analysis in realistic image synthesis. SPDs describe

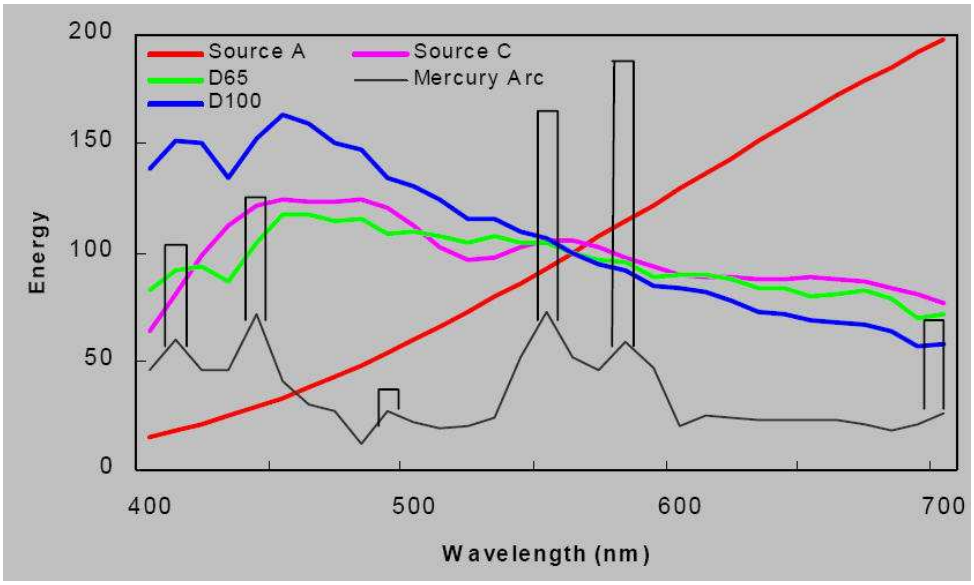


Figure 1.3: SPDs of common CIE standard light sources [Sun00].

the energy for all light paths and determine the colors of the created image. Besides, SPDs are the basis for discussing spectral properties of physical surfaces as well as material volumes.

Figure 1.3 displays the SPDs of common CIE standard light sources. The specific value of light sources and some natural materials we use in our programs can be found in [Sun00]. Note that the spectral curves contain a number of sharp and strong peaks (spikes). The spikes usually convey a significant amount of energy within very narrow bands of wavelength, it is important to represent them accurately in order to generate faithful colors when rendering a computer image.

Spectral Reflectance

A *spectral reflectance* or simply *reflectance* describes light reflection at an object boundary or an interface between two media. Denoted by $R(\lambda)$, it is defined as the ratio between the reflected and the incident light intensity (or energy)

$$R(\lambda) \equiv \frac{I(\lambda)}{I_0(\lambda)} \tag{1.2}$$

where $I(\lambda)$ and $I_0(\lambda)$ are the reflected and incident intensities. Spectral reflectance apply to boundaries of all kinds of materials, including opaque, transparent and translucent.

Usually a spectral reflectance is independent of the intensity of the incident light and is an intrinsic property of the material. This property, which we call the *spectral linearity*, allows for the application of the reflectance data measured for one illuminating condition to other conditions. This means that, given the reflectance $R(\lambda)$ and the incident light intensity $I_0(\lambda)$, we can determine the reflected light intensity using

$$I(\lambda) = I_0(\lambda)R(\lambda) \tag{1.3}$$

Note that although Equation 1.2 and Equation 1.3 appear mathematically equivalent, they convey different meanings; Equation 1.2 is a definition while Equation 1.3 is based on the spectral linearity. Therefore we intentionally write them separately.

It should be pointed out that the definition of Equation 1.2 is meaningful only when the incident and reflected lights are concentrated rays (which implies that the surface must be smooth) and the effect of the incident angle is negligible. If the reflected light is scattered into many directions such as from a rough surface or the incident angle significantly affects the reflected SPD, a spectral reflectance does not suffice to characterize the material surface and we need a BRDF. However, spectral reflectances are useful for deriving BRDFs. As shown above, a reflectance is an intrinsic property of the material (a unique feature associated with a material), but a BRDF also depends on the surface geometrical details. For such a physical surface, a BRDF is a consequence of the combination of the material's reflectance and the surface geometry.

Spectral Transmittances

As the counterpart of a spectral reflectance, a *spectral transmittance* or simply *transmittance* refers to the ratio between the transmitted and incident energies

$$T(\lambda) \equiv \frac{I(\lambda)}{I_0(\lambda)} \tag{1.4}$$

where $I(\lambda)$ and $I_0(\lambda)$ are the intensities of the transmitted and incident lights. A spectral transmittance is particularly useful in predicting the result after a light ray passes through a thin, transparent layer such as a filter [BW75, WS76]. That is, we can use

$$I(\lambda) = I_0(\lambda)T(\lambda) \tag{1.5}$$

to compute the light intensity after the transmission. Similar to the case of reflection, when an object boundary is rough (i.e., light is scattered into many directions) or the dependency of the incident angle is significant, we need a BTDF.

1.1.3 Colors

Colors are sensations that light produces on the human eye. A color is characterized with three independent components called the *tristimulus values*. This property originates from the three cones in the human eye called the S, M and L cones, which are named respectively for their peak responses to relatively short, medium and long wavelengths [Gla95, Mey88]. Regular cameras also have three sensors to generate the tristimulus values and the corresponding processes can be viewed as a simulation of human vision. The measurements and specifications of colors as well as transformations among different color systems are topics of *color science*. The following review focuses on the color models most relevant to this thesis.

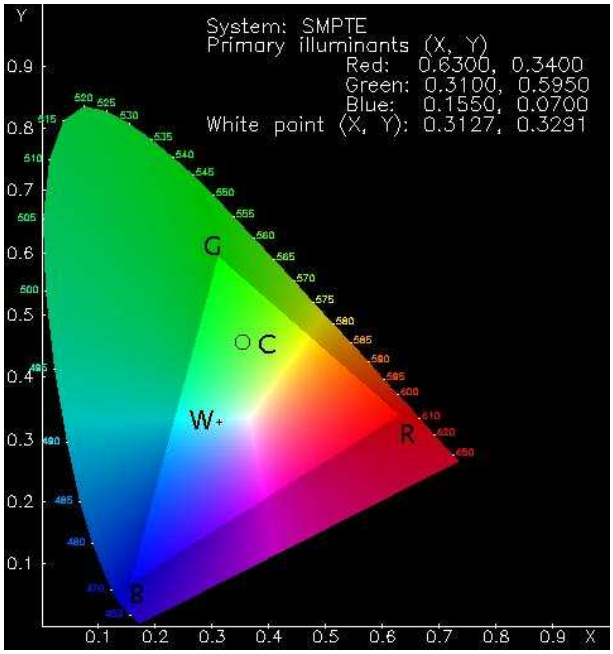


Figure 1.4: CIE chromaticity diagram.

CIE XYZ Model

The commonly adopted color standard was created by the CIE (Commission International d'Eclairage) based on measurements of a large number of human observers [WS76]. The average result of the measurements is presented through three functions called the *CIE color matching functions*, denoted by $\bar{x}(\lambda)$, $\bar{y}(\lambda)$ and $\bar{z}(\lambda)$, respectively (their data and profiles are presented in [Sun00]). Using these color matching functions, the tristimulus values X, Y, Z for light intensity $I(\lambda)$ are

$$\begin{cases} X = \kappa \int_{\lambda_{min}}^{\lambda_{max}} I(\lambda)\bar{x}(\lambda)d\lambda & \text{(1.6a)} \\ Y = \kappa \int_{\lambda_{min}}^{\lambda_{max}} I(\lambda)\bar{y}(\lambda)d\lambda & \text{(1.6b)} \\ Z = \kappa \int_{\lambda_{min}}^{\lambda_{max}} I(\lambda)\bar{z}(\lambda)d\lambda & \text{(1.6c)} \end{cases}$$

where $[\lambda_{min}, \lambda_{max}]$ defines the visible range and κ is a constant given by

$$\kappa = \frac{100}{\int_{\lambda_{min}}^{\lambda_{max}} I_w(\lambda)\bar{y}(\lambda)d\lambda} \tag{1.7}$$

such that Y is 100 for standard white [FvDFH96, WS76]. Moreover, $\bar{y}(\lambda)$ is chosen to exactly match the *luminous efficiency function* for the human eye (i.e., it is the weight function for photometry) such that the Y value corresponds to the perceived luminance.

If we establish a three-dimensional Cartesian frame with the tristimulus values X, Y, Z as its coordinates, any color can be specified as a vector

$$\mathbf{C} = X\hat{\mathbf{X}} + Y\hat{\mathbf{Y}} + Z\hat{\mathbf{Z}} \tag{1.8}$$

and this space is called *CIE color space*. To study the chromaticity of colors, it is adequate as well as convenient to use the relative coordinate values called the *chromaticity values*, defined by

$$\begin{cases} x \equiv \dfrac{X}{X+Y+Z} & \text{(1.9a)} \\ y \equiv \dfrac{Y}{X+Y+Z} & \text{(1.9b)} \\ z \equiv \dfrac{Z}{X+Y+Z} & \text{(1.9c)} \end{cases}$$

Clearly these relative coordinates correspond to the tristimulus values on the $X+Y+Z=1$ plane. The advantage of using the relative coordinates is the separation of the chromaticity and luminance of any color; we can use x and y to specify the chromaticity and use Y to specify the luminance. Thus the tristimulus values can be calculated by

$$\begin{cases} X = \dfrac{x}{y}Y & \text{(1.10a)} \\ Y = Y & \text{(1.10b)} \\ Z = \dfrac{z}{y}Y = \dfrac{1-x-y}{y}Y & \text{(1.10c)} \end{cases}$$

from the specification of (x, y, Y).

Figure 1.4 displays a chromaticity diagram in the two-dimensional space of x and y. The entire area of perceivable colors is divided into many regions corresponding to different descriptive terms of color. The white colors are near the center. The colors for monochromatic lights are along the curved locus (called *spectral locus*), where the marked values are the wavelengths in $\mu m(10^{-6}m)$. A curve is demonstrated for blackbody radiations along with values of the *color temperatures* [WS76]. Finally, the diagram provides the locations and chromaticity values for a number of standard CIE light sources.

While colors are associated with many physical and perceptual properties, some are of particular importance for realistic image synthesis.

- The relation between a SPD and its corresponding color is *linear* - adding two SPDs or multiplying a SPD by a constant is equivalent respectively to adding the corresponding colors or multiplying the color by the constant. As a matter of fact, this linearity is an important basis for color-based rendering.
- Colors have *metamers* - two different SPDs can have the same color. From Equation 1.6, we find that the process of calculating a color from a SPD is actually a projection from the infinite-dimensional space of spectra (where each wavelength can be regarded as an independent abscissa) into a three-dimensional space of colors (where each color

component is an independent abscissa). Therefore, given a set of X, Y, Z, in principle there exist an infinite number of SPDs that can generate the same tristimulus values. (As an analogy, an infinite number of points in a 3D space can project onto the same point in a plane.) In [WS76] we can find many examples of metamers. Thus, it is impossible to exactly recover a spectrum from its color. This principle has an important implication when we derive spectra from colors for spectrum-based rendering.

- Perceptually colors are often described in terms of *hue*, *saturation*, and *value*, which correspond to the spectral dominant wavelength, spectral wavelength purity, and luminance, respectively. In Figure 1.4, the colors on the boundary correspond to different hues and are saturated, and the color at the center (which is white) is unsaturated. The specification based on hue, saturation, and value is called the HSV model.
- CIE color space is not perceptually uniform.

RGB Model

Although the CIE XYZ model is complete (i.e. capable of representing all perceivable colors), it is not convenient for specifying a color on a CRT generated by overlapping the red, green and blue primaries, which are denoted by R, G, and B, respectively. In other words, we are more interested in the relation between any generated color and the weights of the red, green and blue primaries.

Here the linearity of colors is the basic principle that we rely on. Specifically, assume that $I_1(\lambda)$ and $I_2(\lambda)$ are two SPDs and their colors in the CIE XYZ space are $C_1 = (X_1, Y_1, Z_1)$ and $C_2 = (X_2, Y_2, Z_2)$. The linearity states that for any non-negative constants α and β , we have

$$\kappa \int_{\lambda_{min}}^{\lambda_{max}} [\alpha I_1(\lambda) + \beta I_2(\lambda)]\bar{x}(\lambda)d\lambda = \alpha X_1 + \beta X_2 \tag{1.11}$$

and similar equations hold for Y and Z. In particular, when $\alpha + \beta = 1$, $\alpha C_1 + \beta C_2$ represents a point on the line segment formed by C_1 and C_2. Then, Equation 1.11 implies that any color on the line segment can be generated by a linear combination of C_1 and C_2. Extending this result, any color C within the triangle formed by R, G, and B can be generated by the linear combination

$$\mathbf{C} = r\mathbf{R} + g\mathbf{G} + b\mathbf{B} \tag{1.12}$$

where r, g and b are non-negative constants and satisfy

$$r + g + b \equiv 1 \tag{1.13}$$

If we remove the constraint of Equation 1.13, Equation 1.12 generates all colors within the tetrahedron formed by R, G, B, and the origin (the perfect black). Note that we must maintain the condition that r, g and b are non-negative as a CRT cannot generate a negative weight. The (r, g, b) specification is called the *RGB model*.

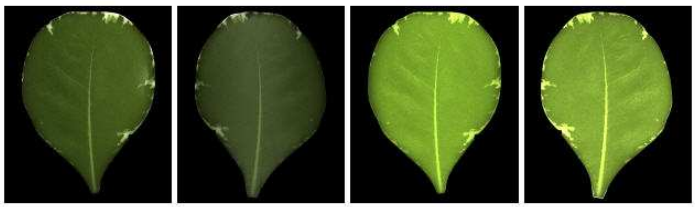


Figure 1.5: A layered leaf lit from front and behind [DJ05].

1.1.4 Multi-Layered Scattering

Some objects in nature are more complex: leaves for example have thin slabs (Figure [DJ05]). Generally, any light entering a material will either be absorbed or return to the surface, but this assumption breaks down for thin slabs. Diffuse light transmitted through the slab does not return and the upward diffuse radiance is equal to the reflected downward radiance at the bottom surface.

Donner and Jensen [DJ05] introduced a model for light diffusion in multilayered translucent materials and presented a technique based on multiple dipoles to account for diffusion in thin slabs. Then, they enhanced the multipole theory to account for mismatching indices of refraction at the top and bottom of translucent slabs, and to model the effects of rough surfaces. A micro-facet model wased used to describe the roughness of the surface, and they model the surface reflection using a Torrance-Sparrow BRDF. The model still has some limitations. For leaves, for example, the authors need to apply both thickness and bump maps on the geometry to simulate the appearance of the leaf veins. The thickness map is effectively used as a displacement map, which increases the distance of the diffused lighting. This gives the effect of thickness, but it is only an approximation as the overall reflectance profile changes as a function of the thickness and specific properties of each layer. For simulating diffraction effects, the Kirchoff scalar theory is commonly used, but the scalar theory is not accurate for complex volume microstructure that characterized most of the structural colors in nature [Pla04].

On the other hand, Wang et al. [WWD$^+$05] proposed a parametric model for thin surfaces (leaves) in terms of spatially variant 6D BRDFs and 6D BTDFs. These functions take into account of the main scattering behaviors inside leaves (the slab interior is assumed to be homogeneous) and rough surface scattering is considered on leaf surfaces. More important, they formulated the reflectance and transmittance functions such that they could be parameterized from real leaves: the Albedo map, the local thickness variation function, the specular intensity map, the specular roughness map.

1.1.5 Structural Colors

A chromatic display based on physical mechanisms, such as interference and diffraction of light, rather than on the more conventional chemical absorption is commonly referred to as *structural color*. The history of the scientific investigation of structural color in nature is long and it includes the illustrious scientific figures Hooke [Hoo65] and Newton [New30]. Continued and

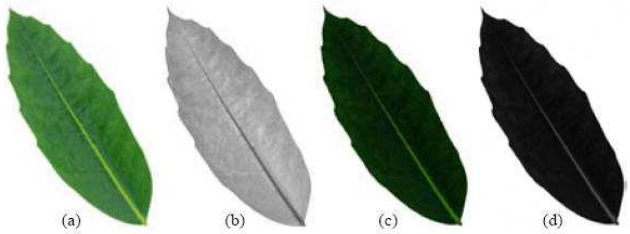
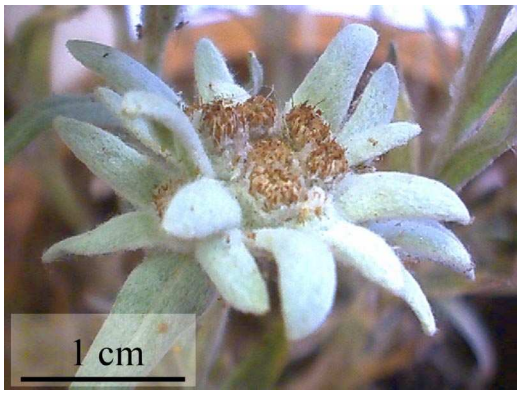
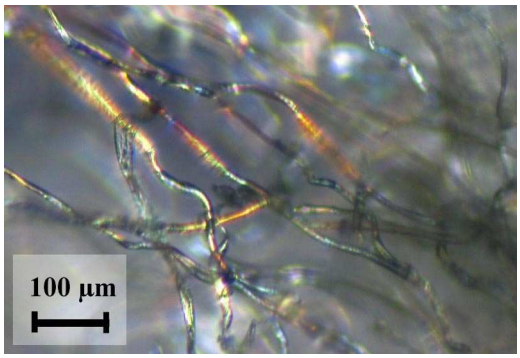


Figure 1.6: A set of BRDF and BTDF parameter maps for the front [WWD+05].

ever increasing interest, supported by the introduction of new investigation techniques, such as electron microscopy, has made it possible to discover countless examples of structural color in nature, amongst both plants and animals. Structural colors result from interference or diffraction effect, often from both. Interference produces colorful effects due the phase differences caused by a wave traveling through thin media of different indices of refraction. Unlike diffraction, interference can be directly modeled using the ray theory of light. Diffraction occurs when the surface detail is comparable to the wavelength of light. As shown in Figure 1.7, the white silver bracts of the edelweiss are due to the complex diffraction of filaments.



(a) Leontopodium nivale (edelweiss) are herbaceous plants which develop a characteristic star of silver-white foliaceous bracts particularly dense on the side facing the sky.



(b) Optical microscope image of the crisscross of transparent filaments which form the white cottony pads covering the edelweiss bracts.

Figure 1.7: Diffraction effects on an edelweiss [VRV+05].

1.1.6 Interference

Interference has previously been used in computer graphics. It is based on the ray geometry of light. We utilize Hirayama et. al's model [HKY+99] to render golden beetles in this thesis. This model works when material properties are given. We will also explore Sun's model [Sun06] which are able to simulate the iridescences on insects.

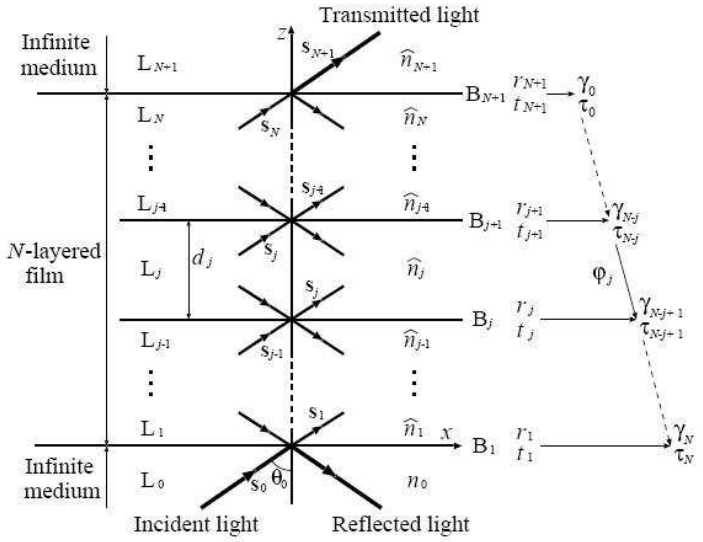


Figure 1.8: The system of multilayer films [HKY+99].

Rendering Interference on Multilayer Films

To visualize a light reflected on or passing through multilayer films consisting of dielectric, metallic, and/or semiconductive materials, it is necessary to calculate a composite reflectance and transmittance of the multilayer film system.

In [HKY+99], they consider the system of the multilayer films which has N-layers and the films are placed between two media (see Figure 1.8). The media infinitely extend toward the opposite direction of the interfaces of the films. We consider a system of multilayer films shown in Figure 1.8. Specifying a thickness and a refractive index for each layer, we can calculate reflectivity and transmissivity of each boundary plane based on wave optics [BW99]. Reflectivity and transmissivity are ratios of amplitudes of incident electromagnetic waves versus reflected and transmitted electromagnetic waves, respectively. Taking into consideration phase differences between both boundaries of layers, we composite the reflectivity and transmissivity in the order of the boundary planes from B_{N+1} to B_1. Finally, the composite reflectivity and transmissivity are converted into a reflectance and transmittance, which represent ratios of the energies of electromagnetic waves.

Assuming the thickness and complex refractive index of j-th layer are d_j and $\hat{n}_j$, respectively, the composite reflectivity and transmissivity between B_{N+1} and B_{j+1} boundaries are γ_{N-j} and τ_{N-j}, respectively. The composite reflectivity, γ_0, and transmissivity, τ_0, are those of the last boundary planes, B_{N+1}. That is, the seed equations at the B_{N+1} boundary plane are

$$\gamma_0 = r_{N+1} \tag{1.14}$$

$$\tau_0 = t_{N+1} \tag{1.15}$$

where r_{N+1} and t_{N+1} are the reflectivity and transmissivity at a single boundary B_{N+1}. After we compute the composite reflectivity, γ_{N-j}, and transmissivity, τ_{N-j}, the composite reflectivity

and transmissivity of the next boundary plane under the condition of multiple reflection can be calculated using the following recurrence equations [BW99].

$$\gamma_{N-j+1} = \frac{r_j + \gamma_{N-j} e^{2i\varphi_j}}{1 + r_j \gamma_{N-j} e^{2i\varphi_j}} \tag{1.16}$$

$$\tau_{N-j+1} = \frac{t_j \tau_{N-j} e^{i\varphi_j}}{1 + r_j \tau_{N-j} e^{i\varphi_j}} \tag{1.17}$$

where i denotes an imaginary number, r_j and t_j are the reflectivity and transmissivity at a single boundary B_j, respectively. φ_j is the phase difference between boundary planes B_j and B_{j+1}, and is expressed by the following equation.

$$\varphi_j = \frac{2\pi}{\lambda} \hat{n}_j d_j s_{z,j}, \tag{1.18}$$

where λ is the wavelength of light in a vacuum, d_j is the thickness of j-th layer, $s_{z,j}$ is a z-component of the unit vector, s_j, that indicates the direction of light in j-th layer, and $\hat{n}_j$ is the complex refractive index consisting of a refractive index, n_j, and an extinction coefficient, κ_j, $(\hat{n}_j = n_j + i\kappa_j)$. The unit direction vector, $s_j = (s_{x,j}, s_{y,j}, s_{z,j})$, is derived using Snell's law.

$$s_{x,j} = \frac{n_0 \sin\theta_0}{\hat{n}_j}, s_{y,j} = 0, s_{z,j} = \sqrt{1 - s_{x,j}^2}. \tag{1.19}$$

Note that the unit direction vector, s_j, of each layer is derived from the refractive index, n_0, the incident angle, θ_0, of the 0th layer, and the complex refractive index, $\hat{n}_j$, of each layer, as we consider a system of multilayer films whose boundary planes are parallel to each other. The reflectivity, r_j, and transmissivity, t_j, at boundary plane, B_j, are derived by Fresnel formulae, and are represented by a parallel component, $\parallel$, and a perpendicular component, $\perp$, of electromagnetic waves to the incident plane of the light.

$$r_{\parallel j} = \frac{\hat{n}_j s_{z,j-1} - \hat{n}_{j-1} s_{z,j}}{\hat{n}_j s_{z,j-1} + \hat{n}_{j-1} s_{z,j}} \tag{1.20}$$

$$r_{\perp j} = \frac{\hat{n}_{j-1} s_{z,j-1} - \hat{n}_j s_{z,j-1}}{\hat{n}_{j-1} s_{z,j-1} + \hat{n}_j s_{z,j}} \tag{1.21}$$

$$t_{\parallel j} = \frac{2\hat{n}_{j-1} s_{z,j-1}}{\hat{n}_j s_{z,j-1} + \hat{n}_{j-1} s_{z,j}} \tag{1.22}$$

$$t_{\perp j} = \frac{2\hat{n}_{j-1} s_{z,j-1}}{\hat{n}_{j-1} s_{z,j-1} + \hat{n}_j s_{z,j}} \tag{1.23}$$

Each component of composite reflectivities, $\gamma_{\parallel N}$ and $\gamma_{\perp N}$, and transmissivities, $\tau_{\parallel N}$ and $\tau_{\perp N}$, of the system of the N-layers film are calculated by repeatedly using Equation 1.16 and Equation 1.17. The reflectivities and transmissivities represent the ratio of amplitudes of reflected and transmitted electromagnetic waves, respectively. On the other hand, reflectance and transmittance represent the ratio of energies of reflected and transmitted electromagnetic waves, respectively. To convert the amplitude into energy, we take a square of the absolute value of the coefficients. A composite reflectance $\mathscr{R}$ and transmittance $\mathscr{T}$ of the system of the multilayer

Figure 1.9: Golden beetles rendered by our system [Don06], we coated the beetle with 50 nm gold film.

films are calculated by averaging the energies of the parallel and perpendicular components, because the contributions of these two components to the reflectance and transmittance are usually equal.

$$\mathscr{R} = \frac{|\gamma_{\parallel N}|^2 + |\gamma_{\perp N}|^2}{2} \tag{1.24}$$

$$\mathscr{T} = \begin{cases} \dfrac{\hat{n}_{N+1} s_{z,N+1}}{n_0 s_{z,0}} \dfrac{|\tau_{\parallel N}|^2 + |\tau_{\perp N}|^2}{2} \\ (\hat{n}_{N+1} \textit{ is a real number}) \\ 0, \quad (\hat{n}_{N+1} \textit{ is a complex number}) \end{cases} \tag{1.25}$$

Note that the transmittance becomes zero, when $\hat{n}_{N+1}$ is a complex number, because the $(N+1)$th layer has the capacity of light absorption. Consequently, the method for obtaining the composite reflectance and transmittance of the system of the multilayer films is as follows.

1. Given a complex refractive index, $\hat{n}_j$, and a thickness, d_j, of each layer, phase change, φ_j, reflectivities, $r_{\parallel j}$ and $r_{\perp j}$, and transmissivities, $t_{\parallel j}$ and $t_{\perp j}$, are calculated for an incident light intersecting with the multilayer films with angle, θ_0, by using Equation 1.18 and 1.20 through 1.23.

2. Composite reflectivities, $\gamma_{\parallel j}$ and $\gamma_{\perp j}$, and composite transmissivities, $\tau_{\parallel j}$ and $\tau_{\perp j}$, are calculated by repeatedly using Equation 1.16 and 1.17, respectively. Equation 1.14 and 1.15 are the seed equations for Equation 1.16 and Equation 1.17, respectively.

3. A composite reflectance, $\mathscr{R}$, and a composite transmittance, $\mathscr{T}$, of the system of the multilayer films are calculated using Equation 1.24 and 1.25, respectively.

In Figure 1.9, we show the golden beetles rendered by our system [Don06] with the above models.

Rendering Biological Iridescences

Brilliant iridescent colors occur on many biological objects. It is interesting to simulate biological iridescences in computer graphics. This will not only extend the rendering capabilities

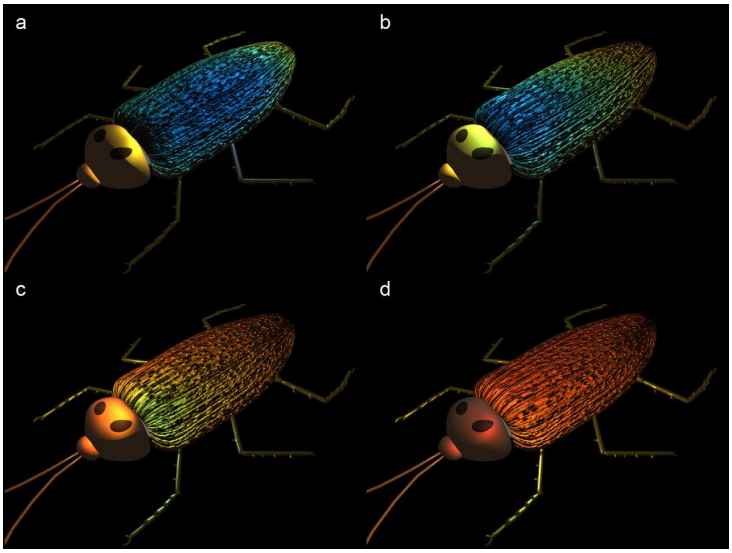


Figure 1.10: Ground beetles rendered with a fixed camera position but different positions of a white light source [Sun06].

in existing graphics applications such as computer game development, but also offer a new potential of using computer graphics to assist education in biology. Sun [Sun06] proposed spectrum-based models to simulate iridescent effects occurring on *Morpho* butterflies and beetles. Impressive results were generated with their system (Figure 1.10), but there were still some drawbacks. First, in [Sun06], the model is not able to simulate interference and volume diffraction together, while usually the iridescences are caused by both. On the other hand, the model is based on a large set of empirical parameters. Sometimes it is very difficult to adjust them to get a predictive result. In fact there is still a lot of work needs to be done in this area.

1.1.7 Rendering Complex Lighting Effects in a Natural Scene

In this thesis, we will present the rendering process of complex scenes with both the full spectra and RGB light and object interactions (Chapter 2). Unlike conventional systems that perform color calculations with tristimulus color values, our system allows to embed any kind of representation of the spectra to calculate colors of complex optical effects. Unlike spectral rendering systems previously proposed [DCWP02], our rendering process allows to use and compute spectra in complex scenes only when it is needed, i.e. when they have a significant impact in the transport process, and the final image.

We develop a new framework for realistic rendering. In this framework, we use RGB and spectrum together to represent the materials, light sources and the pixel colors of the synthesized image. We use a combination of spectrum and RGB model to represent the material, the light source and the rendered pixel color. For example, for a material with several elements, the elements which can cause complex lighting effects (interference, diffraction, etc.) is formulated by spectral models, while the other parts which are independent of wavelength are defined by RGB. We use this model as the intermediate format during the rendering process. The color of

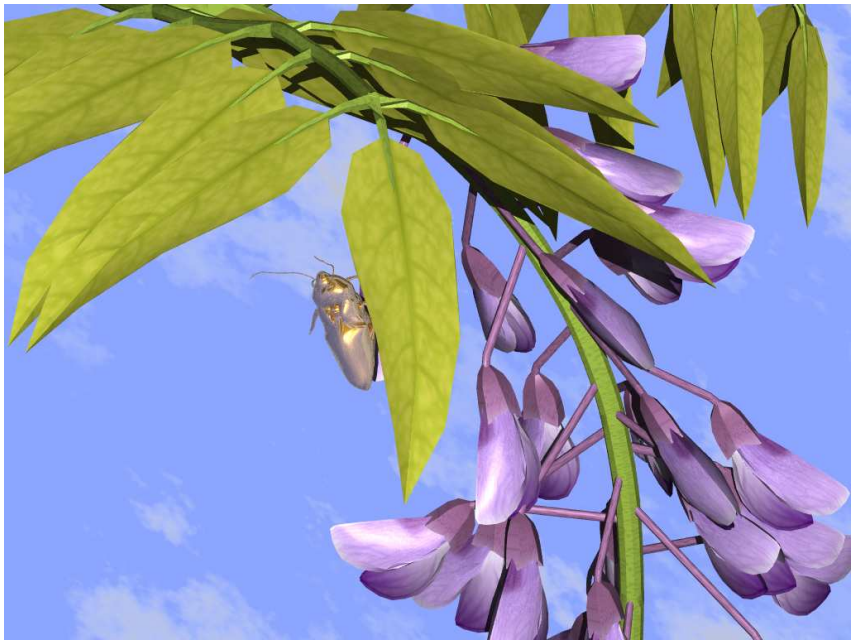


Figure 1.11: Golden beetle in a natural scene rendered with our system.

one point x can be written as follows

$$Color_{inter}(x) = Color_{RGB}(x) + Color_{spectrum}(x) \tag{1.26}$$

where $Color_{RGB}(x)$ is the RGB value generated by the interaction between the RGB based material elements and the light sources (represented with RGB model), and $Color_{spectrum}(x)$ is the spectral effect caused by the spectral elements of the material.

The basis of our framework is a ray tracer, while the other rendering algorithm like photon mapping could also been used. The whole rendering pipeline comprises three stages: preprocessing, rendering and color transformation. In the first stage, the SPD of the spectral light sources and the spectral functions of the spectral material elements are represented through loading data from the spectral database. The intensity of the RGB light sources and the RGB parts of the materials are also set by the user (or from texture). Then we pre-compute the RGB value of the spectral light sources and the spectra of the RGB light sources.We save these values together with the original data. The second stage is most important: here an intermediate image is generated based on local and global illumination models with ray tracing. The intermediate image is similar to a color image except that for every pixel the information is the combination of a spectrum and an RGB value instead of a color. In the rendering process, when calculating the reflectance intensity, if the reflectance of the material is RGB based, we need to convert the spectral part of the reflectance intensity gathered by the reflected ray into RGB. Contrarily, if the reflectance of the material is spectrum based, we need to convert the RGB part of the reflectance intensity gathered by the reflected ray into spectrum. The same for the transmittance calculation. Finally we transform the spectral part of the intermediate image into RGB [SFD99] and plus the previous RGB part. This will generate an RGB image for displaying on the screen.

Figure 1.11 shows two images generated by our system. In this scene, only the beetle contains

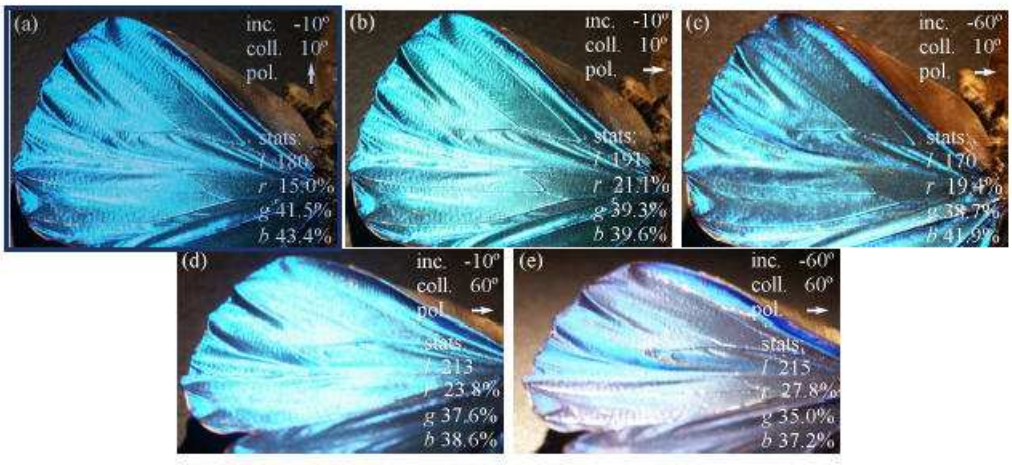


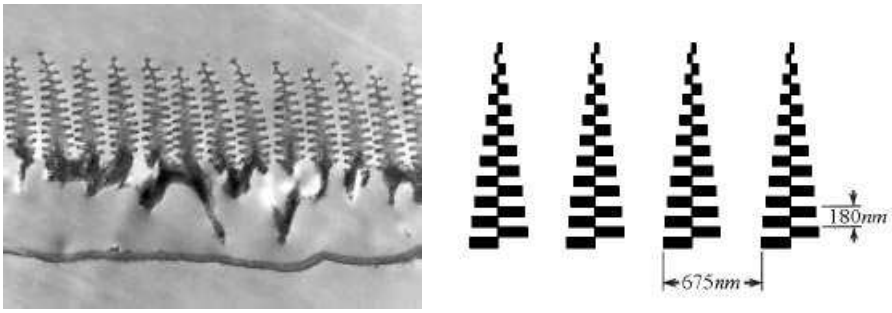
Figure 1.12: Photographs of a portion of the wing of *Morpho rhetenor* taken under different conditions of lighting and collection [Pla03].

spectrum-based material. Spectral calculation only occurs when the ray being traced intersected with the beetle. Compared with previous realistic rendering frameworks [GTS+97, SFDC01], our system uses full spectrum only when it is "necessary". It achieves both accuracy and efficiency during the rendering process.

1.1.8 Challenging Problem: Wave Theory of Light

We also plan to challenge the physically-based rendering by introducing a new class of analytical reflection models that simulate the effects of diffraction. Our goal was related to the microstructure found on the wings of the *Morpho Rhetenor* butterfly, which had been studied by both zoologists and photic engineers. The intricate arrangement of low-index dielectric material achieves, in the short wavelength regime on the visible spectrum, an extremely light and acts as a very efficient low-pass filter [Pla03]. The microstructure (i.e. the complex arrangements of ridges on a wing scale) achieves the extremely blue light and other specific optical effects of this butterfly. Since the Kirchoff Scalar Theory is not accurate for such complex microstructure, our main idea was to Fourier-transform the Maxwell equations and try to develop a new numerical method to simulate diffraction in a pine-tree microstructure. Then we hope that these theoretical investigation will account correctly for the experimental results in photonic engineering.

Photographs of a portion of the wing of *Morpho rhetenor* butterfly are shown in Figure 1.12. The six photographs were taken under different conditions of lighting and collection. As discussed in [KYK02], the mechanisms responsible for the structural color in the *Morpho* butterflies can be summarized as follows. (1) Lamellar structure in a ridge offers constructive interference, which results in the strong reflection within a selective wavelength range. (2) The irregularity in the ridge height eliminates the interference among the ridges, which results in the diffuse and broad reflection of a uniform color. Thus, the combined action of interference and diffraction due to the separate lamellar structure is essential for the structural color. (3) High reflectivity is realized owing to the presence of multiple layers in a lamella and a sufficiently



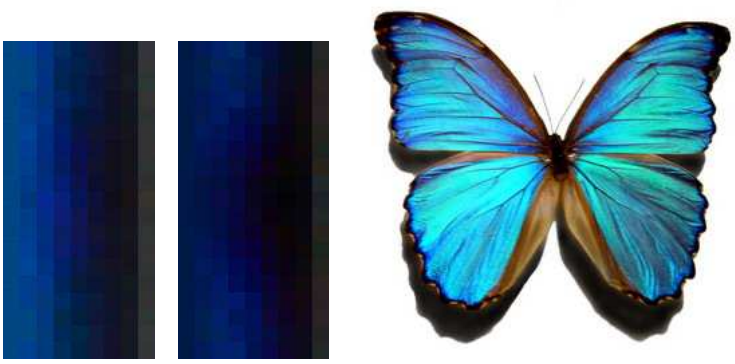
(a) (b)

Figure 1.13: Transmission electron microscope image showing the cross-section of a ground scale of the male butterfly *Morpho rhetenor* [VSLW99].

small separation between adjacent lamellae. (4) The irregularity in ridge height also results in the accidental interference of scattered light in space, manifesting as the glittering speckles. (5) The pigmentation in the scale absorbs the extraneous green to red light and enhances the blue coloring.

Morpho rhetenor butterflies' wings have a complicated structure composed of scales [Ghi91, VS00]. The cross-section of the ground scales (those producing color effects) exhibits a grating structure whose period is made of ridges with a tree-like geometry (Figure 1.13(a)). In the *Morpho rhetenor* butterfly scales, the lamellae (the "branches" of the tree-like structure) run near-parallel to the base of the scale [VS00]. So it is easier to model than some other species where the lamellae are tilted to the base of the scale [VS00, VSLW01], and this is the reason for which we focus on the *Morpho rhetenor* species. According to measurements reported in [VSLW99], the complex optical index of this tree-like structure will be approximated by the constant value $n = 1.56 + i0.06$ in the whole optical range. Note that the $n(\lambda)$ data need to be available. In photonic models, the numerical analysis is performed with the help of a rigorous method dealing with stacks of lamellar grating layers [BCM$^+$81, STP93, Li93]. It is an x-invariant structure in which each grating layer is y-periodic and can contain several rectangular regions with different optical indexes (in the present study, there are only two regions with indexes n and 1, see Figure 1.13(b)). The structure is illuminated by a plane wave with arbitrary wave vector. In each grating layer, the electromagnetic field is expanded on a modal basis, whereas in the superstrata and in the substrate (homogeneous media), the electromagnetic field is expanded on Rayleigh expansions (Fourier basis). On each interface between two gratings layers, or between a grating layer and the substrate or the superstrata, the continuity of the electric and magnetic field components is expressed by projecting the fields expansions on the Fourier basis. Finally, one gets the characteristics of the field reflected and transmitted by the grating. In the present case, we are mainly interested in the reflected field.

We have developed an experimental model based on [BCM$^+$81] with the irregular structure presented above. Results confirm that we could obtain variations of colors as a function of the incident light. But we are far away of the visual simulation. Clearly, at this step our research work was clearly unsuccessful. Moreover, even if we could provide the desire diffraction effects, we have handle with many other lighting effects (interference, subsurface scattering and



(a) (b) (c)

Figure 1.14: Iridescent color simulation of *Morpho rhetenor* butterfly. (a) and (b) are our simulating results. (c) is a photo of *Morpho rhetenor* butterfly

pigmentation), and variations in pigmentation that occurs at any point of the butterfly wings. Clearly the color pattern is also a crucial impact factor of visual appearance of a natural object.

1.2 Color Patterns

Color pattern plays a crucial role in the appearance of natural objects, such as fish, shells, especially flowers. How to realistically exhibit flower color patterns honestly in the virtual world created by computer graphics is an interesting problem. Besides geometry shape, color is the most important and attention-getting feature of flower. It is the result of color pigment distribution. Generate the spatial distribution of flower petal can be very labor consuming. We have developed a botany-based algorithm to generate pattern maps which represent the color of flower. As shown in Figure 1.15, our model is desired to be able to simulate various patterns of flowers.

1.2.1 Botanic Foundations

The modeling of plants and plant ecosystems has the captivating appeal of reproducing the visual beauty of nature while providing insights into the way nature works. This interplay between art and science is rooted in history and can be traced at least as far back as Leonardo da Vinci, whose notes about plant architecture 500 years ago remain valid today. The interdisciplinary character of plant modeling research is echoed by the diversity of existing and prospective applications of the models.

The complex architecture of plants consisting of many individual units, including branches, leaves, and flowers, is difficult to reproduce using traditional modeling techniques, which are better suited for artificial rather than natural objects; the difficulty increases when trying to model entire plant ecosystems. Simulation of plant development, based on botanical knowledge, offers a solution.

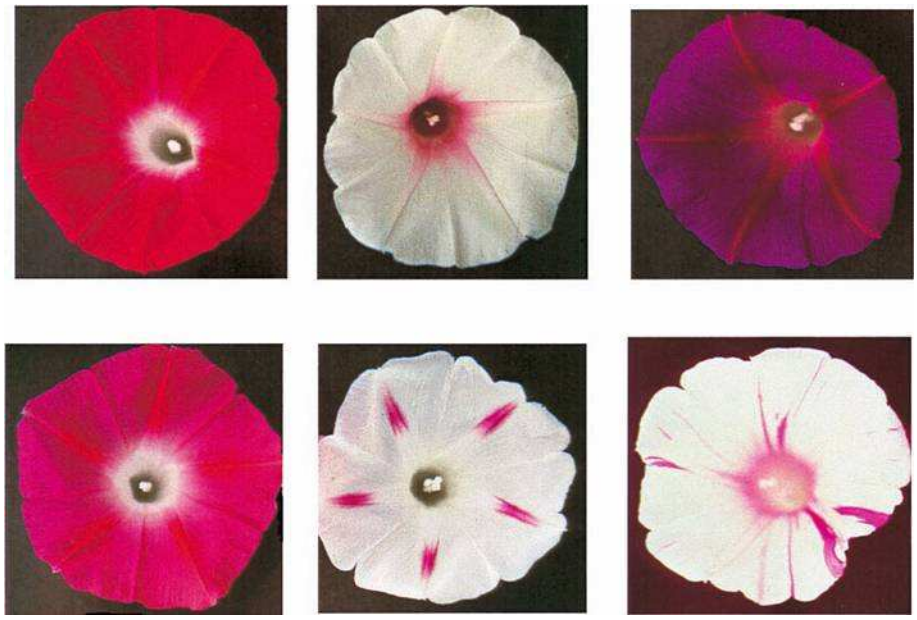


Figure 1.15: Flower color variations [CD00].

Simulation of natural phenomena is one of the most powerful methods for creating realistic models, animations, and rendered images in computer graphics. These simulations are often based in physics, possibly involving the mechanics of objects subject to forces in physically based modeling and animations or the optics of light propagation and distribution in radiative energy in rendering. Over the last 15 years, steady progress has also been made in the simulation of the biological processes governing plant development. This progress has led to biologically justified and visually realistic models of plants and plant ecosystems.

Applications are divided into two broad categories: image synthesis and scientific visualization [Pru00]. In image synthesis applications, the visual presentation of the models is the main objective; consequently, the modeling methods for image synthesis emphasize the ease of specifying and interactively manipulating the models [LD99, WP95]. In contrast, models developed for scientific and scientific-visualization purposes emphasize the biological accuracy of the simulated processes and structures [Pru98]. These models integrate our knowledge of plants and make it possible to examine hypotheses when the exact processes of plant growth and evolution are not known by botanists.

Color Pigment Distribution

The governing factors of petal color are very complicated, including gene, environment, horticultural measures, viral infection, etc. For a certain flower, its color pattern is mostly determined by gene, but environment on some degree can change the pattern. In this project, we do not consider the effect of environment. We only try to simulate how the flower realizes the pattern decided by gene with several pigments. According to botany researchers, "Flower pigments mainly consist of flavonoids, carotenoids and alkaloids. The chemical mechanism for

the expression of flower color is the ratio of different type of pigments in petals; the anatomic and optical mechanism for the expression of flower color is the dimensional distribution of the pigments in petals and their effects on light." The structure of petal is just like leaf, including upper epidermis, palisade cell, spongy cell, and lower epidermis. Some kinds of flower petals don't have obvious palisade cell. Most of the pigments are in the upper epidermis and palisade cell. The density of spongy cell can affect the brightness of petal. The thicker it is, the brighter petals are. If the upper epidermis has papillary protuberance, the petal will have a special silk-like shine. Therefore, we can decide the pigment distribution based on petal thickness map.

Some flowers only have one color in their petals. However, the color often changes during the bloom process. The color will have a smooth transformation. Some other flowers have more than one color in their petals. Those colors are combined together with certain regularity or stochastically. It is called variegation. We need to simulate both of the two kinds. For the smooth change of color, we can use interpolation method. For the sharp change of variegation, we must consider their rules.

Venation Pattern

Vein cells have few color pigments, but they can cause accumulation of color pigments in the nearing cells. So in the petals, we can often find color difference along the veins.

Flower petal and sepal also have the leaf-like structure, but their vein system is not as regular as leaf. The flower vein system is degenerated, and lack collenchyma, so its venation pattern is sparser, and has fewer branches. We don't know if the growth power in flower petal is the same as in leaf. At the present stage, we suppose they are similar kinds of auxin. In flower venation pattern, there is one kind of strong veins called trace. The number of trace in one petal is different by species. It can be from 0 to 6 or more. The trace need to be preset.

1.2.2 Texture-Based Pattern Simulation

As previously discussed, the natural looking of the images obtained with sophisticated light and matter interaction models are due to the texture used for color variations and venation. Wang et al. [WWD+05] developed a framework for rendering of plant leaves in real-time with global illumination. The spatially-variant BRDFs and BTDFs employed to describe leaf appearance are stored in a set of parameter maps. This method is efficient and produces very realistic results (Figure 1.16(a)). However, it is tightly limited by measured data and will be inconvenient when plants with many patterns are required in one scene. Runions et al. [RFL+05] introduce a class of biologically-motivated algorithms for generating leaf venation patterns. These algorithms simulate the interplay between three processes: (1) development of veins towards hormone (auxin) sources embedded in the leaf blade; (2) modification of the hormone source distribution by the proximity of veins; and (3) modification of both the vein pattern and source distribution by leaf growth. These processes are formulated in terms of iterative geometric operations on sets of points that represent vein nodes and auxin sources. Applications of their algorithms

(a) Image-based rendering of plant leaves. (b) Simulate leaves with venation textures.

Figure 1.16: Simulate the leaf color and venation patterns with texture [WWD+05, RFL+05].

include texture and detailed structure generation for image synthesis purposes, and modeling of morphogenetic processes in support of biological research.

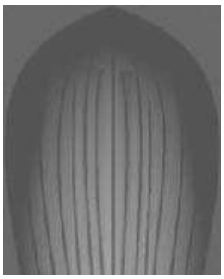
1.2.3 Modeling and Visualization of Flower Color Patterns

In this thesis (Chapter 3), we introduce an efficient system for modeling and visualization of flower color patterns. Our goal is to build a connection between the user-controllable image synthesis and real botanical patterns, that is to provide a flexible control allowing users to model a wide variety of biologically plausible flower color patterns.

The input for our system consists of: (1) the initial state (shape and venation information of the flower, and the initial pigment distribution), (2) parameters characterizing the spread and interplay of pigments, and (3) functions and parameters defining flower color expression. The shape is specified by the user as a black-white image that defines the petal's range. The venation pattern is given as a set of vein nodes, including a vein root and vein width. According to the canalization hypothesis [BMH00], the vein is composed of cells with higher capability in auxin transportation.

Normalized pigment distribution is represented by a grey-scale map called pigment distribution map. Each kind of pigment corresponds to one map. Its concentration at each locus is represented by the grey level of the pixel. The darker the pixel is, the higher the pigment concentration is. The initial pigment distribution is given by users as several grey-scale images, which follow the definition of pigment distribution map.

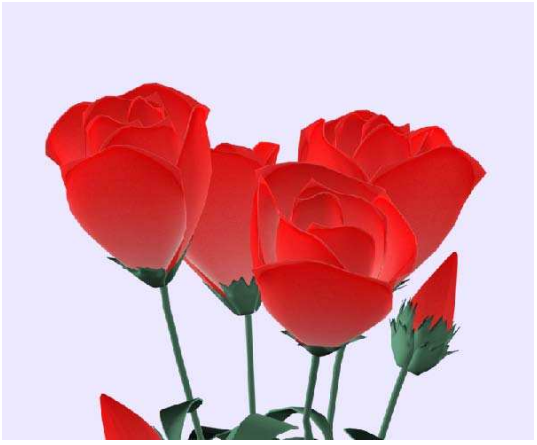
We use reaction-diffusion equations [Tur91] to describe the interaction among different pigments. Based on the assumption that the pigment composition of one cell can be calculated from its neighbors, we can simulate the pigmentation in the growing process if an initial state and transportation kinetics are given. The initial state can be freely assigned by the user. For



(a) Pigment distribution map.

(b) Flower color expression.

(c) Rendering with pattern textures.



(d) Rendering with simple color gradient textures.

(e) Original flower model.

Figure 1.17: Flower color pattern simulation.

convenience, we use an uniform distribution map as the initial state in our experiments. Detailed discussion of pigment distribution map generation with reaction-diffusion is in Chapter 3.

To paint the greyscale pigment distribution map, we first construct a pigment database to store the absorb color data of all pigments of different concentrations, and some other information which is not related with the pigment distribution map, such as the lighting environment. After these pre-computation is finished, we calculate the color pattern of the petal based on the pigment distribution map in runtime.

One of our results in flower color pattern simulation is shown in Figure 1.17. We use the pattern map (Figure 1.17(b)) as the texture of the 3D model in the rendering process. Compared with the results which are rendered with the color map generated by simple gradient function (Figure 1.17(d)) or naive color (Figure 1.17(e)), our result is more natural and realistic in expressing the real appearance of the flowers.

1.2.4 Realistic Simulation of Seasonal Variant Maples

Visualization of organic objects and natural phenomena is still very challenging. Especially if their complexity raise problems concerning their seasonal changes. A large group of applications require the proper presentation of trees of different genus. Additionally they should be visualized and influenced by their environment. In this thesis (Chapter 4), we present an algorithm for the visualization of trees during the passing of the year, taking into account the changing of the tree's leaves color [ZDM06].

Early methods of displaying trees such as fractal methods and billboards could not satisfy the need for good quality visualization including environmental influences. These methods especially lacked seasonal changes. This of course was due to the fact that the complexity of trees used up the existing performance on the hardware available at these times. Furthermore not much was known on seasonal changes of trees and how they are influenced by their environment. With ongoing progress in biology and chemistry the foundations are given to realize a realistic looking seasonal change of a tree, without the need to only rely on experimental and observation data.

To create the effect of changing color of the tree's leaves a decision was made to come up with a more biochemical approach. In detail we first identified the significant elements responsible for the leaf color. The color of a leaf is mainly caused by the pigments it contains, more precisely the light absorption behavior of these pigments. The influence of the chlorophyll and carotene pigments is well known today.

But there are also other pigments that affect the leaves' color especially in autumn. To restrict the complexity of the calculations for the color of a leaf, we decided to concentrate on the most important pigments, these are the previously mentioned chlorophyll, carotene, anthocyanin, and tannin. Chlorophyll is responsible for the greenish color, while carotene creates a yellow-orange color. The anthocyanin pigments mainly cause a red appearance while finally the tannin is the base pigment existent all over the year and it is responsible for the brown base color.

Our system simulates the leaf color seasonal change of maples emphasizing on tree scale. The whole system consists of three steps: (1) environment configuration, (2) climate influence simulation and (3) leaf texture acquisition. Step (1) decides the general color change timing of the maple tree based on its local environment. Step (2) makes further adjustments to the timing determined by step (1) considering the influence of climate in the specific case, and computes the pigment concentration for every leaf. Step (3) generates the leaf textures with the pigment concentration data. Details and results of algorithm are addressed in Chapter 4.

1.3 Large Scale Ecosystems

As previously discussed, we could render a tree with various leaves or flowers with subtle color variation. However, how to deal with large environment is still a problem. Modeling and rendering of natural scenes with thousands of plants poses a number of problems. For example, to generate a virtual scene which is similar as Figure 1.18, the terrain must be modeled and plants

Figure 1.18: A large scale ecosystem containing a lot of sunflowers.

must be distributed throughout it in a realistic manner, reflecting the interactions of plants with each other and with their environment. Geometric models of individual plants, consistent with their positions within the ecosystem, must be synthesized to populate the scene. The scene, which may consist of billions of primitives, must be rendered efficiently while incorporating the subtleties of lighting in a natural environment. Deussen et al. [DHL+98] developed a system built around a pipeline of tools that address these tasks. The terrain is designed using an interactive graphical editor. Plant distribution is determined by hand (as one would do when designing a garden), by ecosystem simulation, or by a combination of both techniques. Given parametrized procedural models of individual plants, the geometric complexity of the scene is reduced by approximate instancing, in which similar plants, groups of plants, or plant organs are replaced by instances of representative objects before the scene is rendered. Impressive results were generated by their system, while the modeling and rendering process are still very complex. On the other hand, plenty of biologic data and 3D models are required to generate an abandon scene.

In this thesis, we have developed texture-based methods to generate large scale ecosystems. Our work is concentrated on real-time synthesis of natural textures (Chapter 5) and the protection of texture global appearances in the results (Chapter 6). Our algorithms work very well on natural textures while also could generate impressive results for other kinds of textures.

1.3.1 Texture and Texture Synthesis

In this section, we describe the goal of texture synthesis. We begin with a brief discussion of the definition of textures.

What is a Texture?

Reproducing detailed surface appearance is important to achieve visual realism in computer rendered images. One way to model surface details is to use polygons or other geometric primitives. However, as details becomes finer and more complicated, explicit modeling with geometric primitives becomes less practical. An alternative is to map an image, either synthetic or digitized, onto the object surface, a technique called texture mapping. The mapped image, usually rectangular, is called a texture map or texture. A texture can be used to modulate various surface properties, including color, reflection, transparency, or displacements. In computer graphics the content of a texture can be very general; in mapping a color texture, for example, the texture can be an image containing arbitrary drawings or patterns.

Unfortunately, the meaning of texture in graphics is somehow abused from its usual meaning. The Webster's dictionary defines texture as follows:

Texture, noun [1578]

1. something composed of closely interwoven elements; specifically a woven cloth
2. the structure formed by the threads of a fabric ...

In other words, textures are usually referred to as visual or tactile surfaces composed of repeating patterns, such as a fabric. This definition of texture is more restricted than the notion of texture in graphics. However, since a majority of natural surfaces consist of repeating elements, this narrower definition of texture is still powerful enough to describe many surface properties. This definition of texture is also widely adopted in computer vision and image processing communities.

In this thesis, we concentrate on the narrower definition of textures, i.e. images containing repeating patterns. Since natural textures may contain interesting variations or imperfections, we also allow a certain amount of randomness over the repeating patterns. For example, a honeycomb texture is composed of hexagonal cells with slight variations of size and shape of each cell. The amount of randomness can vary for different textures, from stochastic (a sand beach) to purely deterministic (a tiled floor). This definition of textures allows us to model textures under a unified framework. We also attempt to generalize the notion of textures beyond images to incorporate other physical phenomena such as animations and articulated motions.

What is Texture Synthesis?

Computer graphics applications often use textures to render synthetic images. These textures can be obtained from a variety of sources such as hand-drawn pictures or scanned photographs. Hand-drawn pictures can be aesthetically pleasing, but it is hard to make them photo-realistic. Most scanned images, however, are of inadequate size and can lead to visible seams or repetition if they are directly used for texture mapping.

Texture synthesis is an alternative way to create textures. Because synthetic textures can be made any size, visual repetition is avoided. Texture synthesis can also produce tileable images

(a) Texture (b) Synthesis result

Figure 1.19: Problem Formulation. Given a sample texture (a), our goal is to synthesize a new texture that looks like the input (b). The synthesized texture is tileable can be of arbitrary size specified by the user.

by properly handling the boundary conditions.

The goal of texture synthesis can be stated as follows: Given a texture sample, synthesize a new texture that, when perceived by a human observer, appears to be generated by the same underlying process (Figure 1.19). The major challenges are:

Modeling How to estimate the texture generation process from a given finite texture sample. The estimated process should be able to model both the structural and stochastic parts of the input texture. The success of modeling is determined by the visual fidelity of the synthesized textures with respect to the given samples.

Sampling How to develop an efficient sampling procedure to produce new textures from a given model. The efficiency of the sampling procedure will directly determine the computational cost of texture generation.

1.3.2 ω-Tile

We choose ω-tile [NWT$^+$05] as the basis as our real-time texture synthesis system. The ω-tile construction approach starts with randomly obtaining a set $\mathscr{F}$ of four small square patches from the input texture $\mathscr{S}$. With these, it forms each time a square block to construct eventually an ω-tile. Each block is a non-overlapping arrangement of four (not necessarily distinct) patches of $\mathscr{F}$. Figure 1.20 shows an example of four such blocks A, B, C and D (obtained from different arrangements of the four patches in $\mathscr{F}$) and the intermediate tiles A_i, B_i, C_i, and D_i cut from the center of A, B, C and D. The seams in each intermediate tile are removed by replacing the interior of the tile with other pattern from $\mathscr{S}$ to generate an ω-tile.

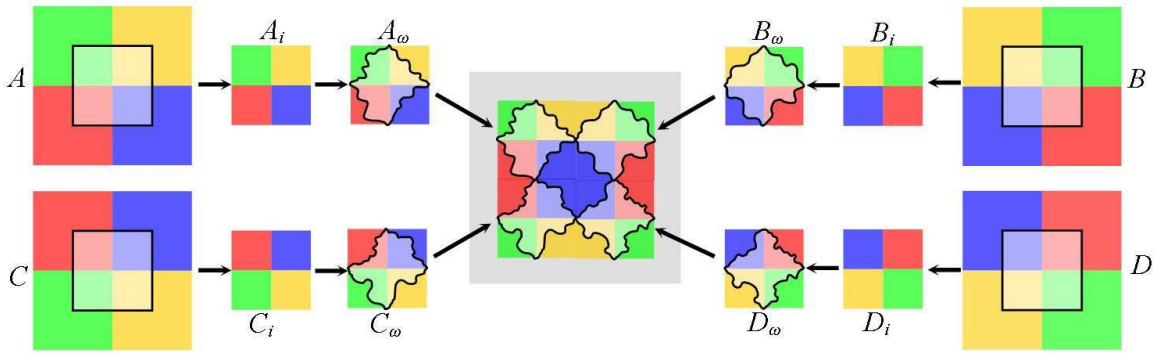


Figure 1.20: Four intermediate tiles A_i , B_i, C_i and D_i cut from blocks A, B, C and D, respectively, are used to generate four ω-tiles A_ω, B_ω, C_ω and D_ω, which, in turn, tile the 2-by-2 area in the middle [NWT$^+$05].

1.3.3 Optimized Tile-Based Texture Synthesis

In this thesis(Chapter 5), we present an approach for tile-based texture synthesis that is based on the optimization of tile set quality within a *Genetic Algorithm* (GA)-based framework. Our main contribution is to merge some locally defined optimization measures into a global *evaluation function* that can jointly optimize the quality of the entire tile set. This evaluation function balances the qualities among tiles and can be optimized by GA with reasonable computational cost. GA is an efficient and near global optimum search method [SP94, Koz95, LL02, PMJ96], which can automatically achieve and accumulate knowledge about the search space and adaptively control the search process to approach the global optimal solution.

Our approach starts with an 8-tiles ω-tile set used in [NWT$^+$05] as the basic tile set. An extendable rule will be given to directly derive new tile sets from existing ones. By employing more sample patches during the tile construction process, the new ω-tile sets introduced in this thesis can generate patterns more abundant than the sets of similar sizes in both [NWT$^+$05] and [CSHD03].

As shown in Figure 1.20, an ω-tile is constructed by finding a matching patch from the input, and merging it into the intermediate tile using graph-cut. The *intermediate tile* is the combination of four sample patch quarters. We propose a global optimization algorithm to search for a feasible set of sample patches. Intermediate tiles formed by these patches will satisfy both local and global optimal conditions. The local condition implies that each intermediate tile could find an adequately well *matching patch* (the texture patch picked from the input example to merge into the intermediate tile for erasing junctions) from the input, while the global condition means that the tile qualities are balanced according to their smallest matching errors. The two conditions are interpreted together as an *evaluation function*. This function is defined as the linear combination of the sum of the *matching errors* (the distance between the intermediate tile and the candidate matching patch) between intermediate tiles and their *closest matching patches* (the candidate matching patch with the smallest distance), and the standard variance of all the errors. Sample patches selection proceeds by optimizing this evaluation function using GA. We start searching with a group of sample patches extractions. The optimization procedure

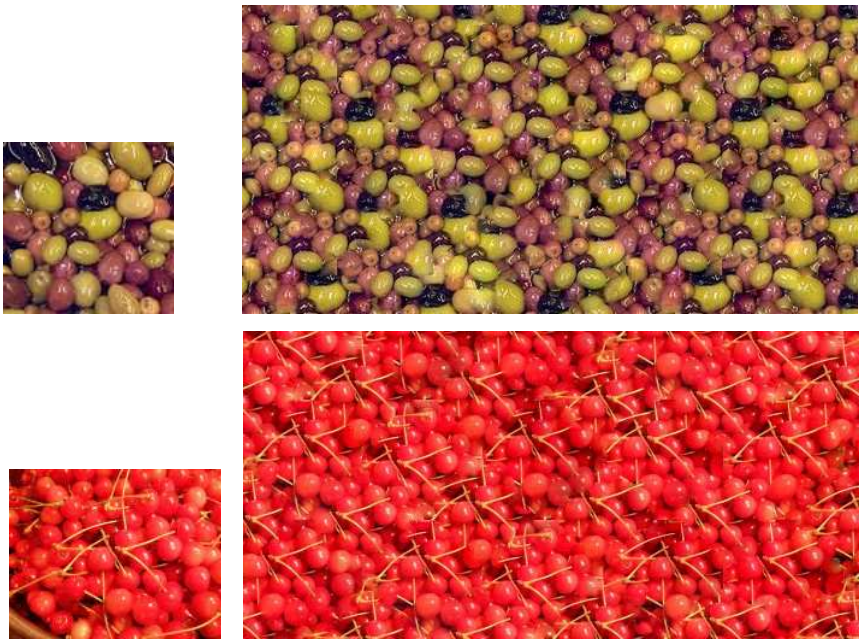


Figure 1.21: Results of our optimized tile-based texture synthesis algorithm. The sample textures are on the left and the synthesis results are on the right. Both results are generated in real-time.

improves the quality of the entire group through successive iterations of the algorithm. The final solution is the best extraction in the group when GA terminates.

The whole working flow of our optimized tile-based texture synthesis algorithm is: in precomputation, first randomly initialize a considerable number of sample patches sets (as the chromosomes) from the input example, then use GA to find the best one. Finally graph-cut is employed for junction elimination. The run-time tiling process is the same as [NWT+05], we can synthesize arbitrary size of texture images in real-time. Two results generated by our algorithm are shown in Figure 1.21, we can see that the optimized tile-based texture synthesis method works well on natural textures.

1.3.4 Feature-Aware Texture Synthesis

Texture synthesis is defined in [Ash01] as "a texture synthesis method starts from a sample image and attempts to produce a texture with a visual appearance similar to that sample."

The visual appearance of an example can be analysed by the *local* continuity of the texels(TEXture ELement), and *global* perceptual features such as texel scale variation. These features are due to environmental changes, e.g. perspective view-point, luminance, object distributions and geometry of the underlying surface.

Traditional approaches analyze local properties of a given sample and create visually similar images by comparing local neighborhoods. Therefore, they could nicely preserve the *local* continuities of the texels. Nevertheless, these approaches are not sensitive to the global perceptual

Figure 1.22: Flowers texture with golden beetles.

features and hence limited to isotropic samples.

In this thesis (Chapter 6), we provide a framework for perceptual-feature-aware texture synthesis. The underlying idea is to consider the perceptual-featured textures (PFT) as the consequences of environmental changes to a general isotropic texture.

We presents several interactive techniques to define and extract these features from isotropy using a multi-modal (scale, type, color), multi-dimensional mapping, which we called a *feature field*. Each feature is quantized into a *feature map* masked by color values. Multiple maps could be integrated together as the constraint for output synthesis. Compared with the deformation field used by Liu et al. [LLH04] for manipulating near-regular textures, we extend their model in three aspects: (1) **Multi-modal feature fields**: extending the concept of texture deformation to include scale and type as well as lighting and geometry so that we can preserve the global appearance of PFT in output images; (2) **Feature field extraction**: to capture and adjust the corresponding features from the input sample with simple user-assisted interactions, we introduce techniques for more intuitive control; (3) **Feature map manipulation**: actively controlling the feature distributions of the output feature maps so that the appearance of the output texture can vary according to users' specifications.

1.4 Conclusions

In this chapter, we first introduce some background knowledge required to conveniently understand the thesis. Then we explore the main contributions of our work, including the efficient rendering framework for complex lighting effects, simulating algorithms of flower color patterns and seasonal leaves, optimized tile-based texture synthesis and feature-aware texture synthesis. As shown in Figure 1.22, the background flowers are generated by our optimized tile-based texture synthesis algorithm, and the golden beetles are generated by our rendering framework. One can see that with our algorithms, impressive results could be generated, even if we still can not achieve the same appearance of Figure 2.

Chapter 2

Rendering Optical Effects Based on Spectra Representation in Complex Scenes

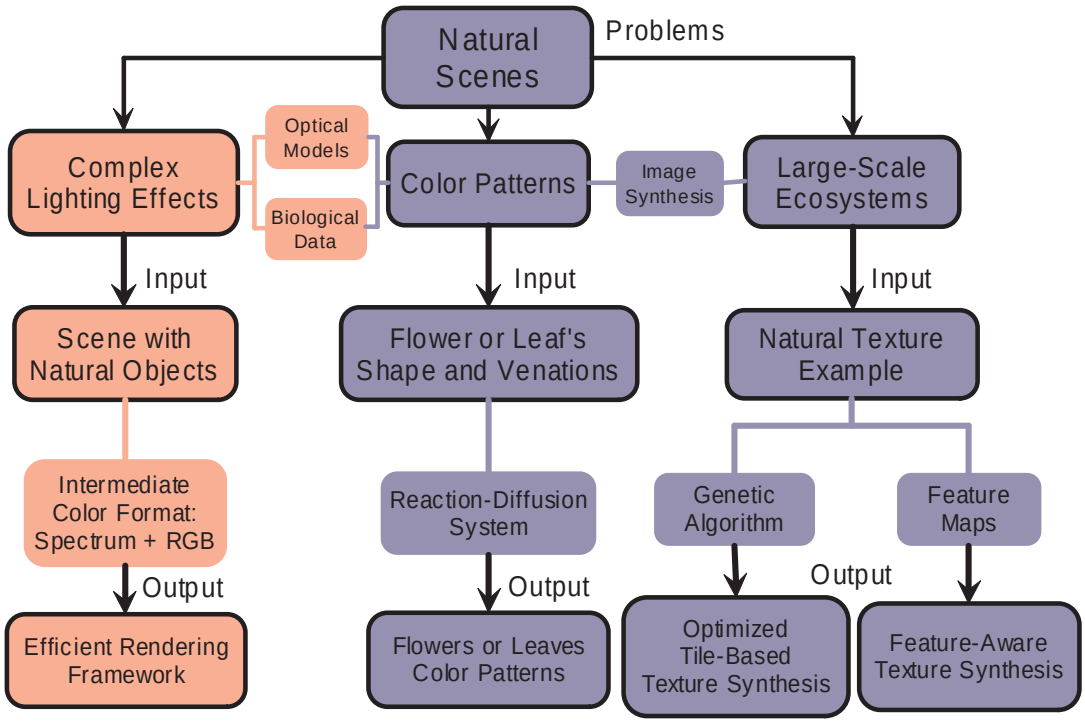


Figure 2.1: Rendering optical effects in complex scenes.

2.1 Introduction

Many phenomena and materials, in nature or in industry, have complex optical effects, namely interference, diffraction, fluorescence, dispersion, phosphorescence, etc. These physical effects cannot be rendered with current RGB-based graphics platforms and software. However, as pointed in [DCWP02] and [Gla95], color computations in a renderer have to be performed in spectral space if the output is to be used for predictive purpose.

Figure 2.2: The *Edgar Poe Gold Bug* is rendered with a full spectral function representation while the main part of the 3D scene is computed with RGB models. With our framework, computing a scene with an object producing wave optics effects costs only a few additional seconds.

To simulate these optical effects, some researches have been focused on the full spectra representation of light and objects. An exhaustive state of the art of these works was published [DCWP02]. These methods are not tractable when a complex scene has to be rendered. In [Sun06], for example, a multilayer films model is coded with their efficient spectra representation, previously published in [SFDC01] and implemented in a popular RGB-based renderer. However, the optical effects of the object presented (insect) are computed independently of its 3D environment, although they are highly depending on the lighting conditions of this environment and interact with it. Our rendering process overcomes this problem.

Our idea is based on the observation that usually, only a part of a scene needs to be simulated with a full spectra representation. In Figure 2.2, only a part of the insect has a structural color. In Figure 2.4, the insects are even partly hidden. In the two figures, color of leaves and flowers, as well as some parts of the insect itself, are due to pigmentation. Moreover, the light transport needs to be simulated with full spectra representation, only when the optical effects of the material have a visual impact on other objects.

This chapter proposes a framework that combines the traditional rendering system with spectra representation. Light sources and surfaces can be described and loaded in term of spectra functions or in term of RGB representation. Note that RGB values can be transformed in spectra if

it is needed. Then an intermediate image is generated based on local and global computations (here with the ray tracer). This image is similar as to a color image except that for each pixel the information is a combination of a spectrum and an RGB value instead of a color. Finally, we project the spectral part of the intermediate image into a CIE XYZ color, before transforming these XYZ values to RGB values. To prove the efficiency of our method, we implemented the multilayered film model proposed in [HKY+99]. Optical phenomena of multilayer films are caused by reflection, refraction interference and absorption of light inside each layer of multiple films. A result of these phenomena is often called iridescence because the color of the material could change greatly, depending of the viewing direction and wavelength of light. However, any other model could be implemented. In our rendering process, the spectra functions representation is coded with the dual representation of spectral functions proposed in [SFDC01, Sun06]. The method decomposes all spectra into a smooth part represented by Fourier coefficients and a list of spikes defined by their location and height. Note that in [SFDC01], the renderer proposed by the authors emphasizes real spectra as input, computes full spectra light-object interactions (with the previous published dual method) and generate spectral images convertible to RGB format. We use this efficient dual method to computing our 3D scenes in our tests for comparison.

Results show that with our framework, computing a scene with an object producing wave-based optics effects costs only a few additional seconds and are very efficient compared to previous method for the same quality image. Note, moreover, that in the different tests, our framework is embedded in a simple ray tracer, but it could be easily embedded in any RGB-based renderer, including Photon Mapping [Jen01] and allow similar performances.

In the following sections, we first discuss some related works. Then we describe the color representation models used in our system, and also give the whole work flow of the render. We show some images generated by our system and address the advantages comparing with the traditional methods in the fifth section. Finally we draw the conclusion and discuss about the future work.

2.2 Previous Work

This chapter presents the rendering process of complex scenes with both the full spectra and RGB light and object interactions. Unlike conventional systems that perform color calculations with tristimulus color values, the system allows to embed any kind of representation of the spectra to calculate colors of complex optical effects, namely metallic films as well as biological iridescence effects due to interference and diffraction. Unlike spectral rendering systems previously proposed (see [DCWP02] for details), our rendering process allows to use and compute spectra in complex scenes only when it is needed, i.e. when they have a significant impact in the transport process, and the final image.

A lot of models describe spectacular optical effects that can be rendered with spectra functions representation. Hall and Greenberg [HG83] developed a full spectra renderer that performs color components using any number of arbitrary special samples as well as RGB and CIE color

models. In [Mey88], Meyer proposed a four fixed wavelength approach. To render photorealistic rendering of natural phenomena, Researchers of Utah University [JRW97, PSS99, PA00]) used spectral rendering that was at least partially similar to the sampling approach. In [Pee93], Peercy presented an alternative to the direct spectral technique. He proposed a representative based on a set of chosen basis functions. In [IP00] and in [RP97], Iehl and Rougeron proposed adaptive rendering systems. Stam [Sta99] developed a reflection models for metallic surfaces that handle the effects of diffraction. In [SFCD99, SFDC01] and [Sun06], Sun proposed a method that overcomes the limitation of the fixed sampling technique and the linear color representation proposed in [Pee93]. We have implemented this method in our rendering process.

Many works have been focused on developing physically based lighting models and perceptually based rendering procedures for computer graphics that will produce synthetic images that are visually and measurably indistinguishable from real-world images [DCWP02]. Greenberg et al. [GTS+97] proposed a framework which subdivided the whole rendering process into three sub-sections: the local light reflection model, the energy transport simulation, and the visual display algorithms. Glassner provided a mathematical framework for phosphorescence and fluorescence [Gla94, Gla95]. They summarized the transport equation that describes this dynamic equilibrium in a scene of surfaces within a participating medium. To accurately simulate the optical phenomena, Sun et al. [SFDC01] proposed a rendering framework which emphasized real spectra as input, retains full spectral light-object interactions, and generates spectral light-object interactions, and generates spectral images (convertible to RGB images for display). This framework is capable of handling wavelength-related optical effects including dispersion, interference, diffraction, and fluorescence, but still very costly, especially when objects with complex spectrum based materials are present only in a part of the scene. Another problem is to render the scene users need to set all the object materials with spectrum, usually it is difficult to get all the spectral data of the materials in the scene.

In all these systems above, full spectra are highly time consuming when the scene to be rendered is composed of a large number of polygons and light interactions. In our approach, since all the other computations use RGB values, the rendering process is very fast. Furthermore, the process is all the more so efficient since the RGB component of the color calculations in the 3D scene is great compare to the spectra component.

An accurate and efficient spectral representation is required in our framework. Many methods have been proposed like sampling [CT82, Mey88], linear model representation [Pee93], and using polynomials [RF91]. Unfortunately these methods all have difficulties in balancing the accuracy and efficiency. To overcome the drawbacks of the previous methods, Sun et al. proposed a composite spectral model by decomposing all spectra into a smooth background and a list of spikes [SFDC01]. They represented the smooth part with Fourier coefficients and a spike is specified by its location and height. Let all spectra be generally denoted by $S(\lambda)$, then the decomposition is

$$S(\lambda) = S_{smooth}(\lambda) + S_{spike}(\lambda).$$

By decomposing any spectrum into a smooth background and a collection of spikes, users can represent all spectra through a small number of parameters while maintaining good accuracy. We also use this spectral representation model in our framework.

We use Hirayama et al.'s work [HKY+99] for rendering objects coated with multilayer thin films to test our framework. Their method is based on wave optics, and is able to accurately visualize the optical effects of multilayer films, taking into consideration such factors as multiple reflection, interference and absorption of light inside the films. In the preprocess of rendering, composite reflectance $\mathscr{R}$ and transmittance $\mathscr{T}$ of the system of multilayer films are calculated using

$$\mathscr{R} = \frac{|\gamma_{\parallel N}|^2 + |\gamma_{\perp N}|^2}{2} \tag{2.1}$$

$$\mathscr{T} = \begin{cases} \dfrac{\hat{n}_{N+1} s_{z,N+1}}{n_0 s_{z,0}} \dfrac{|\tau_{\parallel N}|^2 + |\tau_{\perp N}|^2}{2} \\ \quad (\hat{n}_{N+1} \textit{ is a real number}) \\ 0, \quad (\hat{n}_{N+1} \textit{ is a complex number}) \end{cases} \tag{2.2}$$

for each sampled incident angle and wavelength of light, information is stored in tables. Then in their raytracing process, for all wavelengths of light, only a single ray is used to trace reflected or transmitted light. The intensities of the reflected and transmitted rays are multiplied by the composite reflectance and transmittance (interpolated from the pre-calculated tables), respectively.

In [Sun06], a multilayered films is also proposed. The model has several new nice properties. However, the complex optical effects of the *Morpho Rethenor* that is chosen as an illustration cannot be rendered with an approximated interference scheme. In fact, the irregularity in the ridge height of the *Morpho Rethenor* eliminates the interference among the ridges, which results in the diffuse and broad reflection of a uniform color [KYK02]. So due to its complex microstructure, the structural color combines both diffraction and interference in a complex microstructure.

2.3 Efficient Renderer for Rendering Natural Scenes

We develop a new framework for realistic rendering. In this framework, we use RGB and spectrum together to represent the materials, light sources and the pixel colors of the synthesized image.

2.3.1 Materials and Light Sources

First, to simulate the natural phenomena which can not be accurately calculated with simple RGB models, and protect the efficiency if the objects also have some RGB oriented properties at the same time, we allow RGB and spectrum to work together to represent one material, different element of the material can have different type of representative model. This means that we set the "natural" part of the material with spectral model and let the other parts still be represented by RGB. For example, considering an object with both interference effect and simple diffuse appearance, we integrate the interference part with the related physical based model (like a

spectrum based BRDF model) and define the diffuse value which is independent of wavelength with RGB. Another example is if a transparent medium is dispersive, its refractive index is a spectral function $R(\lambda)$ (where λ is the wavelength value), and the reflective element can be definitely be represented by a simple RGB value.

A real view of the world is determined by the behaviors of light. To synthesize realistic images of natural phenomena, we use light sources described with spectral power distributions (SPDs) in the scenes which have spectral effects. Like Sun's spectrally based framework [SFDC01], we also recommend the using of real spectral data in order to ensure the accuracy of spectral effect simulation. Such data can be collected from experimental measurements [Gla95] or reliable numerical calculations.

At the same time, we also allow RGB light sources in the same scene, this feature is to facilitate the use of monochromatic light sources and the approximation of the real light sources when there is no getatable spectral data. The user can simply choose the RGB color of the light source and convert it into spectrum in the rendering process. In our system, we use Sun's method to derive spectra from colors [SFCD99].

2.3.2 Color Representation

Based on Huygens' principle of independent propagating of light [BC50], we separately calculate the effect of the light sources, no matter it is an RGB or spectral light source. Then we integrate the effect together as the final results of the local illumination. In our system, we decompose the color of one point into two parts: the RGB part and the spectral part. Formally the color is the sum of the two parts. We use it as the intermediate format during the rendering process. The color of one point $\vec{x}$ can be written as follows

$$Color_{inter}(\vec{x}) = Color_{RGB}(\vec{x}) + Color_{spectrum}(\vec{x})$$

where $Color_{RGB}(\vec{x})$ is the RGB value generated by the interaction between the RGB based material elements and the light sources (represented with RGB model), and $Color_{spectrum}(\vec{x})$ is the spectral effect caused by the spectral elements of the material.

In fact, for one light source, we first calculate its irradiance at the point. Then the radiance (color) of the point will be evaluated according to the type of the material element. If the element is RGB based, we convert the SPD of the spectral light source into RGB and add the value (radiance) to the RGB part of the point color. On the other hand, the RGB light source should be converted to spectrum if the material element is spectrum based. So the color of the point generated by one light source can be written as

$$Color_{inter}(\vec{x}) = \sum_{i=1}^{m} Color^{i}_{RGB}(\vec{x}) + \sum_{j=1}^{n} Color^{j}_{spectrum}(\vec{x})$$

where m is the number of the RGB based material elements, $Color^{i}_{RGB}(\vec{x})$ is the RGB radiance (represented with RGB model) generated by the light source, m is the number of the spectral material elements, and $Color^{j}_{spectrum}(\vec{x})$ is the spectral value which is computed (represented

with spectrum) according to the spectral function. So we write the final equation of the color at one point generated by multiple light sources as follows

$$Color_{inter}(\vec{x}) = \sum_{k=1}^{N}(\sum_{i=1}^{m} Color^{i}_{RGB}(\vec{x}) + \sum_{j=1}^{n} Color^{j}_{spectrum}(\vec{x}))$$

where N is the number of light sources in the scene.

2.3.3 Acceleration

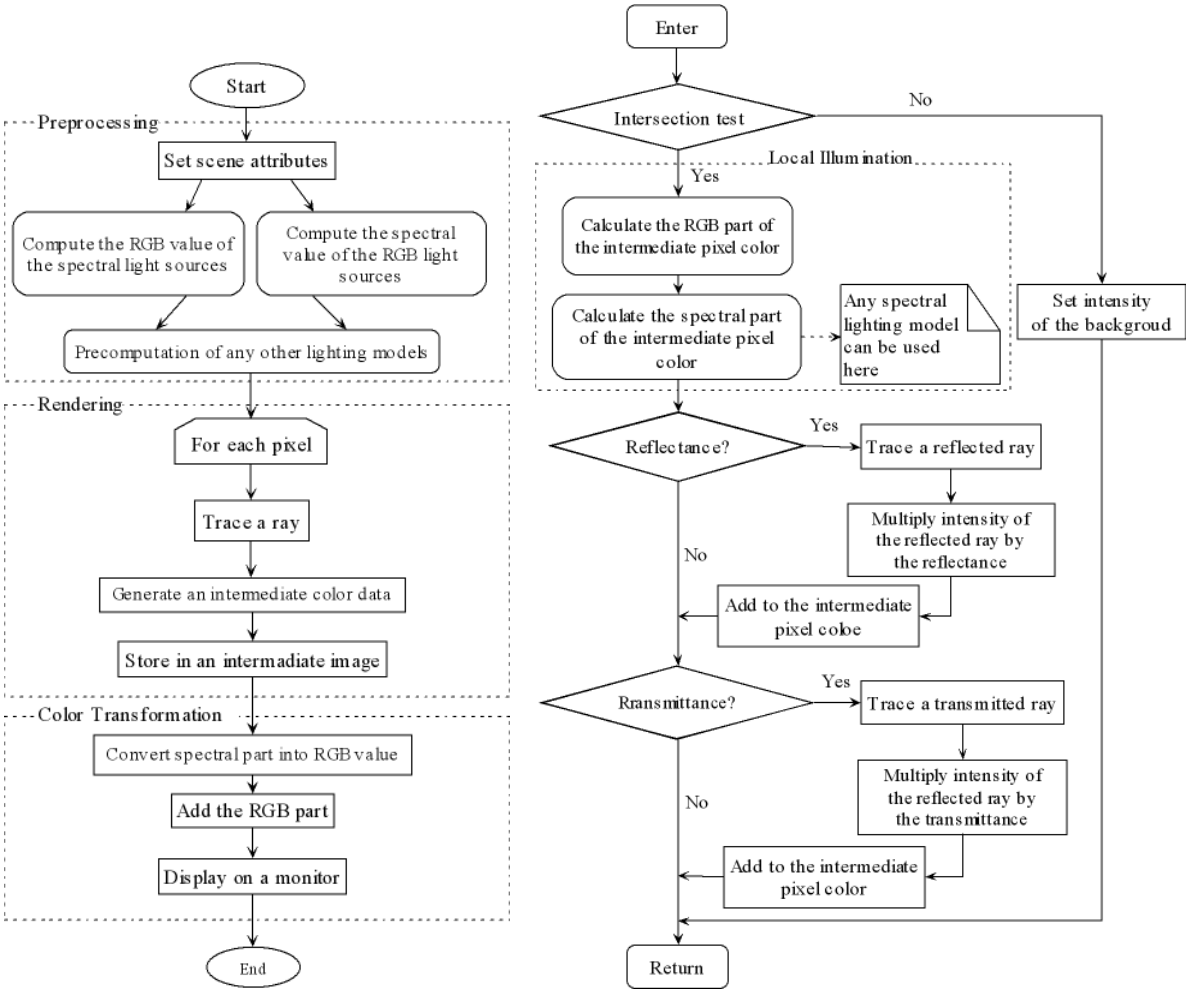
To accelerate the rendering process at run time, we save both the RGB value and the spectral value of all the light sources in the pre-processing step, so we need not do the spectrum-to-RGB or RGB-to-spectrum operation when the irradiance of one point is being calculated. The only work we need to do is to choose the proper value corresponding to the type of the material element.

2.4 Rendering Pipeline

The whole rendering pipeline comprises three stages: preprocessing, rendering and color transformation. In Figure 2.3(a), we show the whole diagram of the work flow. In the first stage, the SPD of the spectral light sources and the spectral functions of the spectral material elements are represented through loading data from the spectral database. The intensity of the RGB light sources and the RGB parts of the materials are also set by the user (or from texture). Then we pre-compute the RGB value of the spectral light sources and the spectra of the RGB light sources. We save these values together with the original data. The second stage is most important: here an intermediate image is generated based on local and global illumination models with ray tracing. As shown in the diagram, here we can change the ray tracer to another render like photon mapping (change the rectangle of "Trace a ray"). The intermediate image is similar to a color image except that for every pixel the information is the combination of a spectrum and a RGB value instead of a color. The detailed flow diagram of the ray tracing process is shown in Figure 2.3(b). Here in the process of calculating the reflectance intensity, if the reflectance of the material is RGB based, we need to convert the spectral part of the reflectance intensity gathered by the reflected ray into RGB. Contrarily, if the reflectance of the material is spectrum based, we need to convert the RGB part of the reflectance intensity gathered by the reflected ray into spectrum. The same for the transmittance calculation. Finally we project the spectral part of the intermediate image into a CIE XYZ color image using

$$X_k = \kappa \int_{\lambda_{min}}^{\lambda_{max}} I(\lambda)\vec{x}_k(\lambda)d\lambda, k = 1,2,3.$$

where X_k are tristimulus values, $\vec{x}_k(\lambda)$ are the CIE XYZ color matching functions, $[\lambda_{min}, \lambda_{max}]$ is the visible range, and κ is a constant. In the last stage, we transform the XYZ value of each



(a) The whole work flow of our framework (b) The work flow of the ray tracing process

Figure 2.3: The rendering pipeline of our framework

pixel to RGB and plus the previous RGB part. This will generate an RGB image for displaying on the screen.

Compared to previous frameworks [GTS+97, SFDC01], the intermediate color format, the separate light-material element interaction and the intermediate image in our pipeline are new elements. Note that we can also store the spectral part of the intermediate image as a spectral image like [SFDC01] to identify errors for particular wavelengths and finding effective improvements. On the other hand, compared with Sun et al.'s [SFDC01] framework, we only add an RGB data which can be stored by three "double" variables for each pixel in the rendering process, the memory increase will not be a problem.

(a) (b)

Figure 2.4: Two examples of our framework. In a), A leaf hides the insect partially. Iridescent colors caused by interference and absorption of light inside the multilayer films can be observed. Compare to Figure 2.2, the color has changed greatly, depending of the new viewing direction and the position of light.

2.5 Results and Discussion

Figure 2.2 shows a natural scene with an Edgar Poe Golden Bug rendered by our system, the material is constructed by coating cooper with a 500 nm gold film, and the specular value for the high light is an RGB value. We use the spectral model described in [HKY+99], and the refractive indices are also obtained from [Pal85]. The plant and the background are both constructed with RGB models. A parallel light source with the spectral distribution of the CIE standard illumination D_{65} [Gla95] is set above the plant. It will cause wavelength computation only when the traced ray intersects with the bug. Figure 2.4(a) shows one bug on a plant with many flowers. In this scene, only the the bug is integrated with spectrum based material. The material is the same as the bug in Figure 2.2, but the position of the camera and the light source (also D_{65}) is different. We can see the different appearance of the iridescence caused by the thin film. Figure 2.4(b) is another example, we change the thickness of the gold film to 300nm, one can see the appearance is different with the previous two images. We can also notice that in the two images of Figure 2.4, the insects only occupy a very small part of the scene. Here our system is much more efficient than the full spectra rendering framework [SFDC01] while the image quality is almost the same. The rendering information of the results is shown in Table 2.1. All the images are generated on a PC of P4 3.2GHz and 1GB RAM. One can see that our system is nearly 15 times faster than the system of [SFDC01]. And we can also see that adding the insect to the scene will only cost a very few additional time comparing with the whole rendering time if the insect will only occupy a small part of the rendering window (for Figure 2.4(a) 1.82 seconds and for Figure 2.4(b) 2.89 seconds).

Table 2.1: The information of result images

Items	Figure 2.2	Figure 2.4(a)	Figure 2.4(b)
Triangle Number of Plant	18086	32848	60894
Triangle Number of Insect	49832	49832	49832
Resolution (Pixels)	680×680	600×600	600×600
Our Rendering Time (Seconds)	7.63	12.84	15.97
Sun's Rendering Time (Seconds)	80.23	188.49	234.13
Our Rendering Time without the Insect (Seconds)	3.12	11.02	13.08
Sun's Rendering Time without the Insect (Seconds)	45.66	173.38	198.52

2.6 Conclusion

This chapter proposed an efficient framework for realistic image synthesis which can use real spectral data and RGB value together as input, retains full spectral interactions between lights and the spectral parts of the material of the objects, and generate images described with a format combining of both RGB and spectrum. We have shown that this framework suffices to describe the natural optical effects in realistic image synthesis, and have facilitated its practical application through a new color representation model - the combination model. While having unified previous research on traditional and spectral modeling and rendering, this new framework provides a useful and efficient basis for simulating general complicated phenomena.

A lot of work is needed to demonstrate the efficiency of the method and to control the visual impact of the rendering in complex global illuminated environments. We plan to implement and test our rendering process in a Photon Mapping Renderer in complex geometric and physical scenes. Our tests use the the *Thin Film Model* for iridescence [HKY+99] and the Sun et al.'s dual method for spectra functions coding [SFDC01]. Improvements can be done in theses two areas. In particular, design a complex biology based micro structure model that can combines interference and diffraction is still an open problem as well as in Computer Graphics than in Optical Engineering.

Chapter 3

Modeling and Visualization of Flower Color Patterns

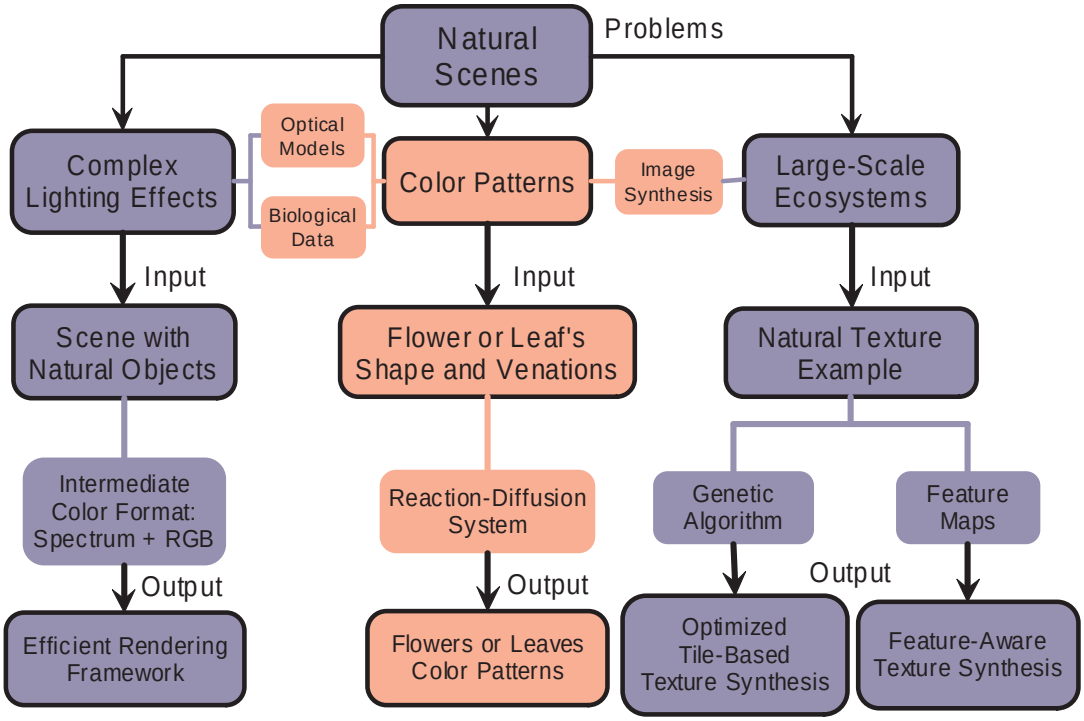


Figure 3.1: Modeling and visualization of flower color patterns.

3.1 Introduction

Flower color patterns, exhibiting various pigmentation and tissue structure influence, are among the most beautiful figures in the natural world. Simulating these patterns has been proved to be an interesting and important challenge for computer graphics. When light interacts with flower petals, several optical effects occur, namely reflection, transmission and absorption. The color we observe on the flowers is formed by the reflective light from the petals, and moreover, the reflection is determined by the absorption, which is real determinant for the appearances of the flowers [Goo76]. Pigments in the flower petals absorb light of certain wavelengths. One pigment has a specific light absorption spectrum, which is mainly determined by its molecule structure. Both the absorbance and distribution of pigments generally decide the flower color pattern.

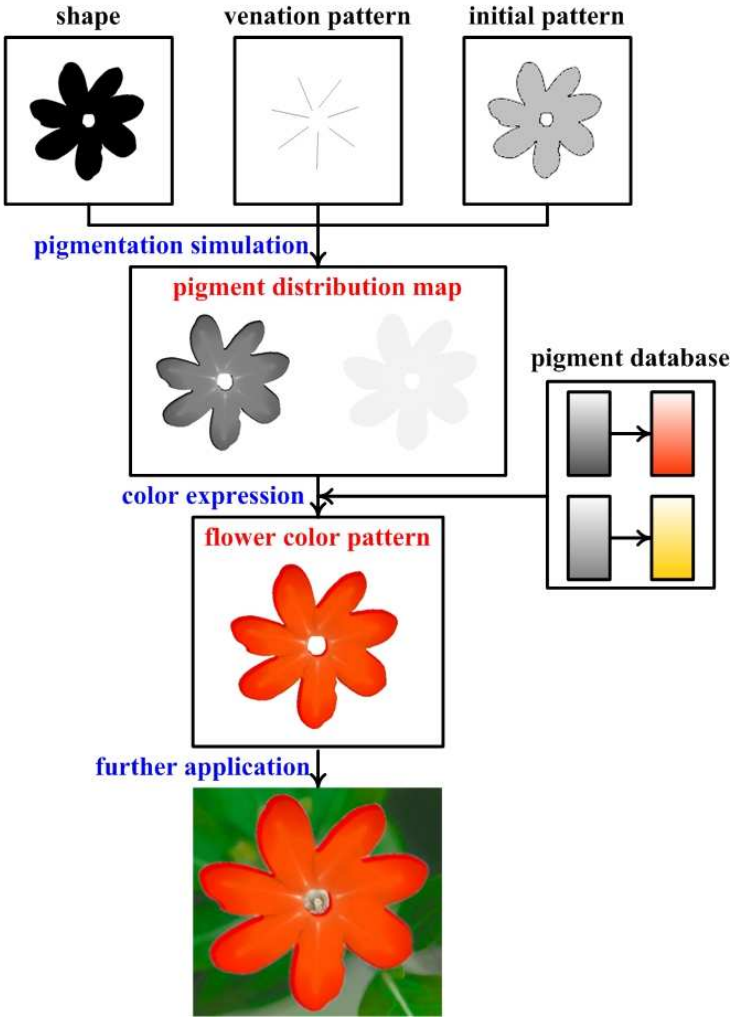


Figure 3.2: Illustration of our system's framework

In this chapter, we introduce an efficient system for modeling and visualization of flower color patterns. Our goal is to build a connection between the user-controllable image synthesis and

real botanical patterns, that is to provide a flexible control allowing users to model a wide variety of biologically plausible flower color patterns. With our system, users can generate realistic flower patterns from corresponding pigment data without the requirement of professional painting skills or any special data acquisition device. Note that we pay more attention to simplifying the modeling process of the flower color pattern and enhancing its diversity than improving the rendering techniques.

The workflow of our system can be divided into three steps. First, we analyze the components of the flower color, and decompose it into two parts: pattern information and environment affection. Then we calculate the pattern information based on the pigment distribution with the reaction-diffusion system, which has been used in natural pattern formation for a long time [Mur03]. According to research results from the botany domain [BMH00], the composition and decomposition of pigment are controlled by some key hormones, which are not created by the petal itself. The hormones are composed in reproduction organs like stamen and ovary, and then transported all over the petal by the vascular system. So pigment manipulation and transportation is highly dependent on the vascular structure. Thus, we modified the standard reaction-diffusion system to fit this demand. The third step is to integrate pattern information into the predefined environment. Combining with rendering systems, our results also can be used as data for texture functions or directly as texture images to render realistic scenes. The whole process from user input to further application, such as rendering, is shown in Figure 3.2.

The remainder of the chapter is organized as follows. Section 3.2 reviews previous work and gives some background botanical knowledge. Section 3.3 introduces the reaction-diffusion system and explains how to fit it with shape and venation information acquired from user input. Section 3.4 discusses how to decompose the flower color space, and its relation with biological knowledge. In section 3.5, we describe how to use our results to synthesize realistic images and discuss about the potential applications. Section 3.6 draws the conclusion and addresses directions for future work.

3.2 Previous Work

Simulation of patterns on living organisms has been studied by researchers of biology, mathematics and computer graphics for a long time. Color patterns are broadly observed in appearance of natural objects, such as plants, animals and insects. It is necessary to simulate the color patterns in order to generate realistic images of those objects. In previous work, attention has been paid to mammalian, fish, sea shell, tree bark, etc. However, pattern formation of flowers is still blank.

Instead of making strict biologically-based simulations, mathematical models have been used to simulate the color pattern formation in nature. Although there is no final conclusion on their validity, they are able to produce visual plausible results with relative higher performance. Previous work includes reaction-diffusion system used in animal fur simulation [Tur91], sea shell pigmentation [FMP92], and fish skin pattern formation [Pai00]; the Clonal Mosaic model for mammalian coat pattern formation [WFR98]; a semi-empirical model for bark gen-

eration [LN02]; an open diffusion limited aggregation model for lichen growth and propagation simulation [DGA04]; and a biologically-motivated algorithm for generating venation patterns [RFL+05].

For synthesizing realistic images, various botanically-based modeling and rendering systems have been developed to support the design and visualization of plants. Lintermann and Deussen [LD99] present a modeling method for generating branching objects. Mündermann et al. [MMPP03] extended the concept of branching structure and developed an interactive method for modeling lobed leaves. Ijiri et al. [IOOI05] supplied a convenient tool motivated by biology knowledge for flower geometry modeling, which can preserve correct botanical structures. Govaerts et al. [GJVU] developed a multi-layered model for leaves based on its structure properties. Baranoski and Rokne [BR97] presented an algorithmic reflectance and transmittance model for plant tissue oriented to computer graphics applications. Wang et al. [WWD+05] developed a framework for rendering of plant leaves in real-time with global illumination. The spatially-variant BRDFs and BTDFs employed to describe leaf appearance are stored in a set of parameter maps. This method is efficient and produces very realistic results. However, it is tightly limited by measured data and will be inconvenient when plants with many patterns are required in one scene.

3.3 Pigmentation Simulation

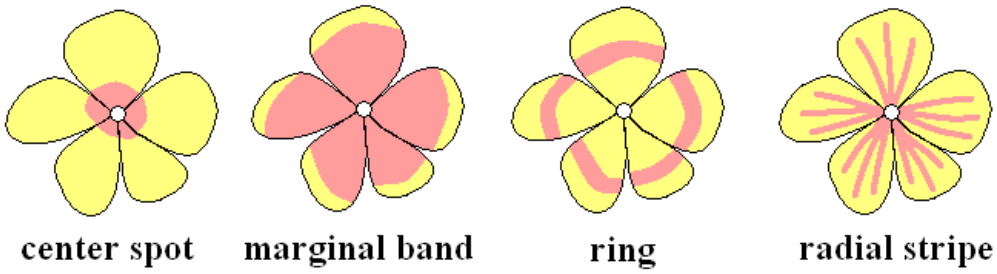
In this section, we explain how to simulate the pigment distribution of flower petals. We first input the information of petal shape, venation, and initial distribution. Then we simulate the pigmentation with a reaction-diffusion system, and save the normalized simulation result in a greyscale map for the next step.

There are two kinds of flowers: monochrome flowers, and flowers bearing variegation [ZGC05]. The latter one often produces near-regular patterns, which we are interested in. Common variegation patterns, including center spot, marginal band, ring and radial stripe, are shown in Figure 3.3. These patterns may appear simultaneously in one flower. The goal of our system is to generate multiple patterns based on single input by variating the parameters of the modified reaction-diffusion equations. Therefore, our system is able to generate various flower color patterns with little user-intervention. The space-time distribution of flower color pigment is the most important determinative factor for flower patterns [ZGC05]. To achieve our goal, it is essential to simulate this distribution.

3.3.1 Preliminaries

The input for our system consists of: (1) the initial state (shape and venation information of the flower, and the initial pigment distribution), (2) parameters characterizing the spread and interplay of pigments, and (3) functions and parameters defining flower color expression.

The shape is specified by the user as a black-white image that defines the petal's range. The



(a) Sketch map of common flower color patterns in nature

(b) Corresponding result samples

Figure 3.3: Flower color patterns.

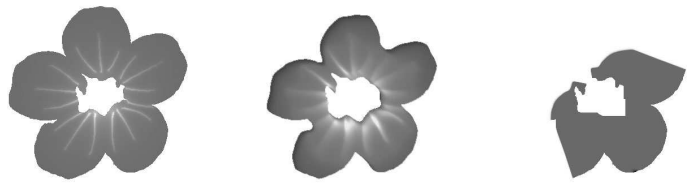
venation pattern is given as a set of vein nodes, including a vein root and vein width. According to the canalization hypothesis [BMH00], the vein is composed of cells with higher capability in auxin transportation. We assume that the connectivity coefficient between two adjacent mesophyll cells (Here we use *vascular cell* to refer to cells which form the venation of leaf, and use *mesophyll cell* to refer to all other kinds of cells in the leaf). equals to p_{mm}; the connectivity coefficient between a vascular cell and an adjacent mesophyll cell is p_{mv}; the connectivity coefficient between two adjacent vascular cells is p_{vv}. p_{mm}, p_{mv} and p_{vv} are empirical constants. In our system, we set $p_{mm} = 1$, $p_{mv} = 2$ and $p_{vv} = 10$. Moreover, we define that the connectivity coefficient between two non-adjacent cells is directly proportional to the vein width and is inversely proportional to the distance between them. Then, we define the transportation function between two cells a and b as follows:

$$I(a,b) = \begin{cases} p_{mm}/\|a,b\| & \text{(3.1a)} \\ p_{mv} \cdot w/\|a,b\| & \text{(3.1b)} \\ p_{vv} \cdot \bar{w}_{ab}/\|a,b\| & \text{(3.1c)} \end{cases}$$

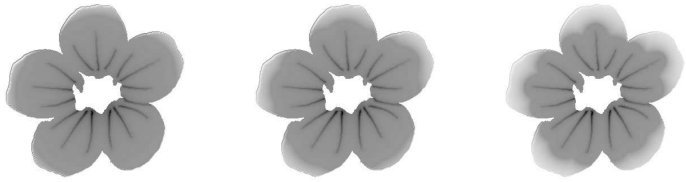
In Equation 3.1a, neither a or b is a vascular cell; in (3.1b), either a or b is a vascular cell and its vein width is w; in Equation 3.1c, a and b are both vascular cells and connected by the vein with average width $\bar{w}_{ab}$. In our system, we determined the parameters from experience.

The venation pattern can be drawn or extracted from photos. Artificial methods such as [RCSC03] and [RFL+05] can also be used to generate a vascular system. Although the method for obtaining the venation pattern makes no theoretical difference to the following steps, the venation pattern will obviously affect the final results.

Normalized pigment distribution is represented by a grey-scale map called pigment distribu-



(a)



(b)

Figure 3.4: Different patterns generated by parameter adjustment. (a) is three pigmentation map of pigment A. It shows the influence of the parameter D (Here it is D_A corresponding to pigment A). From left to right, $D_A = 0.1, 0.3, 0.5$. (b) shows the influence of auxin transportation function. From left to right, $t = 1, 2, 5$, which means the transportation ability of vein cells becomes stronger.

tion map. Each kind of pigment corresponds to one map. Its concentration at each locus is represented by the grey level of the pixel. The darker the pixel is, the higher the pigment concentration is. Standard black corresponds to the pigment concentration of the preset database, which contains the pre-computation results of chromaticity coordinates corresponding to different pigment types and concentrations.

The initial pigment distribution is given by users as several grey-scale images, which follow the before-mentioned definition of pigment distribution map. However, our algorithm does not require any specific initial pattern to achieve realistic results.

3.3.2 Pigment Composition Mechanism

One flower usually contains several pigments, which cooperate to decide the flower color pattern. However, some flowers only contain one kind of pigment. For such case, we introduce a 'blank pigment' which will not be taken into account of flower color expression in the later steps.

We use reaction-diffusion equations to describe the interaction among different pigments. Based on the assumption that the pigment composition of one cell can be calculated from its neighbors, we can simulate the pigmentation of following moments in the growing process if an initial state and transportation kinetics are given. The initial state can be freely assigned by the user. For convenience, we use an uniform distribution map as the initial state in our experiments.

In the two-pigment case, we assume A_t and B_t are the time-dependent concentrations of two

pigments A and B. The reaction-diffusion equations describing the general relation between A_t and B_t are given by

$$\begin{cases} \dfrac{\partial A_t}{\partial t} = F(A_t, B_t) + D_A \nabla^2 A_t & \text{(3.2a)} \\ \dfrac{\partial B_t}{\partial t} = G(A_t, B_t) + D_B \nabla^2 B_t & \text{(3.2b)} \end{cases}$$

Where F and G are the kinetics, and D_A, D_B is the positive constant diffusion coefficients. Different kinetics have been invented to describe the pattern formation process in nature. In our system, we employ the following kinetics designed by Schnakenberg [Mur03]:

$$\begin{cases} F(A_t, B_t) = k_1 - k_2 A_t + k_3 A_t^2 B_t & \text{(3.3a)} \\ G(A_t, B_t) = k_4 - k_3 A_t^2 B_t & \text{(3.3b)} \end{cases}$$

k_1, k_2, k_3 and k_4 are positive empirical constants [Mei82].

The Laplacian is $\nabla^2 A = \frac{\partial^2}{\partial x^2} A + \frac{\partial^2}{\partial y^2} A$. It is difficult to compute the Laplacian in the continuous domain, thus we discretize the continuous plane to a grid, and then we can use the finite difference approximation to compute the Laplacian:

$$l_{i,j} = \frac{1}{4}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) - u_{i,j} \tag{3.4}$$

where $l_{i,j}$ is the Laplacian at point (i, j). $u_{i,j}$ is the value at point (i, j).

To represent the effect of vein, we modify the neighbors of (i, j) in Equation 3.4 with auxin transportation function(Equation 3.1): $u_{m,n}^{(i,j)} = tI((i, j), (m, n)) \cdot u_{m,n}$, where t is the proportion coefficient of venation influence. Then

$$l_{i,j}^{(u)} = \frac{1}{4}(u_{i+1,j}^{(i,j)} + u_{i-1,j}^{(i,j)} + u_{i,j+1}^{(i,j)} + u_{i,j-1}^{(i,j)} - 4u_{i,j}) \tag{3.4′}$$

So substitute (3.4′) to (3.2), the modified discrete form reaction-diffusion equations on a grid can be rewritten as:

$$\begin{cases} \Delta A_{i,j} = S_A(a_{i,j}^2 b_{i,j} - K_A a_{i,j} + \beta) + D_A l_{i,j}^{(a)} & \text{(3.5a)} \\ \Delta B_{i,j} = S_B(1 - K_B a_{i,j}^2 b_{i,j}) + D_B l_{i,j}^{(b)} & \text{(3.5b)} \end{cases}$$

where

$\Delta A_{i,j}$: concentration change of pigment A at (i, j);

$a_{i,j}$: concentration of pigment A at (i, j);

$l_{i,j}^{(a)}$: influence to the concentration of pigment A from neighbors of (i, j);

S, K, D, β: constants depending on transportation kinetics. If we use the kinetics mentioned in Equation 3.3, the constants in Equation 3.5 will be $S_A = k_3$, $K_A = \frac{k_2}{k_3}$, $\beta = \frac{k_1}{k_3}$, and $S_B = k_4$, $K_B = \frac{k_3}{k_4}$.

Table 3.1: Symbol and meaning of the parameters in our system

Symbol	Meaning
S	proportion coefficient between diffusion and reaction
D	reaction coefficient
k	diffusion coefficient
β	dissipation coefficient responsible for slight irregularity
t	proportion coefficient of venation influence
c	normalized concentration of pigments

The reaction-diffusion system is able to simulate both the decrease of inherent pigment and the increase of subsequent pigment during the growth. The parameter adjustment technique can be found in [Tur91]. Our modification embeds the effect of vascular transportation into the primary mechanics, and leads to more control of the final results. To simulate the interplay of more than two pigments, more complex kinetics from [Mei82] should be used. The influence of the auxin transportation function is demonstrated in Figure 3.4(b).

We define a certain concentration value as standard concentration C_0. The pigment concentration calculated by the modified reaction-diffusion equations are normalized by $c = C/C_0$, and then stored in the greyscale pigment distribution map. In our system, we usually use the highest concentration in the pigment color database as standard concentration. Symbols and meanings for all the parameters used for pigmentation simulation are shown in Table 3.1. After the pigment distribution map is computed, it will be painted with pigment color information.

3.4 Flower Color Expression

To paint the greyscale pigment distribution map generated in previous step, we first construct a pigment database to store the absorb color data of all pigments of different concentrations, and some other information which is not related with the pigment distribution map, such as the lighting environment. After these pre-computation is finished, we calculate the color pattern of the petal based on the pigment distribution map in runtime.

From the physics point of view, flower color is determined by light absorption of pigments, which is a function of wavelength. The wavelength range of visible light extends approximately from 380nm to 770nm. Thus, the effective energy of an energy spectrum sensed by human eyes is

$$V = \int_{380}^{770} P(\lambda)v(\lambda)d\lambda, \tag{3.6}$$

and the tristimulus values are

$$X^{(k)} = \int_{380}^{770} R(\lambda)\bar{x}_k(\lambda)d\lambda, k = 1,2,3, \tag{3.7}$$

where λ is the wavelength. Here we assume there is no interplay among the color performances of different pigments.

3.4.1 Pre-Computation of Pigment Color Performance

The standard pigment color performance is calculated based on measured data under standard conditions. Suppose there is a pigment p with standard absorption spectrum $A(\lambda)$. The luminance spectrum is $L(\lambda)$, which is considered to be a constant in the simulation process. The transmittance spectrum is $T(\lambda)$, which depends on the material of the petal. It varies greatly due to the species, and can be somewhat affected by the pigment concentration. As the influence of pigment concentration to transmittance is not obvious, in our system, we use the measure data from several typical plant tissue types to make an approximation of the transmittance [TSP00, SG02], and consider it as constant for a certain flower. Then the reflection spectrum $R(\lambda)$ can be calculated by $R(\lambda) = L(\lambda) - T(\lambda) - A(\lambda)$. We divide the right side of the above equation into two parts: $L(\lambda) - T(\lambda)$ and $A(\lambda)$, and define

$$X_0^{(k)} = \int_{380}^{770} (L(\lambda) - T(\lambda))\bar{x}_k(\lambda)d\lambda \tag{3.8a}$$

$$X_A^{(k)} = \int_{380}^{770} A(\lambda)\bar{x}_k(\lambda)d\lambda \tag{3.8b}$$

Substitute (3.8a) and (3.8b) into (3.6), it can be rewritten as

$$V = V_0 - V_A \tag{3.6′}$$

Similarly, (3.7) can be also rewritten to a composition of two parts

$$X^{(k)} = X_0^{(k)} - X_A^{(k)}, k = 1,2,3. \tag{3.7′}$$

The standard chromaticity coordinates (corresponding to standard concentration C_0) of the pigment are:

$$(x,y) = \left(\frac{X^{(1)}}{X^{(1)} + X^{(2)} + X^{(3)}}, \frac{X^{(2)}}{X^{(1)} + X^{(2)} + X^{(3)}} \right)$$

For pigment of normalized concentration c, according to Beer's law [Mac81], its absorption spectrum approximately follows linear relationship: $A_c(\lambda) \approx c \cdot A(\lambda)$, so its lightness is $V_c = V_0 - c \cdot V_A = V + (1-c) \cdot V_A$, and its tristimulus values are $X_c^{(k)} = X_0^{(k)} - c \cdot X_A^{(k)} = X^{(k)} + (1-c) \cdot X_A^k$, $k = 1,2,3$.

By Equation 3.6′ and 3.7′, we pre-compute values which have no relation with pigment concentration based on measure data of lighting condition and transmittance indices. Only the values related with pigment concentration need to be computed at runtime.

(a) (b) (c) (d)

Figure 3.5: Single pigment color expression. (a) to (d) are generated based on the same distribution map, and the standard color of the pigment is red, orange, saffron yellow, and vivid yellow.

(a) (b) (c) (d)

Figure 3.6: Effects of concentration. The flower color changes when the standard concentration changes. From (a) to (d), the standard concentration C_0 is set to 0.25, 0.5, 0.75 and 1.

3.4.2 Runtime Flower Color Expression

For single-pigment flowers, the flower color expression is determined by the absorption spectrum of that pigment.

Even only with one single pigment, the flower color varies and forms special patterns because of the inhomogeneous distribution. With the same pigment distribution map, if we change the type of pigment, we can get flowers with similar pattern but different colors. Figure 3.5 shows some single-pigment samples generated with our system based on the same pigment distribution map but with different pigment. Change of concentration has stronger influence on saturation than on hue, and it has a linear relation with lightness. Figure 3.6 illustrates how the color expression of the same pigment varies because of different concentration.

In most cases, flower color is not produced by one single pigment. Several pigments cooperate to show a color which is different from all of their original color expressions.

Suppose there are pigments p_i of concentration c_i, and their typical absorption spectra are $A_i(\lambda)$, $i = 1$ to n. Then the total absorption spectrum is $A(\lambda) = \sum_{i=1}^{n} c_i \cdot A_i(\lambda)$, so its tristimulus values are:

$$X'^{(k)} = X_0^{(k)} - \sum_{i=1}^{n} c_i \cdot X_{A_i}^{(k)}, k = 1,2,3. \tag{3.9}$$

(a) (b) (c) (d)

Figure 3.7: Mixed pigments color expression. (a) and (b) are the distribution maps of the two pigments in the flower petal. (c) and (d) are flower color patterns generated by assigning different kinds of pigments to (a) and (b).

(a) (b)

Figure 3.8: Pattern generation result samples.

Its lightness is

$$V_c = V_0 - \sum_{i=1}^{n} c_i \cdot V_{A_i}. \tag{3.10}$$

In case of mixed pigments, the relationship between color and concentration c_i is not linear anymore, though each separate pigment keep its linear relationship. Some mixed pigments samples generated with our system are shown in Figure 3.7. Figure 3.7(a) and 3.7(b) are distribution maps of two different pigments existing in the flower. Figure 3.7(c) and 3.7(d) are results based on same distribution maps but with different pigment contents.

By choosing different pigment content, our system is able to generate numerous flower color patterns corresponding to different species, pigment types and environment conditions. We give an illustration of the aging process of a maple leaf in section 3.5.

3.5 Results and Applications

We implemented our system on a PC with P4 3.2GHz CPU, 1GB RAM and a Geforce 7800 GT graphics card. In this section, we report flower color patterns generated with our system, and rendering results using reflectance data acquired with our algorithm. The performance of our system is shown in Table 3.2. The parameters used to generate the sample results are listed in Table 3.3.

Table 3.2: Performance statistics for figures

Figure Number	Plant Name	Resolution (pixels)	Compute Time (seconds)
3.3(b), 3.8(b)	Petunia	549x528	466
3.4	Geranium	430x387	212
3.5, 3.6, 3.7, 3.8(a)	Daisy	275x276	110
3.9	Maple	577x377	236
3.10	Water Lily	214x364	132

Table 3.3: Parameters used for part of the figures

Figure Number	Parameter			
	S	D_A	D_B	t
3.3(b)-1	0.1	0.15	0.1	2
3.3(b)-2	0.1	0.15	0.1	10
3.3(b)-3	0.1	0.45	0.1	2
3.3(b)-4	0.1	0.15	0.5	2
3.5, 3.6, 3.7	0.05	0.5	0.1	2
3.9	0.1	0.15	0.1	20
3.10	0.05	0.25	0.25	5

Pattern generation The primary contribution of our work is a sufficient tool for generating and editing flower color patters. Given a monochrome flower, our algorithm is able to change its hue, and assign different variegation to it. We split the original image in YIQ space, and extract the shape and venation information. Then we redefine the I channel with user-appointed pigment, and replace the Q channel with the pigment distribution map. The Y channel, which includes geometry detail, material property and lighting environment, is also modified based on the pigment distribution map. By combining them together, a flower with the original shape but a different pattern is synthesized. Two samples are shown in Figure 3.8. The left column is the user input reference images without obvious pattern, while the right column is the results generated by our system based on the shape and venation information contained in the reference images.

In order to acquire profound understanding and intuitive impression of the parameters in the algorithm, we explored the parameter space by varying one parameter while keeping other parameters constant. Two examples of the exploration are shown in Figure 3.3 and Figure 3.4. They exhibit the pattern diversity capability of our algorithm. Different pigments have different absorption spectra, thus express different colors. Figure 3.5, Figure 3.6 and 3.7 show that we can paint a pattern with different colors by choosing different pigment contents.

Aging With flexible adjustment of parameters, we are able to simulate the color change during the flower aging process Because of the similarity between leaves and petals, our algorithm is

(a) (b) (c) (d)

Figure 3.9: Maple leaf color change in the aging process. From (a) to (d), our result show the typical maple leaf color of early spring to late autumn.

also applicable to leaves, which appear even more frequently in natural scenes. In Figure 3.9, we show some sample results of the seasonal color change of a maple leaf as an example to demonstrate the ability of simulating color transformation in the aging process of our system.

By color, the aging process of maple can be divided into three stages: yellowish green, flaming orange, and brilliant red. This transformation is caused by the change of pigment. As autumn comes, concentration of chlorophyll falls down because of the sunshine and temperature, so carotenoid is able to present its yellow color. Then anthocyanin is composed, and maple leaves change to red at the end [Lea86]. A simple illustration of this process is shown in Figure 3.9. Note that here we only use imaginary data to explore the capability of our system.

Rendering The result of our framework can be used directly as texture in rendering tools. However, the flower petal is typically a translucent multi-layered material. To achieve highly realistic results, other techniques, such as BRDF, must be used instead of simple textures. Our algorithm can also be embedded into other models to achieve higher rendering quality. For physically-based models such as ABM(algorithmic BDF model) [BR97], the mechanisms of pigments absorption are taken into account only assuming that the pigment distribution is homogeneous. Our model can be embedded in ABM to reveal the effect of inhomogeneous pigment distributions.

Figure 3.10 shows rendered images with our pattern results as textures. The pre-computation time for patterns (shown in Table 3.2) varies from several seconds to several minutes, which is mainly decided by the resolution of the pattern.

3.6 Conclusion

We have presented an algorithm for flower color pattern modeling and visualization. Cooperated with reaction-diffusion system and flower color analysis, our system combines pigment distribution simulation and flower color expression together under a unified framework. Our system is able to form a wide variety of biologically plausible flower color patterns, which are widely observed in nature. The result patterns can be used to produce more realistic virtual scenes, and our system can also be directly embedded into other modeling or rendering systems to improve their results. Besides biologically-related simulation and application, our work demonstrates that even though parameters are not available from measure data, an approximate

(a) Water lilies

(b) Zoom in view of the flower

Figure 3.10: Samples of rendering results with our patterns. Because the focus of this chapter is not on flowers' rendering, we only use a simple ray-tracer for the example. The rendering quality can be ameliorated by using an advanced renderer.

simulation can be made based on a few empirical parameters. Thus, it reveals the ability of visualizing imaginative flowers, with which data acquisition is impossible.

For future work in this area, extend our algorithm to pattern generation for other objects with similar properties is an interesting direction.

Chapter 4

Realistic Simulation of Seasonal Variant Maples

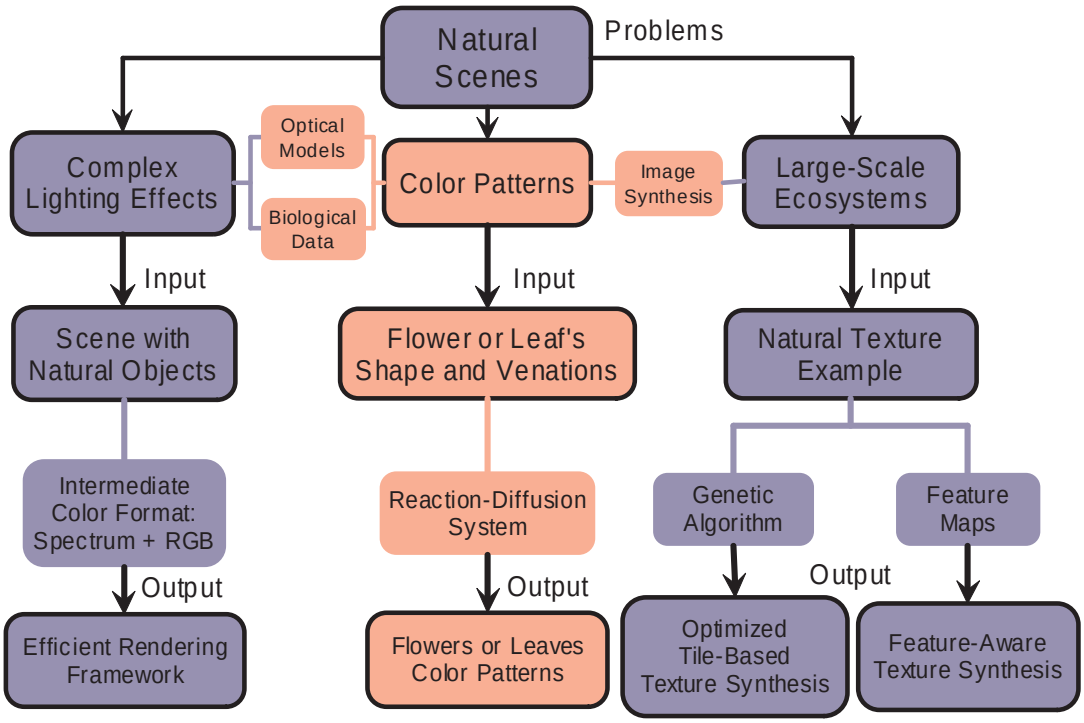


Figure 4.1: Realistic simulation of seasonal variant maples.

4.1 Introduction

The appearance of computer-generated trees has been significantly improved with the help of recent advances in both modeling and rendering techniques. We are able to generate realistic pictures of natural scenes consisting different kinds of trees. However, an effective simulation of leaf color seasonal transformation is still not available. This transformation is caused by the change of pigment. As autumn comes, concentration of chlorophyll falls down because of the sunshine and temperature, so carotene is able to present its yellow color. Then anthocyanin is composed, and maple leaves change to red at the end [Lea86]. This seasonal change happens on different scales: from the large scale such as a forest to the small scale such as one piece of leaf. For a forest, the trees at different position change their leaf color at different time; for a tree, the leaves at different position change their color at different time; even for a single leaf, different parts of it can have very different colors. Our goal is to make a realistic simulation of this beautiful process.

Figure 4.2: Rendering result of our system.

In this chapter, we present a framework that can simulate the seasonal variant appearances of maples realistically. A forest consisting of both maples and other trees can also be rendered by our system. The appearance of each maple tree is determined by its surrounding environment and climate conditions. The color of the maple leaves ranges from greenish yellow to flaming orange to brilliant red between spring and later autumn. We use texture mapping to display the patterns of maple leaves. Various leaf patterns are used for a single maple tree.

We set several key points in the leaf color seasonal change process to divide it into three typical stages: from greenish yellow to flaming orange and then to brilliant red. We use the concentration of chlorophyll (for green), carotene (for yellow) and anthocyanin (for red) as main

parameters to calculate the pixel colors of the leaf texture. Sunlight, temperature and precipitation are used as determinative parameters for texture choice of leaves. A colorful autumn is produced under the joint effect of abundant sunlight, low temperature and ample precipitation. They can accelerate chlorophyll decomposition and promote anthocyanin formation.

This chapter is organized as following: Section 4.2 makes a review of previous works, and introduces some necessary knowledge from the biology domain. Section 4.3 describes our system in detail, and section 4.4 shows some sample results of our system. Finally, section 4.5 draws the conclusion and points out some potential directions for the future work.

4.2 Previous Work

4.2.1 Simulation of Plant-Environment Interaction

Simulation of environment influence on plant is an interesting subject for both computer graphics and biology. Related works have been done on all the three scales mentioned in section 4.1. In the leaf scale, the inhomogeneous pigment distribution of leaf is taken into count to simulate seasonal change of leaf color. In the tree scale, one leaf is generally considered to be one color, and attentions are mainly paid to leaf arrangement. In the forest scale, even one whole tree can be considered to be only one color, and the research emphasizes on tree distribution. In this scale, to get higher performance, the differences among leaves are usually ignored, so our work emphasizes on tree and leaf scale.

Chiba et al. [COMS96] presented a procedural method for simulation of both leaf arrangement and leaf seasonal color based on the estimation of the amount and direction of sunlight. Mochizuki et al. [MCK$^+$01] simulated the autumn coloring process of both maple leaves and entire maple trees based on fractal model. Mech et al. [MP96] presented a L-system based framework to simulate interactions between plants and environment at plant architecture level. Deussen et al. [DHL$^+$98, DCSD02] introduced a system for simulating and rendering large-scale plant ecosystems, performing both biologically-based plant distribution simulation and acceptable rendering efficiency.

However, most of the previous works focus only on the effect of a single environmental factor, such as [MCK$^+$01] on temperature, [BDE04] and [COMS96] on sunlight. Few of them consider the joint influence of more than one factors at the same time. In our system, we make an instructive try of considering several environmental factors under a unified framework.

4.2.2 Biology Explanation of Maple Seasonal Color Variance

Though the complicated reactions during maple leaf color change have not been fully understood yet, we have sufficient knowledge of the role pigments and environment factors play in this process to make a realistic simulation. The leaf color is the joint result of three key pigments: chlorophyll, carotene, and anthocyanin. Chlorophyll is responsible for the green, which

is the main color of leaves in most of the time. It is a key component for photosynthesis. It keeps on being decomposed and synthesized during the whole process of leaf growth, until the synthesis reaction stops when temperature is low. Comparing with chlorophyll, carotene, which expresses the yellow color, is more stable. In the autumn, it still exists when chlorophyll is almost disappeared. It helps with energy transfer. Anthocyanin is produced by reactions between sugar and some proteins. This reaction only happens when the concentration of sugar is quite high, and also requires light. It is responsible for the red-purple color of leaves, and can protect leaf from getting burned by the light.

Generally speaking, a brilliant autumn with colorful maples depends on not only abundant precipitation during the summer, but also bright weather and distinct day-night temperature difference during the autumn [Lea86]. A wet summer ensures the composition of adequate chlorophyll, which act an important role in sugar production. Those sugar not only nourish the plant, but also will later decompose to anthocyanin resulting in the red hue of maple leaf. Bright weather and distinct temperature difference in the autumn will give plant the sign of stopping composing chlorophyll, so that the effect of other pigments will show out.

From physical point of view, leaf color is determined by light absorption of pigments, which is a function of wavelength. The wavelength range of visible light extends approximately from 380nm to 770nm. Thus, the effective energy of a spectrum sensed by human eyes is

$$V = \int_{380}^{770} P(\lambda)v(\lambda)d\lambda, \tag{4.1}$$

and the tristimulus values are

$$X^{(k)} = \int_{380}^{770} R(\lambda)\bar{x}_k(\lambda)d\lambda, k = 1,2,3, \tag{4.2}$$

where λ is the wavelength. Here we assume that there is no interplay among the color performances of different pigments. The absorption spectra of the three before-mentioned pigments are shown in Figure 4.3. In our system, we do not directly simulate the change in color space, but compute the concentration change of the three key pigments instead.

4.3 Simulation and Visualization of Maple Seasonal Color Variance

Our system simulates the leaf color seasonal change of maples emphasizing on tree scale. The whole system consists of three steps: (1) environment configuration, (2) climate influence simulation and (3) leaf texture acquisition. Step (1) decides the general color change timing of the maple tree based on its local environment. Step (2) makes further adjustments to the timing determined by step (1) considering the influence of climate in the specific case, and computes the pigment concentration for every leaf. Step (3) generates the leaf textures with the pigment concentration data. In this section, we describe the three steps in detail.

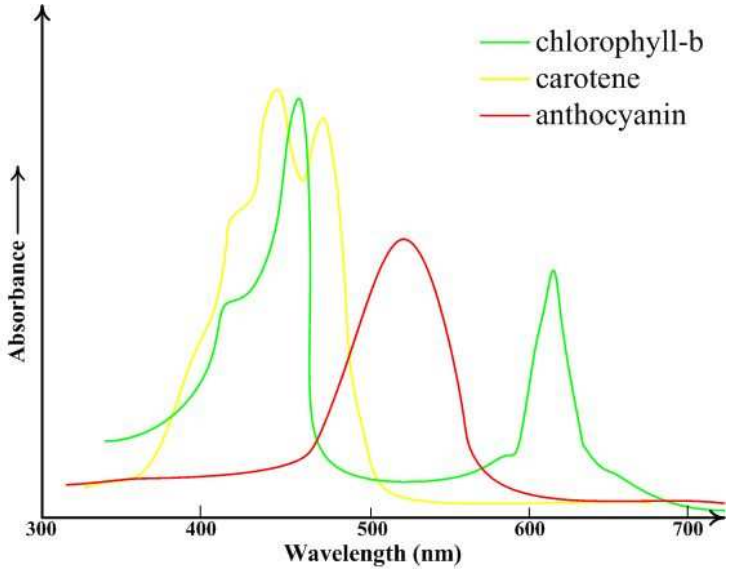


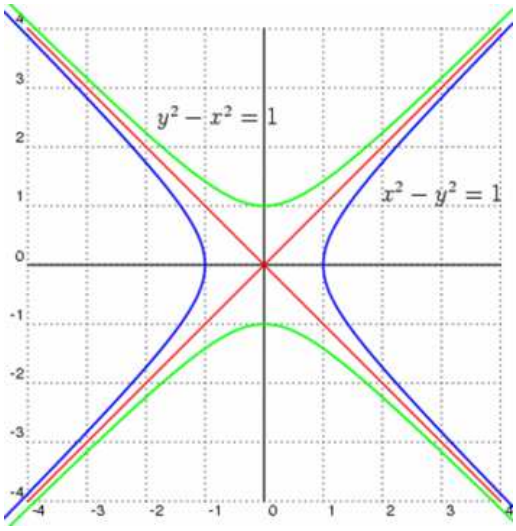
Figure 4.3: Absorbance of some important plant pigments for leaf color. Red, green and yellow lines represent the absorbance of anthocyanin, chlorophyll and carotene [Hop99].

4.3.1 Environment Configuration

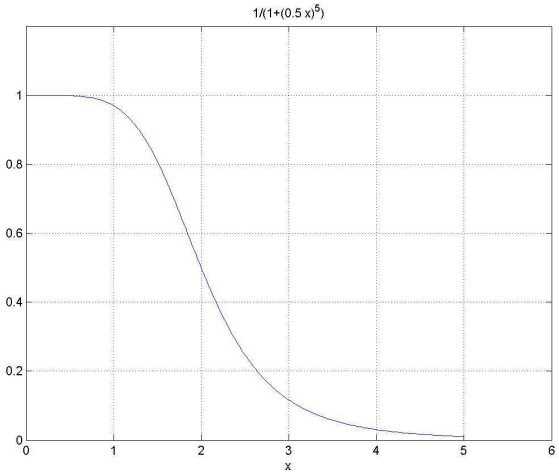
Before consider the specific climate condition in a certain year, in this step, we first configure the normal climate parameters based on these basic environment conditions. Some certain environment conditions such as the latitude, longitude and altitude will generally decide the average climate parameters such as sunlight amount, precipitation and temperature, thus the approximate timing of maple leaf color change. Some previous work in biology domain has already provided some experience models on this subject, such as the ACLT (Autumn-Color-Logical-Timing) system [MCK+01]. In our system, we construct a look-up table for the relation between geography position and normal climate circumstances based on long term observation data [cli]. Surrounding objects such as buildings and other trees will decrease the amount of light and rain and even heat the tree can receive, so in our system, we consider them as a weight coefficient u. After we determine the climate conditions including sunlight, temperature and precipitation, we simulate the concentration variation of the three key pigments.

The concentration of chlorophyll is in a dynamic balance of composition and decomposition. Its composition reaction requires more sunlight and higher temperature. So when autumn comes, the lack of sunlight and low temperature slow down the composition efficiency. As the concentration of chlorophyll falls down, carotene, which is more stable, is able to present its yellow color. At the same time, the low temperature also promotes the composition of anthocyanin, which expresses red in the acid chemical environment of maple leaves, so the leaves turn to red in the end [Lea86]. A simple illustration of pigment concentration variance in this process is shown by Figure 4.5(a). Most relations between climate and pigment concentration can be approximated by a linear relation. The linear coefficient α is

$$\alpha_k = \frac{u}{A_k} \int_0^t I_k(t)dt, k \in s,d,p \tag{4.3}$$



(a) The rectangular hyperbola model



(b) The switch-off threshold response curve

Figure 4.4: Mathematic models for pigment simulation.

where u is the weight for surrounding environment, A is the sum amount of the whole year in normal climate condition, and $I_k(t)$ is the weather circumstance at time t. The lower index s is for sunlight, d for temperature and p for precipitation. Type of the pigment will be pointed out by the upper indices.

For chlorophyll, we assume $U_c(t)$ and $U_d(t)$ are its time-dependent composition and decomposition speed. Its concentration at time t can be given by

$$c_{ch}(t) = \alpha_d^{ch} \alpha_p^{ch} \int_0^t (U_c^{ch}(t) - U_d^{ch}(t))dt \tag{4.4}$$

where α_t and α_p are influence factors of temperature and precipitation. Its composition and decomposition relies on not only sunlight but also other chemicals in the leaf, so we simulate them with the two-substrates model, which is a rectangular hyperbola as shown in Figure 4.4(a).

For a two-substrates reaction, suppose the utilization speeds of the two substrates are equivalent, then the two-substrates model can be described by

$$U = \frac{k' X_1 X_2}{1 + L_1 X_1 + L_2 X_2 + L_{12} X_1 X_2} \tag{4.5}$$

U is the reaction speed. k', L_1, L_2 and L_{12} are constants. X_1 and X_2 are the concentration of the two substrates. To simplify the model, we consider one of the substrates as constant. Then Equation 4.5 can be written in the form

$$U = \frac{kX}{K + X} \tag{4.6}$$

where U is the reaction speed, k and K are constants, X is the concentration of substrate. Equation 4.6 is a common-used rectangular hyperbola model: Michaelis-Menten model [Tho75]. To reveal the influence of sunlight, we let constant $k = \alpha I$ (I is light flux density, and α is the utilization efficiency constant).

For anthocyanin, temperature has a threshold effect to it: only when temperature is low enough will the reaction of anthocyanin composition start. Thus we use the switch-off threshold model to simulate this process. It corresponds to reactions which only happen when the value of influence factors reaches a certain level. So the concentration of anthocyanin at time t is

$$c_{an}(t) = \alpha_s^{an} \alpha_p^{an} \int_0^t U^{an}(t) dt \tag{4.7}$$

where the reaction speed U is

$$\frac{U}{k} = \frac{1}{1 + (X/X_c)^n} \tag{4.8}$$

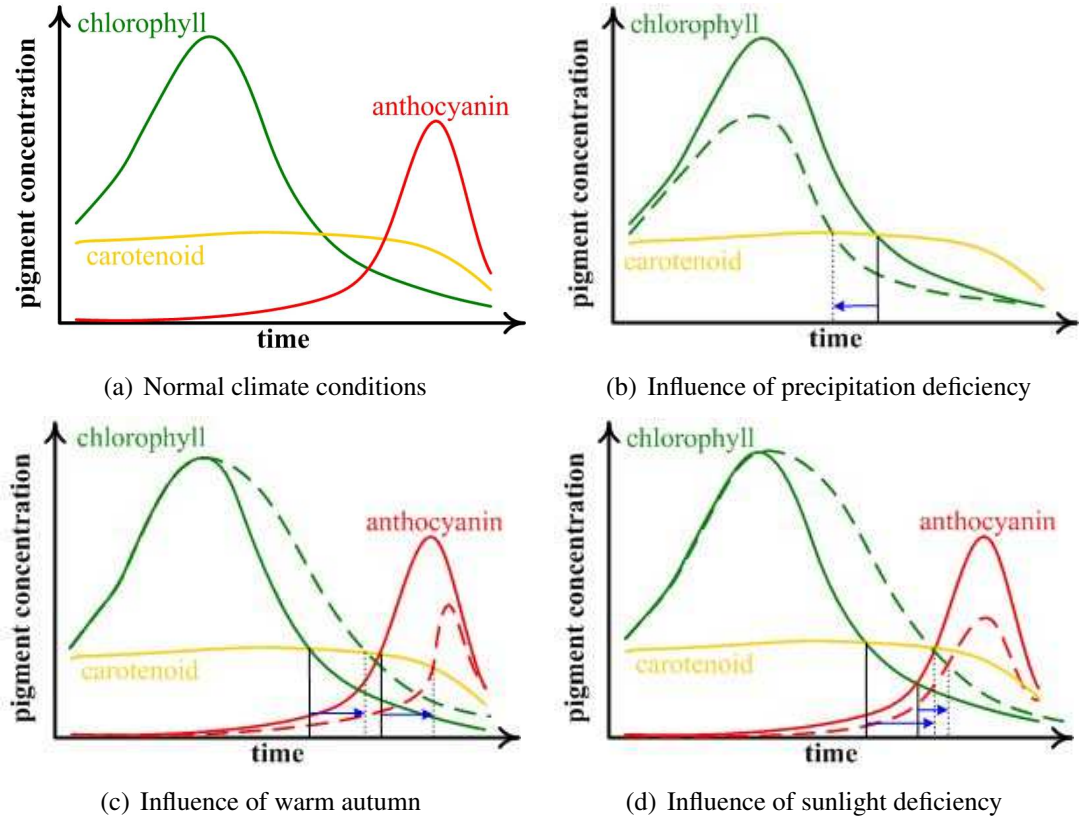
k is a constant. X is the value of influence factor, which is the temperature at time t in our system. X_c is the X value corresponding to half maximum response. n is a positive integer. The larger n is, the more obvious the threshold effect is. In our system, we let $n = 5$. The response curve is shown in Figure 4.4(b).

Concentration of carotene is generally considered as constant. All the before-mentioned models use empirical parameters obtained from previous observation data.

4.3.2 Climate Influence Simulation

A very accurate simulation requires massive climate data and chemical measurement data of the three main pigments, which are difficult to acquire. Therefore, in our system, based on limited observation data of several key stages, a standard process under normal climate conditions (Figure 4.5(a)) is generated by interpolation.

In a certain case, the climate conditions have some differences from the normal case. These difference will influent the leaf color changing timing. In Figure 4.5, we analyze three typical cases of abnormal climate. Figure 4.5(b) is precipitation deficiency. Precipitation affects



(a) Normal climate conditions

(b) Influence of precipitation deficiency

(c) Influence of warm autumn

(d) Influence of sunlight deficiency

Figure 4.5: Influence of climate to pigment concentration. Figure 4.5(a) shows the pigment concentration under normal environment conditions. The dashed lines in Figure 4.5(b) to 4.5(d) indicate the concentration changes under abnormal climate conditions. Red, green and yellow lines represent the absorbance of anthocyanin, chlorophyll and carotene. The arrows show the change of color transformation timing.

the production efficiency. Its deficiency causes a decrease of chlorophyll concentration, so the carotene is able to express its color earlier. The yellow stage starts earlier. Figure 4.5(c) is warm autumn. Temperature decides when the plant stops composition of chlorophyll, and starts to produce anthocyanin from sugar, so a warm autumn results in a lack of anthocyanin, and delays the red stage. Figure 4.5(d) is sunlight deficiency. Sunlight plays an important role in the photosynthesis, in which chlorophyll is consumed and energy is produced, so that deficient sunlight will retain chlorophyll and extend the green stage. In section 4.4 we show the corresponding rendering results of these climate conditions in Figure 4.8.

4.3.3 Leaf Texture Acquisition

According to botany knowledge, the color change process can be divided into several stages, from green to dark red. Though appearance differences exist among the leaves even in the same stage, usually several sample textures obtained from photos of real leaves are used to make an index of all the stages. This method is simple and fast, but it decreases the diversity of

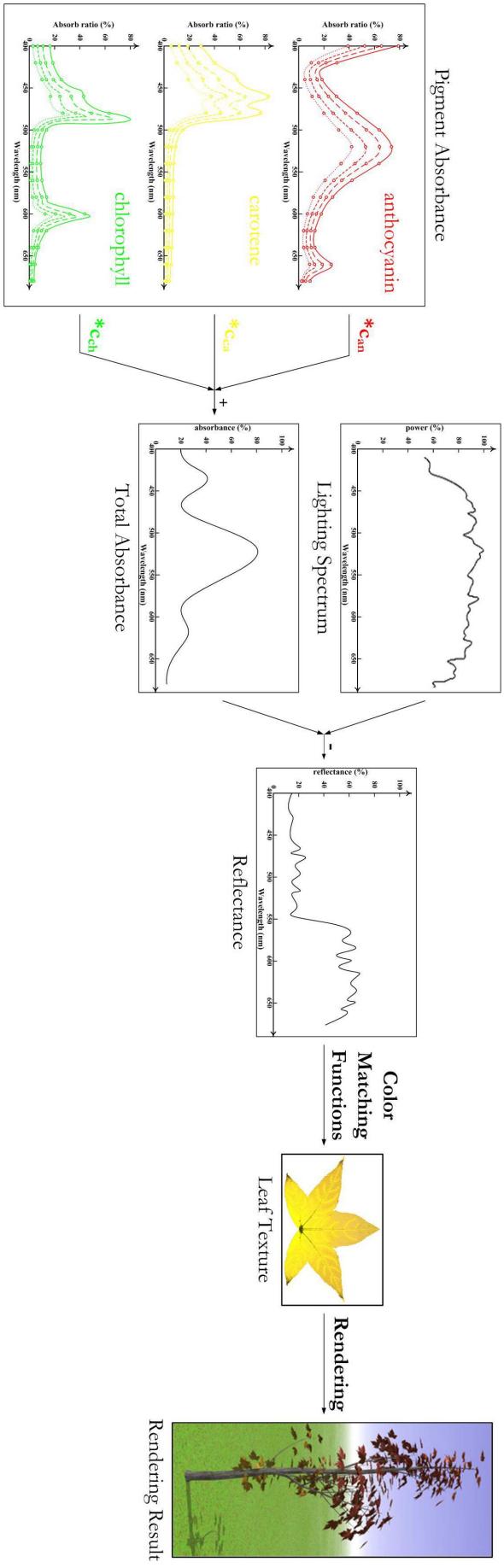


Figure 4.6: The illustration of leaf texture acquisition method.

leaves in the same stage, and results artifact sudden change between leaves of different stages. More accurate simulation can be made based on biological or mathematic models. Though this method requires more precomputation and storage space than the former one, it can achieve higher diversity. In this chapter, we design a joint method of measurement and algorithmic to take the advantage of the both.

The pipeline of our leaf texture acquisition method is shown in Figure 4.6. We use the concentration of chlorophyll c_{ch}, carotene c_{ca} and anthocyanin c_{an} as the input data to calculate the leaf color. Calculating the color of pigments mixture requires a subtractive color system rather than the common additive system. In our system, we directly subtract the pigment absorbance from lighting in the spectrum domain, and then use color matching functions to transform the reflectance spectrum to the color space we want. The calculations in spectrum domain are treated as a linear combination. According to Beer's law [Mac81], the absorption spectrum of a pigment approximately follows a linear relationship with its concentration: $A_c(\lambda) \approx c \cdot A(\lambda)$. The basic absorption A_0 and transmission T_0 of leaf tissue, which vary among different plant species, are approximately by measure data from several typical leaf samples [DCP98]. Thus the reflectance of the leaf is

$$R(\lambda) = L(\lambda) - T_0(\lambda) - A_0(\lambda) - \sum_{p \in an,ca,ch} c_p A_p(\lambda). \tag{4.9}$$

And then, the leaf color can be calculated by Equation 4.1 and 4.2. A basic map containing spacial variance information such as the venation pattern is used to improve the final results.

4.4 Results

We use our system to simulate the seasonal color variance of maple trees. On a PC with P4 3.2GHz CPU, 1GB RAM and a Geforce 7800 GT graphics card, the computation time for rendering a maple tree with 150 leaves is around 25 seconds. The performance is generally determined by the amount of leaves and observation step of climate data. Figure 4.2 shows the rendering result of our system. Figure 4.7 shows the leaf color change of a maple tree in the autumn as time passes.

Figure 4.8 shows the influence of different climate conditions. The four figures are the same scene in the same season. The scene consists not only maples, but also pine, which does not have the seasonal color change, and willow, which only change from green to yellow. Figure 4.8(a) is the simulation result with normal climate conditions. The maples start to turn to red, and the willow is yellow. Figure 4.8(b) to 4.8(d) show the simulation results with precipitation deficiency, warm autumn, and sunlight deficiency. In Figure 4.8(b), precipitation deficiency causes a decrease of chlorophyll concentration, so the green stage ends earlier. More leaves of the maples are yellow and red. In Figure 4.8(c), warm autumn postpones the composition of anthocyanin, hence delays the red stage. In Figure 4.8(d), sunlight deficiency retains chlorophyll, so it inhibits the color express of carotene.

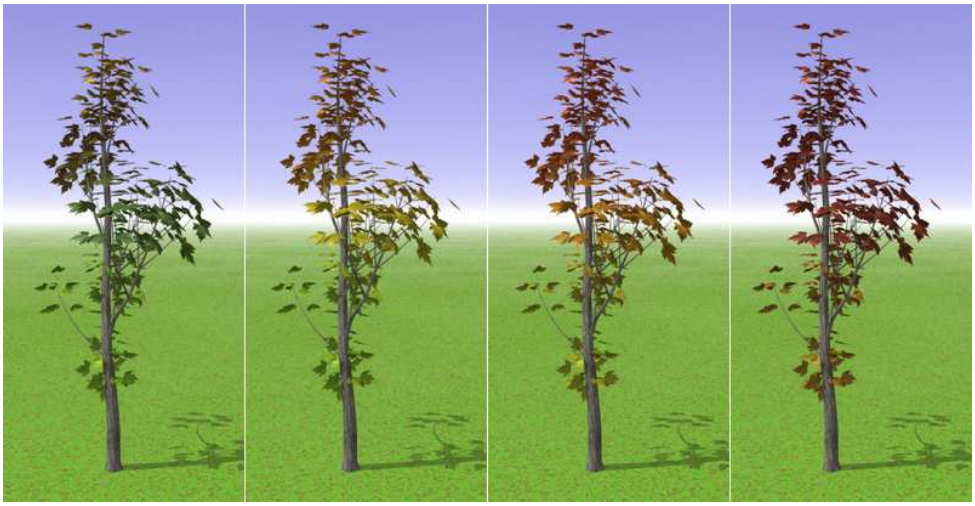


Figure 4.7: The sample result of our system.

(a) Normal climate conditions

(b) Influence of precipitation deficiency

(c) Influence of warmer autumn

(d) Influence of sunlight deficiency

Figure 4.8: Simulation results corresponding to the influences of climate conditions in Figure 4.5.

4.5 Conclusion

This chapter introduces a system to simulate the seasonal color change of maple leaves. Based on the Michaelis-Menten model and the threshold response model, concentration change precesses of three key pigments chlorophyll, carotene and anthocyanin are simulated. Parameters are estimated based on local environment conditions. Particular texture is generated for every leaf to achieve high diversity and natural appearance in the rendering result. Furthermore, given future climate data, Our system is able to predict the color transformation timing of maples. Influence of abnormal climate can also be simulated.

For future work, simulating the shape change of leaf during the growth is an interesting direction.

Chapter 5

Optimized Tile-Based Texture Synthesis

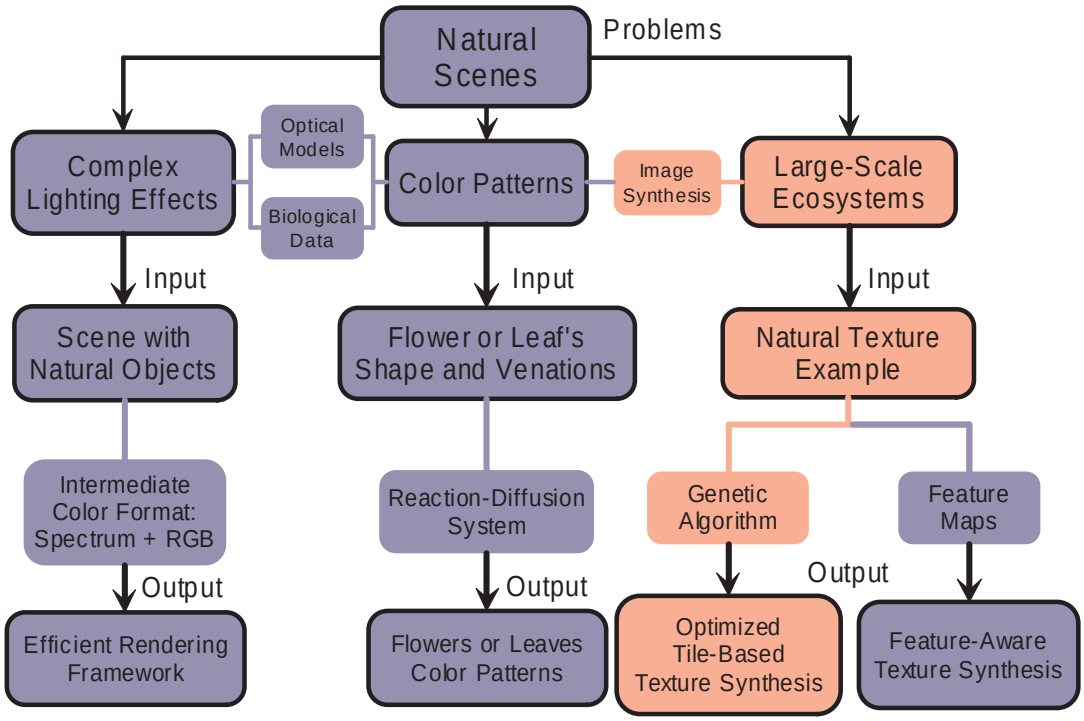


Figure 5.1: Optimized tile-based texture synthesis using *Genetic Algorithm*.

5.1 Introduction

Generating novel photo-realistic imagery from smaller examples has been widely recognized as significant in computer graphics. A wide number of applications require realistic textures to be synthesized for object decoration in virtual scenes. The global repeatability within texture images is essential to texture synthesis techniques. This inherent property makes it possible to express adequate texture information with limited portions.

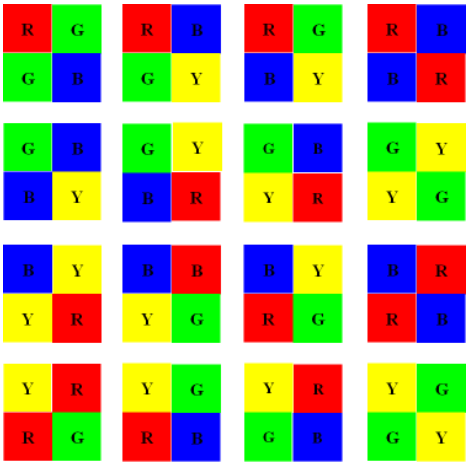
Nowadays local region-growing methods are popularly used in texture synthesis. These methods generate the texture by growing one pixel [Bon97, EL99, WL00, Ash01] or patch [EF01, KSE+03, NA03, WY04, LLH04, KEBK05] at a time with the constraint of maintaining coherence of neighboring pixels in the grown region. Such approaches always suffer the time-consuming neighborhood matching in the example and do not sufficiently meet real-time applications. On the other hand, some near real-time texture synthesis methods usually achieved low quality results for the lack of optimization in the pre-processing [ZG02, ZG04] or needed very complex pre-computation and data structures [LLX+01]. Recently efficient GPU-based texture synthesis techniques [LH05, LH06] have also been proposed, however they always demand a high performance graphics hardware.

An alternative approach is to pre-compute a set of small tiles and use these tiles to generate arbitrary size of non-periodic images at run time [CSHD03, Wei04, NWT+05, DSP05]. The tile-based method usually employs a set of *sample patches* which are extracted from the input example as texturing primitive. Then tiles are constructed by stitching sample patches together following some given rules. This technique requires only a small amount of memory and is very useful in many real-time applications, although sometimes achieving low quality results or dull patterns for the lack of optimization on the tile set.

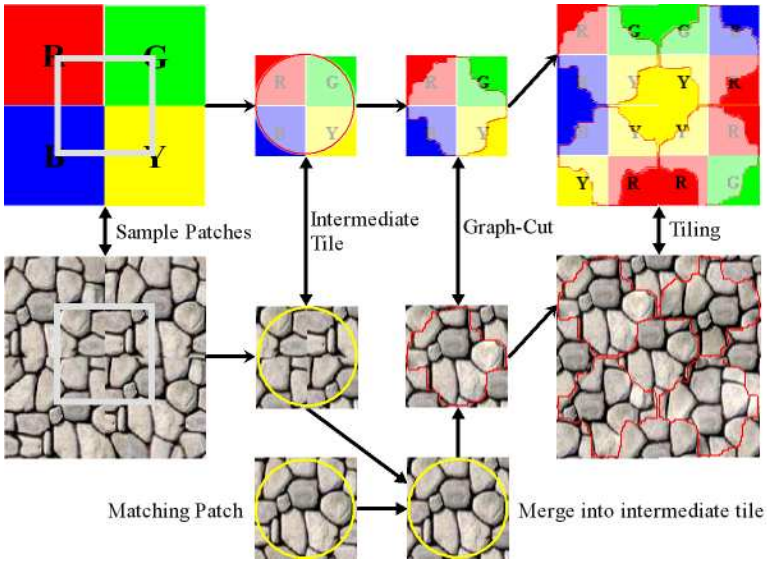
In this chapter, we present an approach for tile-based texture synthesis that is based on the optimization of tile set quality within a *Genetic Algorithm* (GA)-based framework. Our main contribution is to merge some locally defined optimization measures into a global *evaluation function* that can jointly optimize the quality of the entire tile set. This evaluation function balances the qualities among tiles and can be optimized by GA with reasonable computational cost. GA is an efficient and near global optimum search method [SP94, Koz95, LL02, PMJ96], which can automatically achieve and accumulate knowledge about the search space and adaptively control the search process to approach the global optimal solution.

Our approach starts with an 8-tiles ω-tile set used in [NWT+05] as the basic tile set (Figure 5.3(a)). An extendable rule will be given to directly derive new tile sets from existing ones. By employing more sample patches during the tile construction process, the new ω-tile sets introduced in this chapter can generate patterns more abundant than the sets of similar sizes in both [NWT+05] and [CSHD03].

As shown in Figure 5.2(b), an ω-tile is constructed by finding a matching patch from the input, and merging it into the intermediate tile using graph-cut. The *intermediate tile* is the combination of four sample patch quarters. We propose a global optimization algorithm to search for a feasible set of sample patches. Intermediate tiles formed by these patches will satisfy both local and global optimal conditions. The local condition implies that each intermediate tile could



(a) Set of 16 ω-tiles in [NWT+05].



(b) Basic construction process of an ω-tile. The matching patch is merged into the intermediate tile by graph-cut.

Figure 5.2: The previous ω-tile set formation process.

find an adequately well *matching patch* (the texture patch picked from the input example to merge into the intermediate tile for erasing junctions) from the input, while the global condition means that the tile qualities are balanced according to their smallest matching errors. The two conditions are interpreted together as an *evaluation function*. This function is defined as the linear combination of the sum of the *matching errors* (the distance between the intermediate tile and the candidate matching patch) between intermediate tiles and their *closest matching patches* (the candidate matching patch with the smallest distance), and the standard variance of all the errors. Sample patches selection proceeds by optimizing this evaluation function using GA. We start searching with a group of sample patches extractions. The optimization procedure improves the quality of the entire group through successive iterations of the algorithm. The final solution is the best extraction in the group when GA terminates.

Finally, we have extended the tile-based texture synthesis approach to be suitable for multiple input examples. We implement an interactive system that allows for extracting objects from source images and merging them into some tiles. This permits the generation of "meaningful" images with more variety and perspective patterns, using the same tiling method as for texture synthesis.

5.2 Previous Work

Texture tiling. Cohen et. al [CSHD03] developed a stochastic algorithm to non-periodically tile the plane with a small set of Wang-tiles at run time. Wei [Wei04] extended this work with GPU to improve tile-based texture mapping. Ng et al. [NWT+05] presented another approach to generate a set of small texture tiles from an input example. These tiles could also be tiled together to synthesize large textures. Our technique uses their ω-tiles as the tile set pattern. Figure 5.2(a) shows a typical ω-tile set in [NWT+05]. All these approaches require a set of sample patches extracted from the input example to generate the primary tile patterns, so the quality of their texture tiling results is not stable due to the uncertainty of the sample patches.

The previous ω-tile construction process is shown in Figure 5.2(b). The approach starts with randomly obtaining a set $\mathbb{T}$ of square sample patches from the input example. Each patch is assigned to be related with one color square. Then an intermediate tile can be cut from the middle of the texture block. Different intermediate tiles are generated according to the different arrangements of the sample patches (corresponding to the arrangements of the color squares). The cross junctions where four quarters meet is erased by picking a matching patch from the input and merging it into the intermediate tile with graph-cut. The matching patch is the same size as the intermediate tile. A circle is employed to constrain the boundary of the cutting curve in order to maintain the continuity of the patterns between matching sides in the tiling image.

Genetic Algorithm. GA is a stochastic search method for solving optimization problems. It is so named because the scheme is based on the mechanics of natural selection and natural genetics. Research interests in heuristic search algorithms with underpinnings in natural and physical processes began in the 1970s, when Holland [Hol75] proposed GA. GA generates a sequence of populations using a selection mechanism, and applies crossover and mutation as search mechanisms. It has demonstrated considerable success in providing good solutions to many complex optimization problems [LL02, SLC+05], such as capital budgeting, vehicle routing problem, critical path problem, parallel machine scheduling, redundancy optimization, open inventory network etc. The advantage of GA is due to its ability to obtain a global optimal solution fairly in a multidimensional search landscape, which has several locally optimal solutions as well. Hence it is a powerful technique that can be applied in texture synthesis.

Applications of Tilings. An early application of tilings in computer graphics was described in [Sta97], which filled the tiles with procedurally generated waves and caustics. Glassner presented various interesting tiling patterns [Gla98]: regular, semi-regular, and aperiodic. Triangular tiles were used in [NC99] to generate a non-periodic texture over a mesh while the approach in [FL05] reformulated the mechanism of Wang-tiles for texture tiling on arbitrary

topological surfaces. Decaudin and Neyret applied tiling to generate and render forest scenes in real-time [DN04]. Chenney employed Wang-tiles to contain and generate animated flow patterns [Che04]. In [LD05], a procedural object distribution function was presented. This texture basis function distributed procedurally generated objects over a procedurally generated texture. Kopf et al. introduced a recursive technique for tile-based generation of blue noise point set [KCODL06]. The use of Wang-tiles in their work enabled the generation of infinite non-periodic tilings. Our approach can be a good alternative for these applications because it supplies equal capability with Wang-tiles

Image Composition. Image composition is to extract an object from a source image and then paste it onto a target image naturally using a matting or feathering operation. It can also be done by stitching up multiple image slices from a single image or multiple ones. Graph-cut textures [KSE+03], for instance, picked a set of source images and combine them to form a single output image by finding the best seams with graph-cut technique. Pérez et al. used a guided interpolation based on solving poisson equations to mix two images with a user-specified boundary [PGB03]. Method to optimize the boundary condition for poisson image editing has also been presented [JSTS06], through an objective function and a blended guidance field. We use a similar method as [PGB03] and [JSTS06] to fertilize the patterns of the tiles, in other words, embed meanings in them. To efficiently extract a foreground object from a complex scene, we use the "GrabCut" tool described in [RKB04], which can segment an image by energy minimization technique.

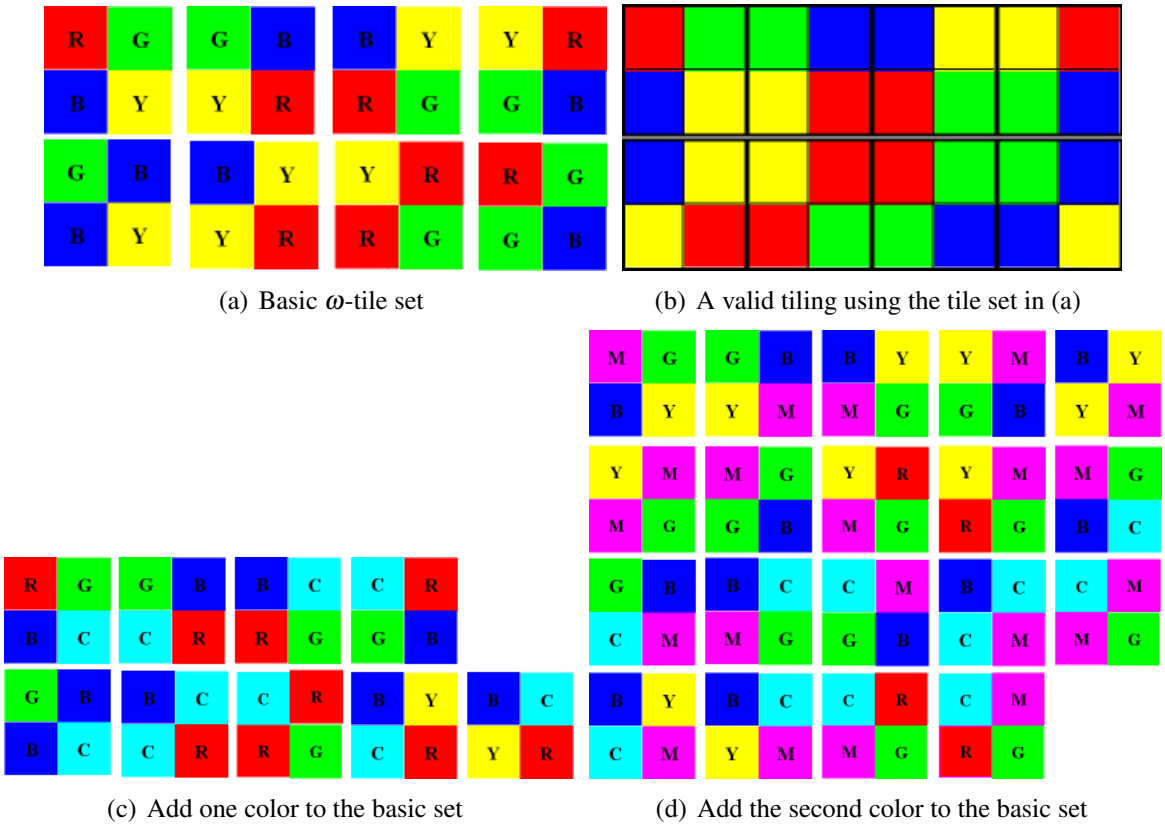
5.3 Tile Set Formation

We choose the size-8 ω-tile set in [NWT+05] as the basic tile set, as shown in Figure 5.3(a). Here we use $\mathbb{B}$ to denote it. A set of squares $\mathbb{P} = \{R, G, B, Y\}$ are used to compose the color blocks.

5.3.1 Tile Patterns Analysis

The tiling process using ω-tiles is carried out in scan-line order. Once the tile in the left-top corner is fixed, the rest tiles in the tiling are laid one by one from left to right, and top to bottom, consistent with the square colors of their neighbors. A valid tiling using $\mathbb{B}$ is shown in Figure 5.3(b). The basic tile set composed of only 8 tiles will draw undesirable repetitive patterns in a large synthesis image. Two methods are proposed in [NWT+05] to overcome this artifact. One way is to pick two patches from the input for each original tile. It can only partly eliminate the repetitive patterns because we should not neglect the repetitive patterns caused by the central parts of the sample patches. As shown in the rightmost column of Figure 5.2(b), the central pattern of the sample patch still possesses an important role in the tiling. To solve this problem, we develop an effective method to directly increase the number of the sample patches, without losing the characteristics of the whole tile set. The other way used in [NWT+05] is to

increase the tiles quantity by using more arrangements of the sample patches. This method is also suitable for our approach.



(a) Basic ω-tile set

(b) A valid tiling using the tile set in (a)

(c) Add one color to the basic set

(d) Add the second color to the basic set

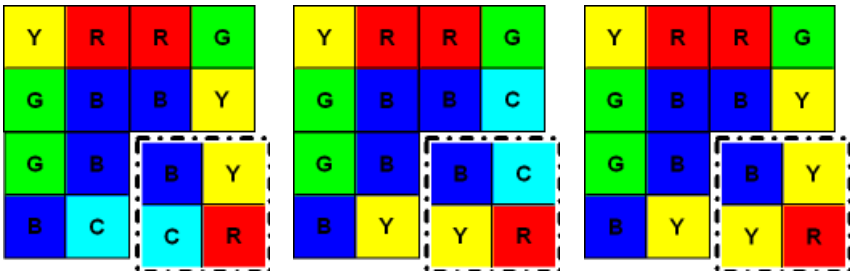
Figure 5.3: Tile set formation from basic size-8 ω-tile set

5.3.2 Increase Sample Patches

We add new patterns into the ω-tile set by enlarging the size of the sample patches set $\mathbb{T}$. This operation is proceeded directly on $\mathbb{B}$. We derive new tiles with the following steps:

1. Randomly pick a square from set $\mathbb{P}$ as the "reference" square (or reference color). Without losing generality, we choose the yellow square as the reference square.
2. Add a new square to the set $\mathbb{P}$. Here we use C (Cyan) to represent it. Then the new square set is enlarged to be $\mathbb{P}_1 = \{R, G, B, Y, C\}$.
3. Generate new tiles by replacing the yellow squares with cyan squares in $\mathbb{B}$.
4. Add another two tiles by replacing only one yellow square with cyan in the tile $\langle B, Y, Y, R \rangle$.

The new tiles formed by the above method is shown in Figure 5.3(c). The last two tiles are the additional tiles generated by step 4. The new tiles together with $\mathbb{B}$ forms the new ω-tile set $\mathbb{N}_1$. Despite of the two "additional" tiles, the other new tiles plus the tile $\langle R, G, G, B \rangle$ can be considered as a copy of the basic set $\mathbb{B}$. It can be used independently to tile arbitrary size area.



(a) (b) (c)

Figure 5.4: The cases why the two additional tiles in Figure 5.3(c) are required.

We use $\mathbb{N}_1'$ to indicate the tile set which possesses all the tiles in $\mathbb{N}_1$ except the two additional tiles. During the tiling process, if only $\mathbb{N}_1'$ is used, there will be no problem if no yellow square and cyan square need to be matched at the same time when placing a tile. It can be treated as a pattern duplication with the basic tile set. The cases shown in Figure 5.4(a) and Figure 5.4(b) appear where yellow square and cyan squares are needed to be matched simultaneously, there will be no matching tile in $\mathbb{N}_1'$. Now the two additional tiles are necessary to complete the tiling process. In fact these two patterns are just the extensions of the pattern in Figure 5.4(c), which is one of the tiling patterns using the basic set $\mathbb{B}$. Therefore, the new tile set $\mathbb{N}_1$ can be used to tile any large area and maintains all the properties of the ω-tile set which are stated in [NWT+05].

We can continue to increase the number of sample patches into six by following the same rule. Based on the tile set $\mathbb{N}_1$, we pick red as the reference color, and generate new tiles by replacing the red squares with magenta. The tiles with double red squares also need to be derived into two additional tiles by replacing only one square at a time. The new tiles are shown in Figure 5.3(d). Packed with set $\mathbb{N}_1$, we get another new ω-tile set $\mathbb{N}_2$. Here the square set is enlarged to $\mathbb{P}_2 = \{R, G, B, Y, C, M\}$. Obviously, we can keep on increasing the sample patches in $\mathbb{P}_2$ by following this rule if there is enough memory during the rendering process. Thus in the tiling process, this method can effectively eliminate the repetitive patterns caused by both the tiles and the sample patches themselves. Usually the set $\mathbb{N}_2$ which contains 38 tiles is enough to achieve a less-repetitive tiling result.

5.4 Tile Construction

The tiling quality is directly decided by the quality of the final tile set. As illustrated in Figure 5.2(b), the graph-cut result performed between the intermediate tile and the matching patch will directly affect the tile quality. Thus, special care should be taken when picking sample patches from the input, so that the intermediate tiles can get feasible matching patches for graph-cut operations. The random picking method used in [CSHD03] and [NWT+05] is not optimal for this requirement. The intermediate tile formed by randomly chosen sample patches can not insure the finding of a good matching patch from the example. On the other hand, because of the great quantity of pixels in an image, it is almost impossible to use brute-force searching method to find the best sample patches extraction. For example, for an input example of size

128×128, if we set the tile size to be 80×80, there will be $(48 \times 48)^n$ choices for the sample patches extraction (n is the number of sample patches). It is an unacceptable computational requirement for a normal PC in reasonable time. Our approach can efficiently and accurately solve this problem.

5.4.1 Algorithm Overview

Our optimized sample patch selection algorithm is essentially based on GA. GA starts with an initial set of randomly generated chromosomes called a population where each chromosome encodes a solution to the optimization problem. All chromosomes are evaluated by an evaluation function which is some measure of fitness. A selection process based on the fitness values will form a new population. The cycle from one population to the next is called a generation. In each new generation, all chromosomes will be updated by the crossover and mutation operations. Then the selection process selects chromosomes to form a new population. After performing a given number of cycles, or other termination criteria is satisfied, we denote the best chromosome into a solution, which is regarded as the optimal solution of the optimization problem. We will follow this framework to develop our algorithm.

5.4.2 Optimized Sample Patches Selection

The original problem of sample patches selection is to search for n sample patches from the input example to fill the m tiles. For example for the ω-tile set $\mathbb{N}_2$, $\{n = 6, m = 38\}$. Then the intermediate tiles filled by the these sample patches can safely find feasible matching patches under the given rules for junction flattening. Here we use GA to optimally find the n sample patches:

Initialization: To ensure an optimal solution be obtained in a reasonable runtime, an initial population consists of a considerable amount of chromosomes (n sample patches) is necessary. To start the algorithm, an integer *pop_size* is defined as the number of chromosomes. From the input example, *pop_size* chromosomes is selected randomly, denoted by $\{A_1, A_2, ..., A_{pop\_size}\}$. Every chromosome contains n sample patches (n genes) selected from the example image:

$$A_i = (g_i^1, g_i^2, ...g_i^n), i = 1, 2, ..., pop\_size$$

We encode each gene as the coordinate of the sample patch in the input example.

Evaluation: In GA, the selection of chromosomes to reproduce is determined by a probability assigned to each chromosome A_i. This probability is proportional to its fitness relative to other chromosomes in the population, i.e. chromosomes with higher fitness will have more chance to produce offsprings by the selection process. In the context of sample patches selection, the fitness of a chromosome, is evaluated by an evaluation function, $E(A_i)$ which measures the performance of the sample patches derived from that chromosome. The evaluation function, in essence, computes the minimum matching error between the intermediate tile and the candidate matching patch. It involves searching for translations of the input image that match well with

the intermediate tile. Let $I(p)$ and $T(p)$ be the pixels at the position p in the input example and the intermediate tile, the evaluation function is defined as:

$$C_j(t) = \sum_{p \in S_t} |T(p) - I(p-t)|^2, (j = 1,2,...,m; t \in P_T) \quad (5.1)$$

$$D_j = \min\{C_j(t), t \in P_T\} \quad (5.2)$$

$$V_i = Variance(D_1, D_2, ..., D_m) \quad (5.3)$$

$$E(A_i) = \lambda \cdot \sum_{j=1}^{m} D_j + (1-\lambda) \cdot V_i \quad (5.4)$$

where $C_j(t)$ is the matching error at translation t of the jth tile. It defines the distance between the intermediate tile and the candidate matching patch. S_t is the portion of the translated input overlapping the tile, P_T is the set of valid translations (candidate matching patches) in the input, and D_j is the minimum matching error within all the translations. V_i is the variance of all the minimum errors, we add this factor in order to protect the global quality of the final tile set. It avoids that intermediate tiles with extremely high and extremely low matching errors appear together in the same set. So the evaluation function $E(A_i)$ is the linear combination of the minimum error sum and the variance. We set $\lambda = 0.8$ for all the experiments in this chapter. Note that our evaluation function is very similar to the energy function used in [KEBK05] for texture optimization, while here we use GA rather than Expectation Maximization (EM) to optimize it.

To obtain an optimal tile set, we need to determine the best sample patches that has the least overall matching error. Hence, the fitness function $F(A_i)$ for selection is defined as the inverse of the evaluation function, i.e. $F(A_i) = \frac{1}{E(A_i)}$. Based on the value of the fitness function for each chromosome, the population of chromosomes $\{A_1, A_2, ..., A_{pop\_size}\}$ is rearranged from high fitness to low fitness.

Selection: The selection process is basically a "spinning the roulette wheel" process. The roulette wheel is spun *pop_size* times and each time a chromosome from the rearranged population $\{A_1, A_2, ..., A_{pop\_size}\}$ is selected. As we have stated, the chromosome with higher fitness should have a higher probability to be selected. It is achieved by the following steps:

1. Define a ranking function for each chromosome

   $$eval(A_i) = \alpha(1-\alpha)^{(i-1)}, i = 1,2,...,pop\_size$$

   where $\alpha \in (0,1)$ is a predefined parameter.

2. Based on this ranking function, calculate the cumulative probability, q_i for each chromosome $A_i \cdot q_i$ is given by

   $$q_0 = 0, q_i = \sum_{k=1}^{i} eval(A_k), i = 1,2,...pop\_size$$

3. Generate a random real number γ in $(0, q_{pop\_size}]$.

4. Select the i^{th} chromosome A_i such that $q_{i-1} < \gamma \leq q_i$. Repeat step 3 and step 4 until *pop_size* copies of chromosomes are obtained. These *pop_size* copies of selected chromosomes $\{\hat{A}_1, \hat{A}_2, ..., \hat{A}_{pop\_size}\}$ are the mother chromosomes for the reproduction of the next generation.

Crossover: The crossover process will produce a new generation of population based on the set of mother chromosomes $\{\hat{A}_1, \hat{A}_2, ..., \hat{A}_{pop\_size}\}$ resulting from the selection process. We define $P_c \in [0,1]$ as the probability of crossover. Hence, the expected number of mother chromosomes that will undergo crossover is $P_c \cdot pop\_size$. To pick the parents for crossover, we perform the following action:

For $i = 1$ to *pop_size*

generate a random number γ;

if $\gamma < P_c$ put $\hat{A}_i$ in a parent list

else put $\hat{A}_i$ in a non-parent list

end.

We denote the parent list as $\{\tilde{A}_1, \tilde{A}_2, ...\}$ and the non-parent list as $\{\bar{A}_1, \bar{A}_2, ...\}$. If there are odd number of members in the parent list, the last member will be switch to the non-parent list. With an even number of member in the parent list, we group the members into pairs $\{(\tilde{A}_1, \tilde{A}_2), (\tilde{A}_3, \tilde{A}_4), ...\}$. A random natural number $c \in [2, n]$ is generated and applied to the crossover operation to each parent pair to produce two children given by:

$$X = (\tilde{g}_1^1, ..., \tilde{g}_1^{c-1}, \tilde{g}_2^c, ..., \tilde{g}_2^n)$$

$$Y = (\tilde{g}_2^1, ..., \tilde{g}_2^{c-1}, \tilde{g}_1^c, ..., \tilde{g}_1^n)$$

A new generation of population is produced by combining the children produced by the parent pairs and the non-parent chromosomes.

Mutation: In GA, to avoid the solution being bounded by a local optimum, a mutation process is applied to the chromosomes in the new generation. We define $P_m \in (0,1)$ as the probability of mutation. Hence, the expected number of chromosomes that will undergo mutation is $P_m \cdot pop\_size$. Similar to the picking of chromosomes for crossover, chromosomes are picked for mutation based on P_m. For each chromosome picked, denoted by $A_l = (g_l^1, g_l^2, ...g_l^n)$, two mutation position n_1 and n_2 is randomly chosen where $1 \leq n_1 < n_2 \leq n-1$. For $j = n_1$ to n_2, change g_l^j to a random patch from the input example which does not equal to any of the existing genes. After all the genes in and between the two mutation positions are changed, A_l changes to form a new mutated chromosome A_l'.

Termination: Two termination criteria are used. Either the process is executed to produce a fixed number N_g of generations, or no further improvement for the best solution is observed in N_o consecutive generations. The best solution among all these generations (the chromosome

Figure 5.5: Results for image texture synthesis. For each texture, the input is on the left and the output is on the right. All results are generated in real-time with the corresponding ω-tiles.

with the highest fitness) is chosen as the result when GA is terminated. Otherwise, the algorithm goes back to the evaluation step and begins the next iteration.

5.4.3 Working Flow

The GA-enhanced sample patches selection algorithm can effectively reduce the boundary artifacts caused by the mergence of the intermediate tile and its matching patch during the graph-cut process. The whole working flow of our optimized tile-based texture synthesis algorithm is: in pre-computation, first randomly initialize a considerable number of sample patches sets (as the chromosomes) from the input example, then use GA to find the best one. Finally graph-cut is employed for junction elimination. The run-time tiling process is the same as [NWT+05], we can synthesize arbitrary size of texture images in real-time.

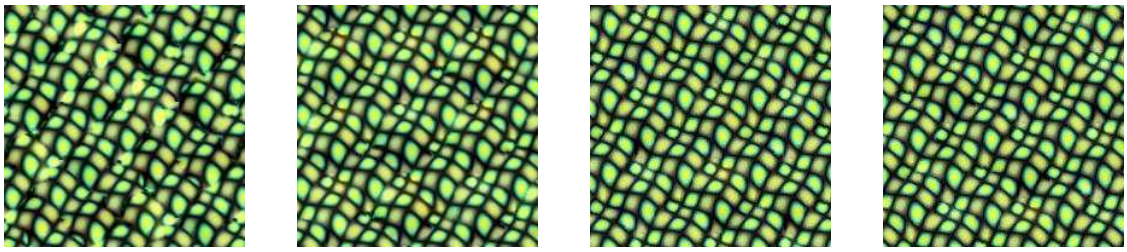


Figure 5.6: Results comparison of using different population sizes in GA. From left to right: $pop\_size = 10, 20, 40, 80$.

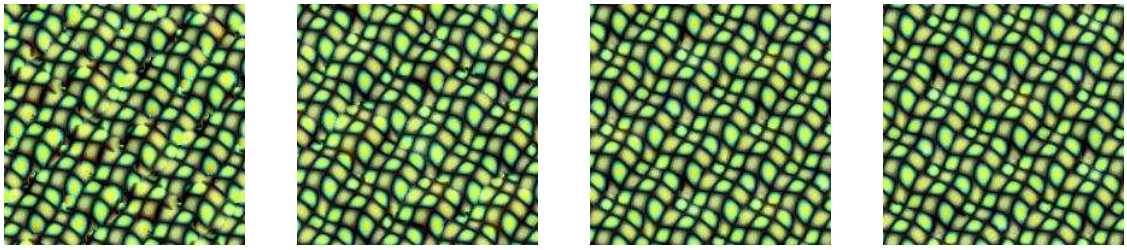


Figure 5.7: Results comparison of using different generation numbers in GA. From left to right: $N_g = 10, 30, 50, 80$.

5.5 Results and Discussion

Figure 5.5 shows some texture synthesis results using our algorithm. All the results are tiled in real-time with the ω-tile set $\mathbb{N}_2$ which contains 38 tiles (illustrated in Figure 5.3). Here we use the same parameters for GA with $\{pop\_size = 40, \alpha = 0.05, P_c = 0.3, P_m = 0.2, N_g = 50, N_o = 5\}$ in all the experiments of Figure 5.5. The most time-consuming procedure in our GA is the fitness evaluation of the chromosomes. In this procedure, we use approximate-nearest-neighbor search (ANN) [AMN+98] to accelerate the neighborhood matching operation between each intermediate tile and the input example. Note that the other techniques such as TSVQ [GG91, WL00], FFT [KDM02, SCA02] and mixture trees [DKO05] can also be employed here. Execution times of the GA-based sample patches selection process are listed in Table 5.1. All timing results are reported for our unoptimized C++ code on a Pentium 4 3.2GHz PC with 1GB RAM. The sequence of the example names is consistent with the image positions in Figure 5.5, from left to right and top to bottom. The timings indicate that using GA is an efficient way to select feasible sample patches.

The most important parameters in GA are the population size *pop_size* and the generation number N_g. In Figure 5.6 and Figure 5.7 we show comparisons of using different population sizes and different generation numbers in GA. The other parameters are set to be the same as Figure 5.5. The input examples are the same as the input texture in the first row of Figure 5.9. Normally the setting of $\{pop\_size = 40, N_g = 50\}$ is enough for most synthesis.

We can generate the distance function defined in Equation 5.1 to incorporate other characteristics of the texture besides color. For example, in order to use image gradients as an additional

Table 5.1: Sample patches selection timings for the examples in Figure 5.5.

Example	Example size	Tile size	GA
Beans	128×128	80×80	25 sec.
Yellow leaves	128×128	80×80	34 sec.
Tree barks	128×128	80×80	28 sec.
Caustics	128×128	80×80	24 sec.
Stones	128×128	80×80	35 sec.
Bread	108×99	70×70	18 sec.
Grape	144×144	100×100	43 sec.
Grass	128×128	80×80	26 sec.
Fabric	128×128	80×80	24 sec.
Bricks	128×128	80×80	37 sec.
Eggs	128×102	80×80	23 sec.

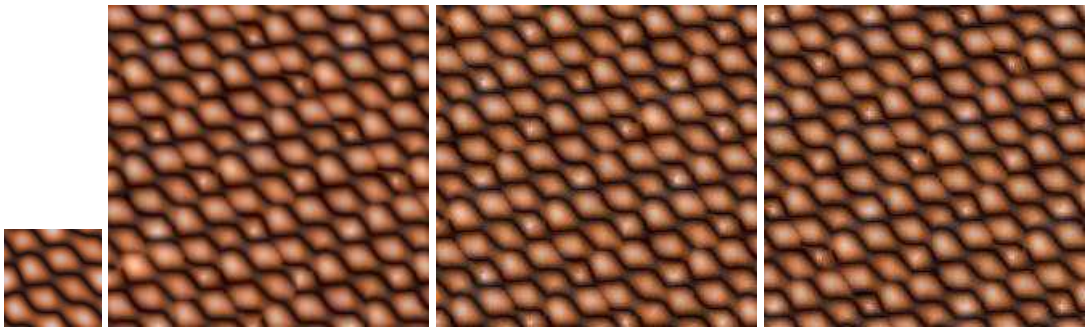


Figure 5.8: Results comparison when using different distance metrics. From left to right: the input example, result using Equation 5.1, 5.5 and 5.6.

similarity metric, we could define the distance as

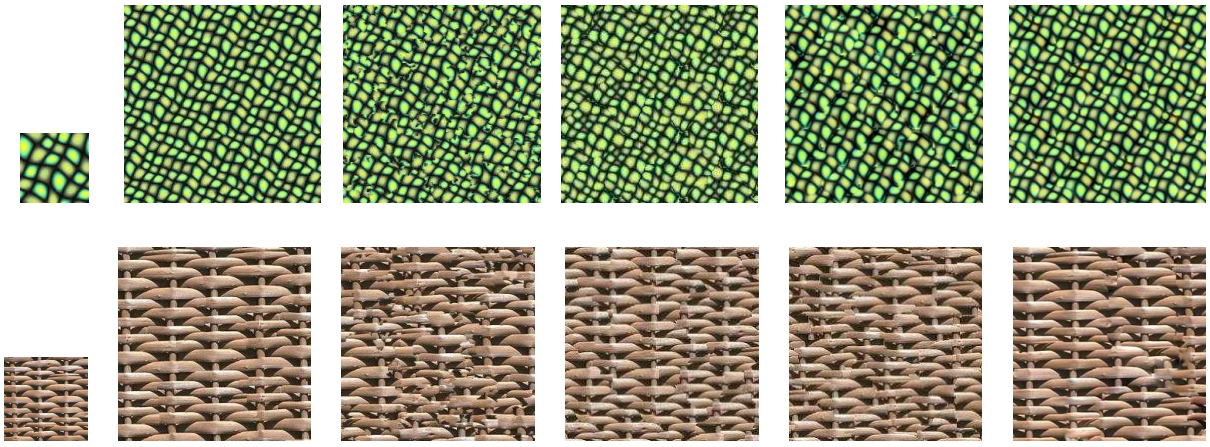
$$C(t) = \sum_{p \in S_t} \frac{|T(p) - I(p-t)|^2}{\mu(\|\nabla T\| + \|\nabla I\|)} \tag{5.5}$$

or

$$C(t) = \sum_{p \in S_t} |T(p) - I(p-t)|^2 + \mu \sum_{p \in S_t} |\nabla T - \nabla I|^2 \tag{5.6}$$

where $\nabla = [\frac{\partial .}{\partial x}, \frac{\partial .}{\partial y}]$ is the gradient operator and μ is a relative weighting coefficient ($\mu = 10$ in our experiments). Figure 5.8 shows the synthesis results using different distance metrics. Even though we have experimented with color and gradient, one could use other distance metric which measures some property of the texture patch. For most textures, we can simply use the color as the distance metric, as all the experiments in Figure 5.5 do.

We compare our results with other techniques in Figure 5.9. We can see that the qualities of our results are comparable with the off-line graph-cut method [KSE+03] (even though, their



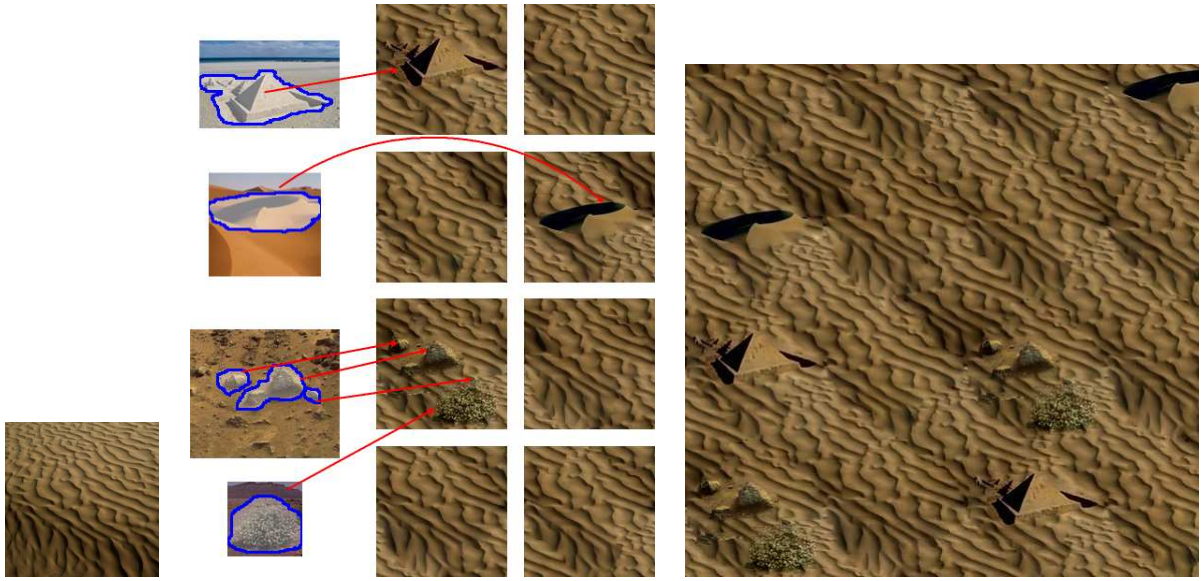
(a) INPUT (b) GRAPH-CUT (c) JUMP MAP (d) WANG-TILES (e) ω-TILES (f) OURS

Figure 5.9: Comparison of texture synthesis results with various other techniques. Results for other techniques are obtained from their web pages.

results outperform ours when synthesizing some structural textures) while better than the other CPU-based real-time techniques. The results in Figure 5.5 and Figure 5.9 show that our method is a very good choice for textures without very clear structures, especially for natural textures [Ash01].

5.6 Meaning Embedment

In the previous section, the optimized technique is used to create arbitrary size texture images from an input example. Moreover, efficient generation and seamless connection of high quality ω-tiles has some other potential applications in computer graphics. One extension is to construct the tile set from multiple examples. Thus we can embed meanings in some tiles corresponding to the content of the input examples. We call this "meaningful image synthesis", or texture synthesis with meanings. As illustrated in Figure 5.10, a desert texture is selected as the background. This kind of textures can be frequently explored in many computer games. Without losing generality, we use the basic 8-tiles ω-tile set $\mathbb{B}$ as the bearer in our demos so that more "meaningful" patterns can be shown in a normal size tiling. To embed meanings in some tiles, we first use GrabCut [RKB04] to produce the rough boundaries of the interested objects from the "meaning" (source) images, then the algorithm in [JSTS06] is proceeded to find the optimized boundaries. Finally we simply use poisson image editing to compose the objects with the specific tiles, as shown in Figure 5.10(b). In Figure 5.10(c), we can see a meaningful image which illustrates a desert with pyramids, stones, desert plants and sand dunes. It is generated in real-time with the tiles in Figure 5.10(b) and will be more vivid than a simple bare desert when it appears in a computer game. We can choose the ω-tile set with more tiles like $\mathbb{N}_1$ or $\mathbb{N}_2$ to construct meaningful tiles when synthesizing large size images, especially for certain applications where continuous patterns are required. This can efficiently avoid the repeat of the same patterns. Figure 5.11 shows another result of meaningful image synthesis. It is a busy water



(a) (b) (c)

Figure 5.10: A desert with pyramids, stones, desert plants and sand dunes. (a) Background example texture. (b) Embed meanings in some tiles with poisson image editing. The blue curves are the rough boundaries of the "meaning" images. (c)Meaningful image synthesis.

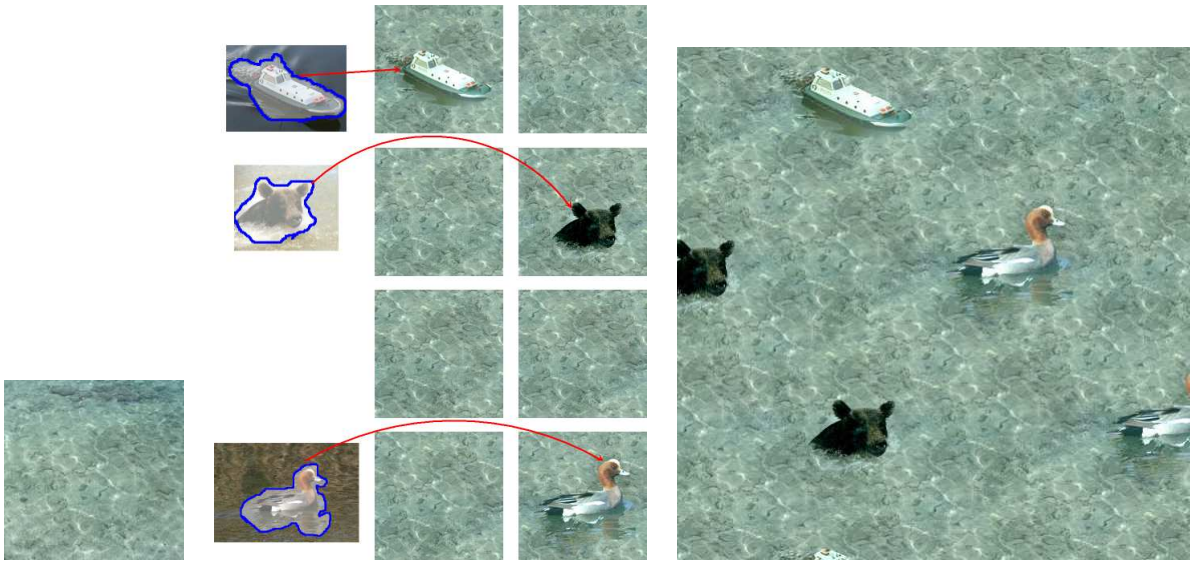
surface with bears, water birds and boat models.

5.7 Conclusion

We have presented a novel optimization-based technique for tile-based texture synthesis. Our results for both texture synthesis and image tiling are comparable to state-of-the-arts. We define a pattern repetitive principle that allows us to derive new ω-tile sets from the existing one. An optimized sample patches selection algorithm based on GA is used to improve the quality of the whole tile set. This framework is also fit for quality improvement of Wang-tile based texture synthesis [CSHD03]. Our technique can be nicely applied in the environment where real-time texture synthesis is needed, such as 3D games and real-time virtual reality systems, while the local region-growing methods such as image quilting, graph-cut and texture optimization are not applicable (need seconds or minutes to generate an image).

We also extend our technique to perform meaningful image synthesis in real-time by adding more examples to the tile construction step. It is very difficult for the local region-growing texture synthesis methods. This approach is very useful in many fields such as interactive decorative pictures design and land map design of computer games.

A limitation of our technique is that because it tries to erase the junctions in the intermediate tiles by a single patch from the input example, it is always constrained by the patterns of the intermediate tiles. It is manifested as relatively low qualities when synthesizing some structural textures, for example the eggs texture in Figure 5.5.



(a) (b) (c)

Figure 5.11: Water surface with bears, water birds and boat models. (a) Background example texture. (b) Meaning embedment. (c)Meaningful image synthesis.

For future work, we wish to extend our tile-based synthesis technique to handle image or geometric textures on 3D models. Another potential direction is to experiment with other local region-growing texture synthesis methods, such as texture optimization [KEBK05] and fractional Fourier texture masks [NMMK05], in the tile construction step to improve the synthesizing quality of structural textures.

Chapter 6

Feature-Aware Texture Analysis and Synthesis

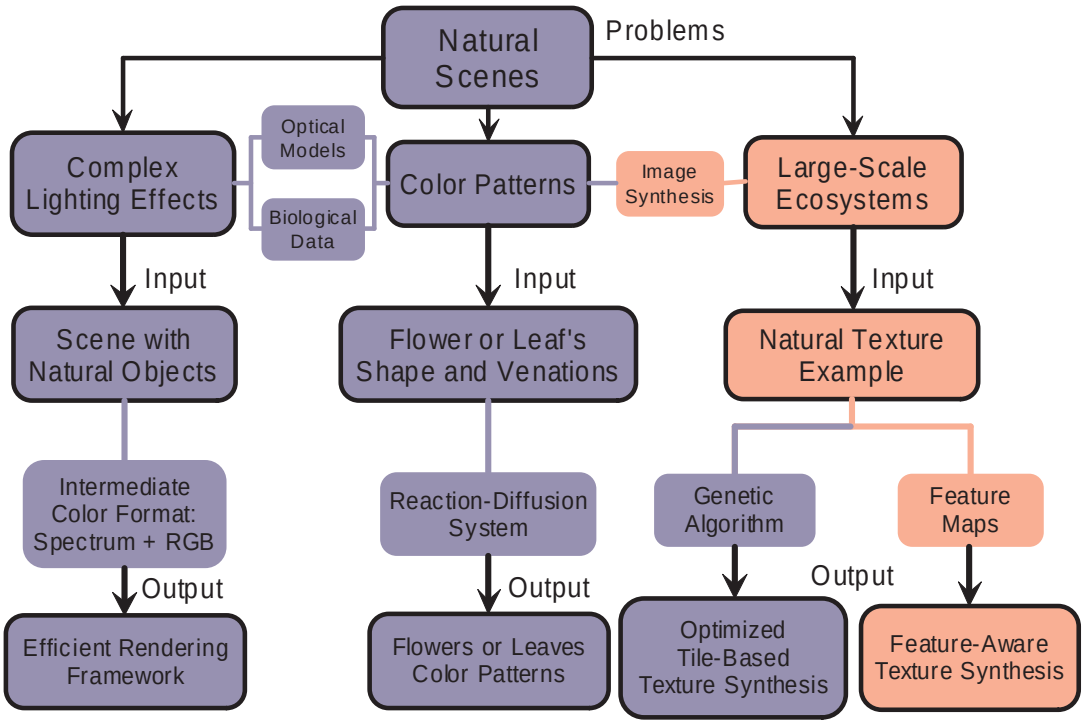


Figure 6.1: Feature-Aware Texture Synthesis.

6.1 Introduction

Texture synthesis is defined in [Ash01] as "a texture synthesis method starts from a sample image and attempts to produce a texture with a visual appearance similar to that sample."

The visual appearance of an example can be analysed by the *local* continuity of the texels(TEXture ELement), and *global* perceptual features such as texel scale variation. These features are due to environmental changes, e.g. perspective view-point, luminance, object distributions and geometry of the underlying surface.

Traditional approaches analyze local properties of a given sample and create visually similar images by comparing local neighborhoods. Therefore, they could nicely preserve the *local* continuities of the texels. Nevertheless, these approaches are not sensitive to the global perceptual features and hence limited to isotropic samples.

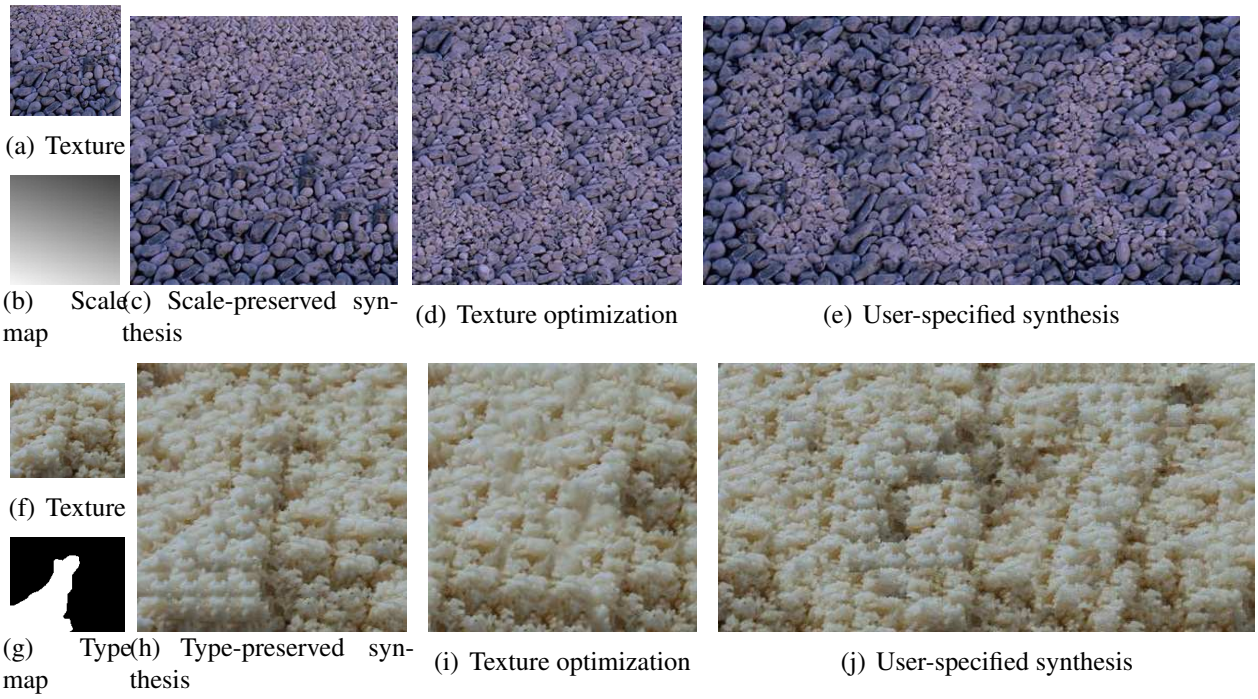
The goal of this chapter is to provide a framework for perceptual-feature-aware texture synthesis. The underlying idea is to consider the perceptual-featured textures (PFT) as the consequences of environmental changes to a general isotropic texture (as shown on the right). PFTs can depart from isotropic textures along different axes of appearance, thus could have (1) scale variations between texture elements (Figure 6.2(a)); (2) different types of textures or interest areas (Figure 6.2(f)); or (3) discontinuity of spatial layout in color intensity due to the environmental conditions (Figure 6.9(a)). Note that there are still some other possible features in a texture like direction and geometric distortion, while we only explore the above three aspects here.

This chapter presents several interactive techniques to define and extract these features from isotropy using a multi-modal (scale, type, color), multi-dimensional mapping, which we called a *feature field*. Each feature is quantized into a *feature map* masked by color values. Multiple maps could be integrated together as the constraint for output synthesis. Compared with the deformation field used by Liu et al. [LLH04] for manipulating near-regular textures, we extend their model in three aspects: (1) **Multi-modal feature fields**: extending the concept of texture deformation to include scale and type as well as lighting and geometry so that we can preserve the global appearance of PFT in output images (Figure 6.1); (2) **Feature field extraction**: to capture and adjust the corresponding features from the input sample with simple user-assisted interactions, we introduce techniques for more intuitive control; (3) **Feature map manipulation**: actively controlling the feature distributions of the output feature maps so that the appearance of the output texture can vary according to users' specifications.

6.1.1 Contributions

The principal contributions of this chapter are the two major components involved in our approach:

1. The analysis of PFT, where we explain how to define, extract and quantize different types of feature fields that associate global-featured textures to their normal counterparts (Sec-



(a) Texture (b) Scale map (c) Scale-preserved synthesis (d) Texture optimization (e) User-specified synthesis

(f) Texture (g) Type map (h) Type-preserved synthesis (i) Texture optimization (j) User-specified synthesis

Figure 6.2: Feature-aware texture synthesis. Our results (c) and (h) are synthesized with the constraint of feature maps. While the results (d) and (i) from texture optimization are not satisfactory.

tion 6.3, Section 6.4 and Section 6.5). User assistance is needed to initiate the process or edit the automatic-computed feature maps.

2. The synthesis and manipulation of PFT through multi-dimensional feature fields (Section 6.6).

These two components are connected by the idea of feature field duality: treating a feature field both as a function that acts on a texture and as a map that is acted upon. Texture synthesis results and further applications on image completion will be presented in Section 6.7.

6.2 Previous Work

A number of work has been presented on synthesizing 2D textures from input samples. Local region-growing techniques generate textures one pixel or one patch at a time. Pixel-based synthesis algorithms [Bon97, EL99, WL00, Ash01, HJO+01] grow an output texture pixel by pixel, normally using spatial neighborhood compare to match across different frequency bands. These approaches are fit for stochastic textures, but usually fail on textures with more coherent structures. Patch-based methods [LLX+01, EF01, KSE+03, WY04, NMMK05] copy selected source regions into the output instead of single pixels. They are generally more suitable for synthesizing structural textures. Some intermediate techniques [NA03, KEBK05] between pixel and patch based methods have also been presented, which somewhat combine the advantages

of both. Recently, efficient GPU-based texture synthesis techniques [LH05, LH06] have also been proposed. Another category is to precompute correlative tiles [CSHD03, Wei04, DSP05, NWT+05, DZP07] or similarity sets [TZL+02, ZG04] for runtime references. They usually achieve real-time performance while sacrifice some result qualities. Above techniques always assume the input samples as isotropic (while most of them are not strictly match this demand), so some global features of the inputs could be lost if no special treatment is employed during the synthesis process, as shown in Figure 6.2(d) and Figure 6.2(i) (generated by the texture optimization [KEBK05] method).

Existing work on analyzing and manipulating input sample has yielded impressive results for texture synthesis. Dischler et al. [DMLC02] decomposed textures into elementary components and recomposed similar textures by taking into account the previously computed arrangements. Liu et al. [LLH04] developed a multi-modal framework to define and capture deformation fields from near-regular textures. Using their formulation, simple parametric models could be constructed from input texture samples to purposefully manipulate the regularity of near-regular textures. The technique for synthesizing and editing structured textures has also been presented [DZ06]. They assimilated texture images to corresponding 2D geometric meshes, while synthesis was then based on the creation of new vertex/polygon distributions matching some arrangement map. However, both techniques will have difficulties in getting a feasible lattice or mesh when the texels are too complex to be segmented, or even there is no clear texel shape in the example, especially for some natural textures. Our method does not require the segmentation of texels, but use some very simple user assistance instead to evaluate the exemplar global features.

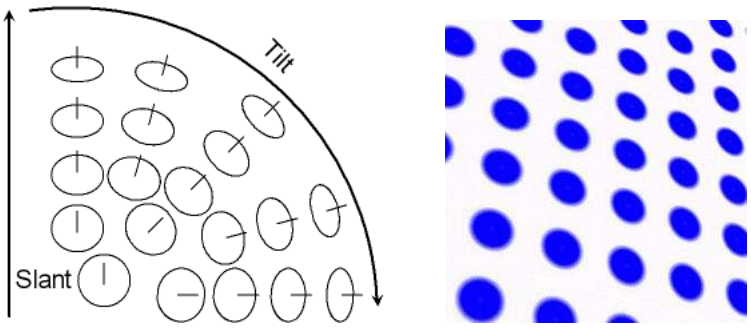
Similar work of our type-aware synthesis (Figure 6.2(h)) is described as texture transfer in Synthesizing Natural Textures [Ash01], Image Analogies [HJO+01] and Texture Quilting [EF01]. Like the texture replacement approach in [LLH04], our method treats the type differences in one texture as a feature field which can be separately or jointly employed as constraint mask rather than a simple data association model in intensity.

To preserve the texel scale variations in output images (Figure 6.2(c)), we need to extract the scale field form the example. This approach is concurrent with the research work in shape-from-texture. A wide variety of sophisticated shape-from-texture algorithms exist for recovering the shape and orientation of a surface through measuring the distortion of the texels on it [LG93, RH00, PLB01, CM02a, WF06]. These methods usually treat shape-from-texture as a statistical estimation problem and measure relative metric changes between the surface and the image plane. The results are 3D coordinates of the surface or slant and tilt angle for a plane. The texel scales are also evaluated during the recovery process.

6.3 Scale Feature

In our approach, we assume that the texture image captured from the real 3D surface is locally planar, such that the texture variations are only produced by the projective geometry. We do not deal with the geometric deformation as in [LLH04] for near-regular textures. The same method

could be employed if necessary. For visually plausible texture synthesis, we do not need to find a strictly consistent scales and transform angles. Instead we will evaluate some visual properties of the texture to compute the relative scale differences among the texture elements, and perform texture synthesis according to the constraint of the scale map.



(a) (b)

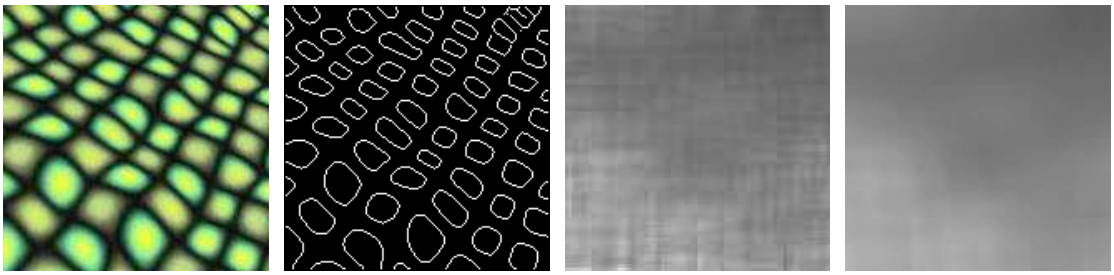
Figure 6.3: Illustration of texture scale variation. (a) A set of circular patches arranged on a sphere to illustrate the slant and tilt components of surface orientation. The line at the center of each patch is aligned in the direction of the surface normal. Note that the slant and tilt components from a spherical coordinate system, in which lines of latitude have constant slant, and lines of longitude have constant tilt. (b) A circle texture image with $slant = 45^\circ$ and $tilt = 30^\circ$.

The texel scale variation in a texture is directly affected by the surface orientation [NTN+06]. Here we represent surface orientation using a viewer centered spherical coordinate system that is parameterized in terms of *slant* (σ) and *tile* (τ). We consider the possible optical projections of a circular disk at varying orientations relative to the line of sight (see Figure 6.3(a)). The optical projection of a circle is always an ellipse. The slant of the circle in 3D space determines the aspect ratio of its projected ellipse, whereas the tilt component determines the orientation of the ellipse within the image plane. We could see the shape and scale variations of different circles in one image caused by the perspective projection in Figure 6.3(b). Note that in this chapter we use a 90°-rotated version for the slant definition in [NTN+06]. This kind of textures which possesses perspective deformation features are usually called perspective textures.

6.3.1 Scale Recovery from Feature Mask

For any perspective texture t_p, there exists an underlying mapping principle M_{sca} that illustrates the scale distortion of each point from an isotropic texture t_i. Clerc and Mallat [CM02a] gave the formulations for recovering the normals and projective angles from a textured image through the texture gradient equations, with these information the scales could be evaluated. But we find the following feature mask-based model works very well with many examples to calculate the scale maps.

Wu and Yu [WY04] introduced the notion of feature mask to help guide the synthesis process. Their idea can be simply utilized by our scheme. Given a binary feature mask of the input



(a) (b) (c) (d)

Figure 6.4: Extract scale map through binary feature mask. (a) Input texture. (b) Binary feature mask. (c) Original noisy scale map. (d) Final smoothed scale map.

texture t_p (Figure 6.4(b)), for each point p, we compute the quantity of the feature pixels in the mask within its user-assigned $n \times n$ neighborhoods ($n = 32$ for Figure 6.4(c)). We denote the point with minimum feature pixel number a_{min} as the largest scaled point, while the point holding maximum a_{max} feature pixels has the smallest scale. We consider the point with the medium number $a_{mid} = (a_{min} + a_{max})/2$ as the non-distorted point which has the scale of 1. Then the scale $S(p)$ of each point is calculated as

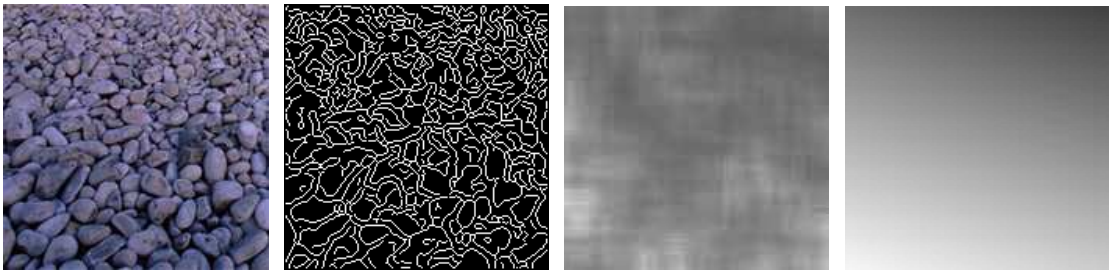
$$S(p) = a(p)/a_{mid} \tag{6.1}$$

where $a(p)$ is the number of feature pixels within the neighborhood of p. Now we can get a very noisy scale map through equation 6.1 (Figure 6.4(c)). The reason is that the neighborhoods of some points might not contain the proper quantity of feature pixels that can be mapped to its real scale. So we use a 13×13 Gaussian filter to smooth this map and the result can be seen in Figure 6.4(d). Usually the user could adjust the size of the Gaussian filter according to the noisy degree of the original map. Note that we visualize the scale map as a gray image. A gray value of 128 means the original scale ($S(p) = 1$), and the increase of white value means a magnification on the original texel scale.

6.3.2 Scale Recovery from Angles

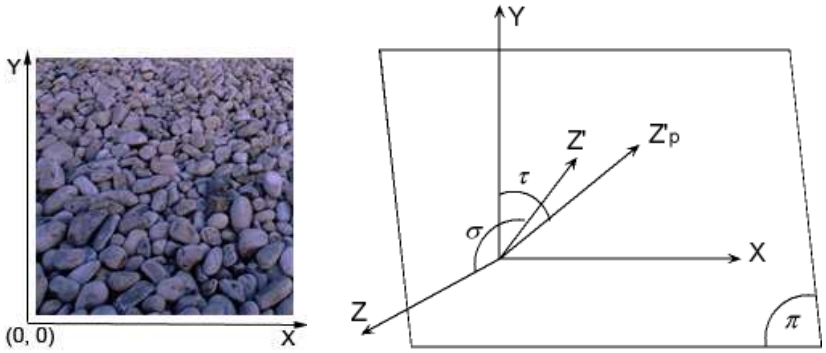
The feature-mask-based texel scale evaluation method could not get feasible scale maps for some textures. For example in Figure 6.5, the texel shape of the stones texture is randomly formed, so it is difficult to *automatically* compute the scale map through the statistical model in Section 6.3.1.

On the assumption that the texel distortion of a locally planar texture image is only due to the perspective projection, we build an approximate scale map generation algorithm based on the user-provided slant and tilt angles. As shown in Figure 6.6(a), we set the origin of the image plane to the left-bottom corner of the input sample. And we always transform the texel with the largest scale to the origin while the smallest is on the right-top, through mirroring or flipping operations. Figure 6.6(b) shows the definitions of the slant angle and tilt angle in 3D space. They are the only required user-input parameters in our algorithm.



(a) (b) (c) (d)

Figure 6.5: Calculate texel scales from given slant and tilt angles. (a) Input texture. (b) Binary feature mask. (c) Scale map calculated from the feature mask. (d) Scale map calculated from given projective angles with $slant = 60^\circ$ and $tilt = 18^\circ$.



(a) Image coordinate system for scale evaluation

(b) Slant and tilt representation

Figure 6.6: Geometric illustration for recovering scale map from projective angles. In (b), XY: the image plane; Z: the normal of image plane; π: the texture plane; Z': the normal of texture plane; Z'_p: the projection of Z' on XY; σ: the slant angle (angle between Z and Z'); τ: the tilt angle (angle between Y and Z'_p)

We assume that the maximum and minimum scale values are determined only by the slant angle σ, which can be calculated as

$$S_{max} = 1.0/\cos\sigma,\ S_{min} = \cos\sigma,\ S_{dif} = S_{max} - S_{min}$$

where S_{max} is the maximum scale value and S_{min} is the minimum one, S_{dif} is the difference of them. Let (w,h) indicate the width and height of the texture image, then the point coordinate on the image plane is varying from $(0,0)$ to $(w-1,h-1)$. We compute the maximum Y-coordinate value Y_{max} and the minimum one Y_{min} of the image plane points on the texture plane as

$$\begin{aligned} Y_{max} &= (w-1-w/2)\cdot\sin\tau + (h-1-h/2)\cdot\cos\tau \\ Y_{min} &= -Y_{max} \\ Y_{dif} &= Y_{max} - Y_{min} \end{aligned} \tag{6.2}$$

where Y_{dif} is the difference value between Y_{max} and Y_{min}. Apparently Equation 6.2 is based on the assumption that the focus of the camera is on the center of the texture plane. Then the local

scale $S(p)$ of each point on the image plane is specified by

$$p(Y_{proj}) = (p(x) - w/2) \cdot \sin\tau + (p(y) - h/2) \cdot \cos\tau$$
$$S(p) = Y_{max} - (p(Y_{proj}) - Y_{min}) * S_{dif}/Y_{dif}$$

where $p(x)$ and $p(y)$ are the coordinates of image point p, $p(Y_{proj}$ evaluates the Y-coordinate value of an image point when it is projected back onto the texture plane. The scale map computed for the texture image in Figure 6.5(a) is shown in Figure 6.5(d). We always get a smooth scale map with this method. Compared with previous work on shape-from-texture such as [RH00] and [PLB01], our method do not require to know the camera local length and the distance between the inclined plane and the camera. We treat the scale feature field f_{sca} as a function that formulates the relatively scale differences among image points.

6.3.3 Interactive Scale Editing

The scale feature field recovered from the feature mask of a texture or user-provided projective angles are sometimes unsatisfying due to various reasons: random texel shape formation, non-uniform texel distributions and different texture types etc.. Moreover, the lighting conditions (shadows, caustics) sometimes also affect the texel appearances. Instead of attempting to handle these complications automatically, we develop several intuitive tools for tuning the result interactively.

Visualizing the scale feature field as a gray-valued image, the user can increase or reduce the scale values by increasing white or black values of the pixels. The user can also duplicating some values from one area to another. Finally, the variation of scales over a region can be manipulated, as demonstrated in Figure 6.7.

(a) (b) (c)

Figure 6.7: Points value variations of the scale feature field in Figure 6.4(d) is magnified (a), minified (b) and duplicated (c).

6.4 Type Feature

The type feature in a texture image could be a different material texture (Figure 6.2(f)), an interest region (Figure 6.8(a)) or a non-textural foreground object (Figure 6.8(c)). Using normal

(a) Texture with interest regions

(b) Type map

(c) Texture with foreground object

(d) Type map

Figure 6.8: Type feature field extraction.

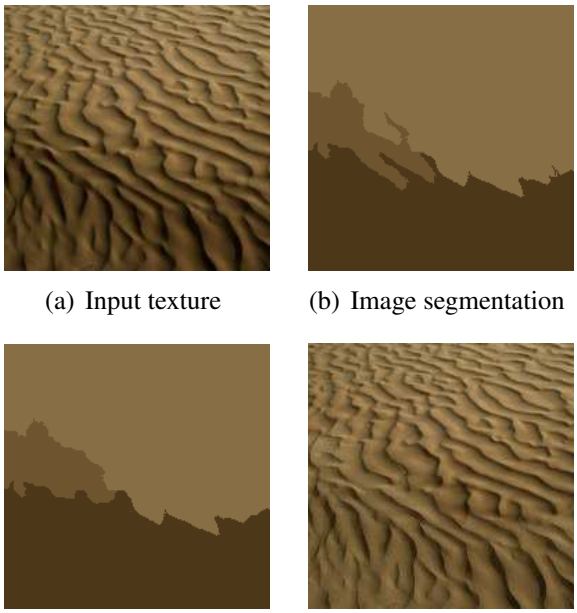
texture synthesis methods without any constraint, there is no guarantee of preserving the shapes or relative spatial positions of the type features in the output images. Our technique uses type feature maps to solve this problem. The user provides a type mask that masks the parts of the input texture whose features should be preserved. Assuming there are b type features in the image, we denote the mask by $M = \{m_0, m_2, ..., m_{N-1} | \, t \in [0,b]\}$, such that $m_p = t(t \in [1,b])$ if pixel p belongs to the tth type feature and $m_p = 0$ otherwise. $N = w \times h$ equals to the size of the input.

Apparently the type value $T(p)$ of each point in the type feature map is equal to the corresponding mask value where $T(p) = m_p$. We visualize the type feature map by representing each type with different gray values while always keeping the background to be pure black. In fact the type feature field f_{type} is just defined as a function that indicates the positions of a series of special regions in the input texture. We will subscribe two models of using the type feature map to guide the texture synthesis process.

6.5 Color Feature

There are many reasons that could cause the variation of global color or intensity in one texture, such the lighting condition, special environmental affection, and even the inner property of the texture material itself. The concept of our color feature field f_{col} is different with the color deformation field in [LLH04]. Their work emphasizes the estimation of local color space variations between tiles while ours explores the global color changes in different regions. We do

not concern the lighting map extraction problem in this chapter, Tsin et al. [TLR01]'s algorithm could be used if necessary.



(a) Input texture (b) Image segmentation

(c) Color map (d) Local color transfer

Figure 6.9: Color feature field extraction.

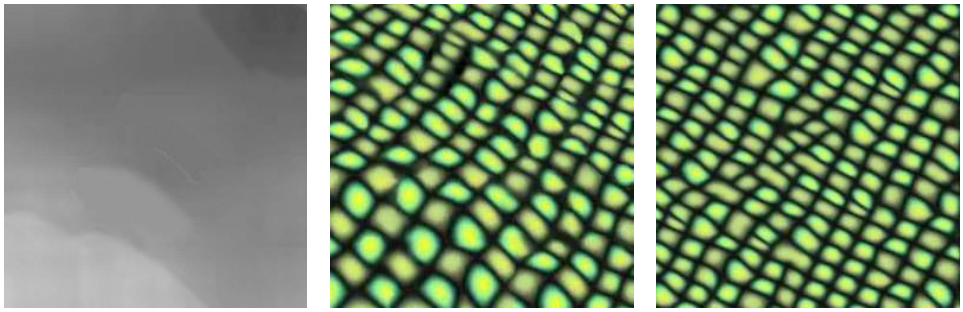
Our algorithm for extracting the color feature field works as follows: (1) apply Comaniciu and Meer [CM02b]'s algorithm for color-based image segmentation (Figure 6.9(b)); (2) edit the unsatisfying areas of the segmentation map and save it as the color map (Figure 6.9(c)), we call each area a color feature; (3) transfer color from the largest segmented area to the other areas (Figure 6.9(d)), here we simply use Reinhard et al. [RAGS01]'s method; (4) for all the color features, compute the means and standard deviations for each axis separately in $l\alpha\beta$ space, as in [RAGS01], and save them along with the color map. The step (3) is not indispensable for all the textures but in some cases it can help the synthesis process to produce results with more abundant patterns (Section 7).

6.6 Feature-Aware Manipulation

A feature field $f = f_{sca} \times f_{type} \times f_{col}$ is a multi-dimensional function that pixel-wisely represents the composition of texel scale variations, element types and global color alterations inside a texture. On the other hand, a feature field itself is an image that can be subjected to texture synthesis and manipulation, it is a combination of one or more feature maps. Our approach to synthesize perceptual-featured-textures is achieved through manipulation of their feature fields.

6.6.1 Feature-Preserved Synthesis

To produce a similar global appearance in the synthesized image as input, we first re-sample the input feature maps according to the required output size (Figure 6.10(a)). Then the stretched maps will be used as the target feature constraints in the synthesis process. The map containing different feature could be be separately used or work together for joint control. Gal et al. [GSCO06]'s technique could be employed if the shapes of some regions need to be specially preserved.



(a) Output scale map (b) Scale-preserved synthesis (c) Texture optimization

Figure 6.10: Re-sample the input scale map for scale-preserved synthesis.

We test two texture synthesis frameworks which are extended to use the feature fields as the constrained guidance. One is the texture optimization method [KEBK05], they formulate the synthesis problem as minimization of energy function, which is optimized using an Expectation Maximization (EM)-like algorithm. We apply the controllable synthesis framework in our approach but not exactly the same. The energy representing the control criterion is expressed as the sum of squared distances between the target and input feature values:

$$E_c(\mathbf{f}(\mathbf{x});\mathbf{f}'(\{\mathbf{z}_p\})) = \sum_{p\in X^+} (\mathbf{f}(\mathbf{x}_p) - \mathbf{f}'(\mathbf{z}_p))^2$$

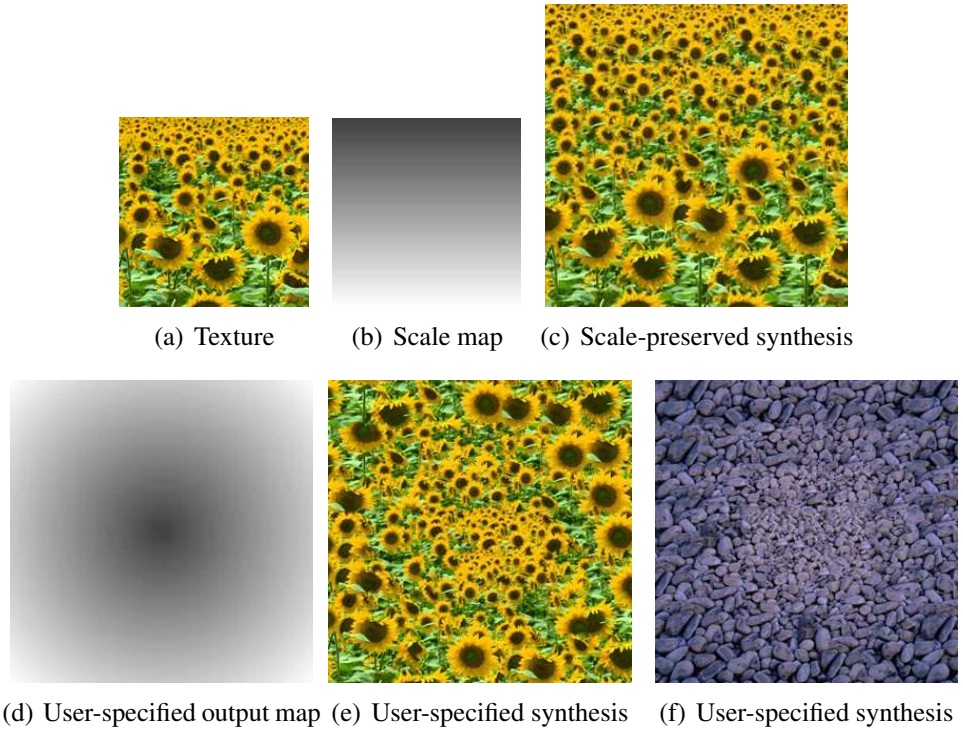
where $\mathbf{f}(\mathbf{x})$ and $\mathbf{f}'(\{\mathbf{z}_p\})$ are vectors containing the feature values in the input and target feature maps (refer to [KEBK05] for the detailed definition of the vectorized pixel neighborhood). We then define the total energy to be optimized as

$$E(\mathbf{x}) = E_t(\mathbf{x};\{\mathbf{z}_p\}) + \lambda E_c(\mathbf{f}(\mathbf{x});\mathbf{f}'(\{\mathbf{z}_p\})) \tag{6.3}$$

where λ is a relative weighting coefficient ($\lambda = 0.5$ in our experiments). The control term, $E_c(\mathbf{f}(\mathbf{x});\mathbf{f}'(\{\mathbf{z}_p\}))$, attempts to preserve the global features in the synthesized texture. We modify the initialization and E-step of the original controllable synthesis framework as follows.

In the initialization, rather than using random neighborhoods, we generate the initial neighborhoods $\mathbf{z}_p^0$ according to the input and target feature maps, i.e.,

$$\mathbf{z}_p^0 \leftarrow argmin_{\mathbf{v}}[E_c(\mathbf{f}(\mathbf{x});\mathbf{f}'(\{\mathbf{v}\}))]$$



(a) Texture (b) Scale map (c) Scale-preserved synthesis

(d) User-specified output map (e) User-specified synthesis (f) User-specified synthesis

Figure 6.11: Scale-aware texture synthesis.

where $\mathbf{v}$ is a neighborhood in the input texture Z. We also modify the E-step that minimizes the weighted energy function. For the feature-preserved synthesis described above, this corresponds to separately minimizing the two terms in Equation 6.3 instead of a weighted average in [KEBK05]. Then we use the same M-step as [KEBK05] to fix the nearest neighborhoods.

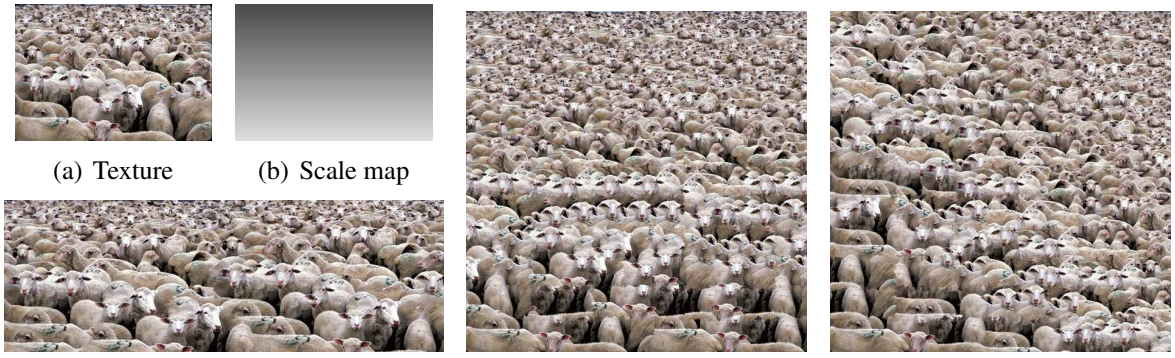
The other method is to build a similar framework as Wang et al. [WWYS04]'s work for example-based painting. They use a hierarchical image-quilting [EF01] like technique for transfer an artistic style from one photo to another. We reorganize their algorithm to be fit for our feature-preserved synthesis demand as follows:

1. Decompose the output image into regular cells, with each cell of the same size as the largest patch size specified (typically 32×32).

2. Using the target feature map as reference, recursively subdivide each cell into four smaller cells of equal size until the features of pixels belonging to the same region in every cell do not differ from each other by a prespecified tolerance. This step produces a hierarchy of cells of different sizes.

3. Synthesized each cell by composing patches of the input texture contained in that cell. All cells are synthesized in the depth-first order, with the cells at the same level visited from bottom to top and left to right.

We modify the error term of their method to be the weighted sum in our algorithm, α times the block overlap matching error (like image quilting) plus $(1-\alpha)$ times the squared error between the correspondence feature map pixels within the source texture block and those at the current



(a) Texture (b) Scale map (c) Graph-cut result (d) Scale-preserved synthesis (e) User-specified synthesis

Figure 6.12: Comparison of our scale-aware texture synthesis results with graph-cut method.

target feature map position.

6.6.2 User-Specified Feature Manipulation

Besides persevering the input global features in the result images, we could also synthesize or edit the target feature maps to be any appearances according to the user's requirements. Different features could be separately manipulated and then combined together to jointly control the synthesis process. Sometimes two or more features are strongly associated in some textures, then we can also use the image analogies-based method as in [LLH04] to synthesize these feature fields simultaneously.

6.7 Results and Discussions

We demonstrate several applications of our feature maps in texture synthesis and image completion that would be difficult or at least cumbersome to achieve with current software.

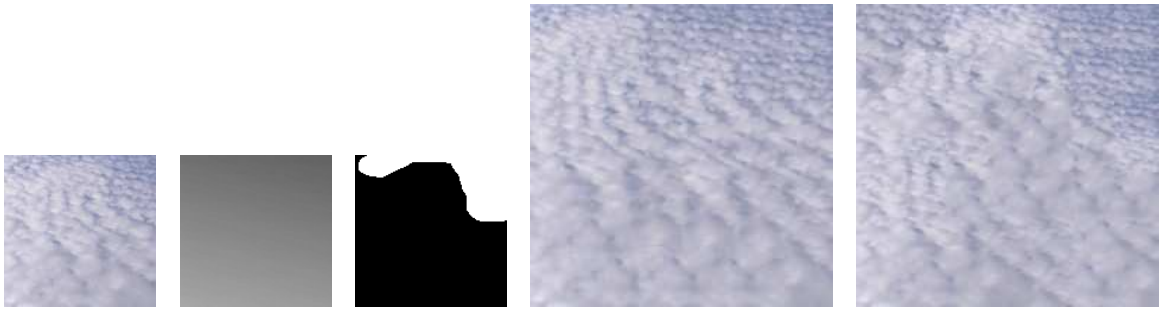
Scale-aware synthesis. The texture synthesis results which preserve the scale variations of the input are shown in Figure 6.2(c), Figure 6.10(b) and Figure 6.11(c). Compared with the texture optimization [KEBK05] results in Figure 6.2(d) and Figure 6.10(c), our images successfully preserve the global texel scale variational trends. We compare our results with graph-cut [KSE$^+$03] in Figure 6.12. The graph-cut method could only expand the example in one direction, as shown in Figure 6.12(c), the output image must have the same height as the input. While with our scale-aware synthesis method, more abundant results could be generated, as shown in Figure 6.12(d) and Figure 6.12(e). The user-specified scale-aware synthesis results are shown in Figure 6.11(e) and Figure 6.11(f), we synthesize these two images by following the given output scale pattern in Figure 6.11(d). The "SIG" banner in Figure 6.2(e) is another user-specified synthesis example.

Type-aware synthesis. We use type maps to preserve the characteristics of different interest areas in output images. We test two modes for type-aware texture synthesis, one is called

(a) Soft-constraint (b) Tight-constraint (c) Texture optimization

(d) Soft-constraint (e) Tight-constraint (f) Texture optimization

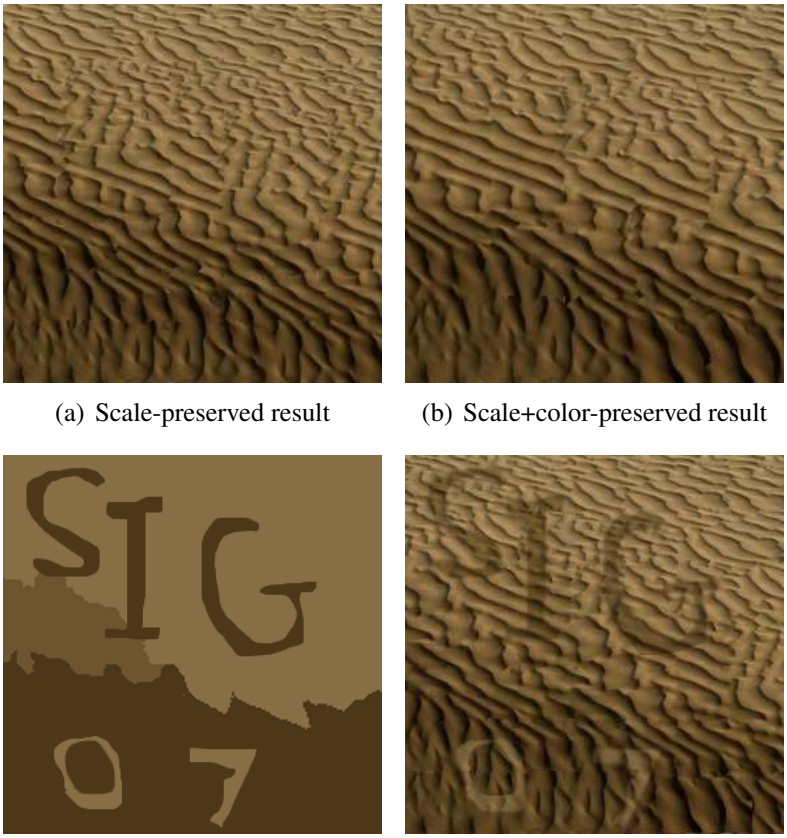
Figure 6.13: Type-aware texture synthesis of the input samples in Figure 6.8. (a) and (d) are soft-constraint results. (b) and (e) are generated by tight-constraint method which treats the background area to be a type feature, too. (c) and (f) are texture optimization results.



(a) Texture (b) Scale (c) Type (d) Scale-preserved synthesis (e) Type-preserved synthesis

(f) Multiple constraints (g) Texture optimization

Figure 6.14: Feature-aware texture synthesis using multiple feature maps. (e) is synthesized using both scale and type constraints.



(a) Scale-preserved result (b) Scale+color-preserved result

(c) Output color map (d) User-specified synthesis result

Figure 6.15: Feature-aware texture synthesis with both scale and color constraints. (a) only scale feature is used, (b) use both scale and color features, (c) user-specified output color map, and (d) user-specified color-aware result.

soft-constraint mode, which we could use any texel in the output background area (Figure 6.13(a) and Figure 6.13(d)); the other one is *tight-constraint* model, we only use "background texels" (the pure black region in the type map) when synthesizing the output background (Figure 6.2(h), Figure 6.13(b) and Figure 6.13(b)). Compared with the synthesis results generated by texture optimization method, our result nicely preserve the shapes (Figure 6.2(i)), the materials (Figure 6.13(c)) and the relative spatial position of the foreground objects (Figure 6.13(f)). The user-specified synthesis result is shown in Figure 6.2(j), the word "07" is constrained to be synthesized by the texels of the type area (the white area in Figure 6.2(g)).

Multiple-constraint synthesis. Multiple feature maps could be employed together to guide the synthesis process. For this we just need to combine these maps into one image using different channels. As shown in Figure 6.14, we can preserve both scale and type features of the input texture in the output image. In Figure 6.15, we use scale and color maps to preserve the perceptual features of the texture in Figure 6.9(a). For Figure 6.15(d), we first generate the output image with the "color consistent" example in Figure 6.9(d) with only scale constraint, then the colors of different areas are transferred back according to the user-specified color map (Figure 6.15(c)) by using Reinhard et al. [RAGS01]'s method.

(a) Input image (b) Unknown region (c) Recovered scale map

(d) Our result (e) EBI result

Figure 6.16: Image completion for textured region by feature-aware synthesis.

Image completion. We have extended our feature-aware texture synthesis to image completion where textured regions are required to be completed. As shown in Figure 6.16, we recover the scale feature from the background and then synthesize the unknown region by treating the other area to be the example texture. Compared with the Exemplar-Based Inpainting (EBI) method [CPT03], our method successfully preserve the texel scale variations in the result image.

The results in Figure 6.12 and Figure 6.15 are generated using the image-quilting based method (Section 6.2), while the others are all synthesized by the texture optimization based algorithm (Section 6.1).

6.8 Conclusion

We have presented a method that allows performing feature-aware texture synthesis for samples with special global appearances. Our method is guided by feature maps that roughly masks the specific features in the texture image; the synthesis results produced by our algorithm then preserve the appearances of those features or present user-defined feature constraints.

Automatic feature analysis for a given perceptual-featured-texture is a very complicated problem. In our current implementation, a little simple user assistance is employed to help to extract feasible feature maps, including (1) the slant and tilt angles adjustment for texel scale estimation; (2) the interest area masking for type map and (3) image segmentation map editing for color map generation. The feature-aware technique is also useful in many applications involv-

ing texture synthesis, for example the image inpainting approach in this chapter.

In our future work, we will investigate more deeply into the analysis and synthesis of perceptual-featured texture. For example, we hope to add the information of texel directions to the synthesis process, this could help to generate more realistic images together with the other features. And how to extract scale map from non-planar textures is also a challenging problem.

Conclusion

In this thesis, we study several problems on rendering natural scenes.

We first develop a framework which allows computing full spectra light object interactions only when it is needed, i.e. for the part of the scene that requires simulating special spectra sensitive phenomena. Achieving the rendering of complex scenes with both the full spectra and RGB light and object interactions in a ray-tracer costs only some additional fractions of seconds. To prove the efficiency of our framework, we implemented a "Multilayer Film" in a simple ray-tracer. However, the framework is convenient for any complex lighting model, including diffraction or fluorescence.

Then for patterns, we present a biologically-motivated algorithm for modeling and visualization of flower color patterns. It is able to produce various flower color patterns with little user interaction. The flower color is decomposed into three independent channels according to the YIQ color system. Information of each channel is either decided by pigmentation, or comes from other material properties or environment conditions. In our system, pigmentation is simulated by a modified reaction-diffusion system, and the simulation function is controlled by a few experiential parameters. Thus it can be adjusted to produce various flower color patterns widely observed in the real world. Furthermore, our algorithm can be easily embedded into other advanced shading models to improve the quality of their rendering results.

We also present a biologically-motivated system of seasonal variant scenes generation for maples, which has a obvious leaf color transformation during the time.

In the texture part, we introduce an optimized approach that can stably generate an ω-tile set of high pattern diversity and high quality. Firstly, an extendable rule is introduced to increase the number of sample patches to vary the patterns in an ω-tile set. Secondly, in contrast to the other concurrent techniques that randomly choose sample patches for tile construction, our technique uses Genetic Algorithm to select the feasible patches from the input example. This operation insures the quality of the whole tile set. Experimental results verify the high quality and efficiency of the proposed algorithm.

To make the texture synthesis approaches to be widely used, we also present an interactive tool for anisotropic 2D texture synthesis guided by feature maps, that preserves some global perceptual characteristics of the example. Using isotropic texel (TEXture ELement) distribution as our anchor point, and with simply user-assisted feature extraction, we can explicitly model the feature of a texture with respect to scale variation, element type, color and other prominent properties. We treat a feature field both as a function that acts on a texture, and as a map that is

acted upon. We develop a multi-modal framework where each feature field is subject to analysis and output manipulation. Further applications include the expansion or image completion of a featured texture region in a digital photo, and the appeared transfer of features from one texture to another in synthesis.

On the other hand, simulating insects with diffraction effects is still an open problem. We can render a set of color patterns, but our algorithms still have strong limitations for the insect patterns. And currently we can only render a large ecosystem with a texturing approach. In the future, we would like to continue our research work in this area, more accurate physical models and more advanced rendering techniques are needed to realistically render these kinds of lighting effects.

List of Figures

Bibliography

[AMN+98] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, 1998.

[Ash01] Michael Ashikhmin. Synthesizing natural textures. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 217–226, New York, NY, USA, 2001. ACM Press.

[BC50] Bevan B. Baker and E. T. Copson. *The Mathematical Theory of Huygens' Principle*. Oxford University Press, Oxford, England, second edition, 1950.

[BCM+81] C. Botten, M. S. Craig, R. C. McPhedran, J. L. Adams, and J. R. Andrewartha. The finitely conducting lamellar diffraction grating. *Optica Acta*, 28:1087–1102, 1981.

[BDE04] M. Braitmaier, J. Diepstraten, and T. Ertl. Real-time rendering of seasonal influenced trees. In Paul Lever, editor, *Procceedings of Theory and Practice of Computer Graphics 2004*, pages 152–159. Eurographics UK, 2004.

[BMH00] T. Berleth, J. Mattsson, and C. S Hardtke. Vascular continuity, cell axialisation and auxin. *Plant Growth Regulation*, 32:173–185, 2000.

[Bon97] Jeremy S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 361–368, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[BR97] Gladimir V. G. Baranoski and Jon G. Rokne. An algorithmic reflectance and transmittance model for plant tissue. *Computer Graphics Forum*, 16(3):141–150, 1997.

[BW75] Max Born and Emil Wolf. *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*. Pergamon Press, Oxford, 1975.

[BW99] Max Born and Emil Wolf. *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*. Cambridge U.P., New York, seventh edition, 1999.

[CD00] Michael T. Clegg and Mary L. Durbin. Flower color variation: A model for

the experimental study of evolution. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 97, pages 7016–7023, 2000.

[Che04] Stephen Chenney. Flow tiles. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 233–242, New York, NY, USA, 2004. ACM Press.

[cli] http://www.ncdc.noaa.gov/.

[CM02a] M. Clerc and S. Mallat. The texture gradient equation for recovering shape from texture. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):536–549, 2002.

[CM02b] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619, 2002.

[COMS96] Norishige Chiba, Ken Ohshida, Kazunobu Muraoka, and Nobuji Saito. Visual simulation of leaf arrangement and autumn colours. *The Journal of Visualization and Computer Animation*, 7, 1996.

[CPT03] A. Criminisi, P. Peréz, and K. Toyama. Object removal by exemplar-based inpainting. In *CVPR '03: Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)*, volume 2, pages II–721–II–728, 2003.

[CSHD03] Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. Wang tiles for image and texture generation. *ACM Trans. Graph.*, 22(3):287–294, 2003.

[CT82] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24, 1982.

[DCP98] Terence P. Dawson, Paul J. Curran, and Stephen E. Plummer. Liberty - modeling the effects of leaf biochemical concentration on reflectance spectra. *Remote Sensing of Environment*, 65:50–60, 1998.

[DCSD02] Oliver Deussen, Carsten Colditz, Marc Stamminger, and George Drettakis. Interactive visualization of complex plant ecosystems. In *IEEE Visualization 2002*, pages 219–226, 2002.

[DCWP02] Kate Devlin, Alan Chalmers, Alexander Wilkie, and Werner Purgathofer. State of the art report: Tone reproduction and physically based spectral rendering. In *Proceedings of Eurographics 2002*, pages 101–123, 2002.

[DGA04] Brett Desbenoit, Eric Galin, and Samir Akkouche. Simulating and modeling lichen growth. *Computer Graphics Forum*, 23(3):341–350, 2004.

[DHL+98] Oliver Deussen, Pat Hanrahan, Bernd Lintermann, Radomír Měch, Matt Pharr, and Przemyslaw Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 275–286, New York, NY, USA, 1998. ACM Press.

[DJ05] Craig Donner and Henrik Wann Jensen. Light diffusion in multi-layered translucent materials. *ACM Trans. Graph.*, 24(3):1032–1039, 2005.

[DKO05] Frank Dellaert, Vivek Kwatra, and Sang Min Oh. Mixture trees for modeling and fast conditional sampling with applications in vision and graphics. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 619–624, Washington, DC, USA, 2005. IEEE Computer Society.

[DMLC02] Jean-Michel Dichler, Karl Maritaud, Bruno Lévy, and Djamchid Chazanfarpour. Texture particles. In *Proceedings of Eurographics 2002*, pages 401–410, 2002.

[DN04] Philippe Decaudin and Fabrice Neyret. Rendering forest scenes in real-time. In *Rendering Techniques (Eurographics Symposium on Rendering - EGSR)*, pages 93–102, june 2004.

[Don06] Weiming Dong. Rendering optical effects based on spectra representation in complex scenes. In *Computer Graphics International*, pages 719–726, Hangzhou, China, 2006.

[DSP05] Weiming Dong, Shuangxian Sun, and Jean-Claude Paul. Optimal sample patches selection for tile-based texture synthesis. In *CAD-CG '05: Proceedings of the Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05)*, pages 503–508, Washington, DC, USA, 2005. IEEE Computer Society.

[DZ06] Jean-Michel Dischler and Florence Zara. Real-time structured texture synthesis and editing using image-mesh analogies. *Vis. Comput.*, 22(9):926–935, 2006.

[DZP] Weiming Dong, Ning Zhou, and Jean-Claude Paul. Feature-aware texture synthesis. Submitted to The Visual Computer.

[DZP07] Weiming Dong, Ning Zhou, and Jean-Claude Paul. Optimize tile-based texture synthesis. In *Proceedings of Graphics Interface 2007*, Montréal, Canada, 2007.

[EF01] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, New York, NY, USA, 2001. ACM Press.

[EL99] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 1033, Washington, DC, USA, 1999. IEEE Computer Society.

[FL05] Chi-Wing Fu and Man-Kang Leung. Texture tiling on arbitrary topological surfaces using wang tiles. In *Rendering Techniques*, pages 99–104, Konstanz, Germany, 2005.

[FMP92] Deborah R. Fowler, Hans Meinhardt, and Przemyslaw Prusinkiewicz. Modeling seashells. In *SIGGRAPH '92: Proceedings of the 19th annual conference on*

Computer graphics and interactive techniques, pages 379–387, New York, NY, USA, 1992. ACM Press.

[FvDFH96] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice in C*. Addison-Wesley Professional, Massachussetts, second edition, 1996.

[GG91] Allen Gersho and Robert M. Gray. *Vector quantization and signal compression*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[Ghi91] Helen Ghiradella. Light and color on the wing: structural colors in butterflies and moths. *Applied Optics*, 30(24):3492–3500, 1991.

[GJVU] Y. M. Govaerts, S. Jacquemoud, M. M. Verstraete, and S. L. Ustin. Three-dimensional radiation transfer modeling in a dicotyledon leaf. *Applied Optics*, 35.

[Gla94] Andrew S. Glassner. A model of fluorescence and phosphorescence. In *Proceedings of the 5th Eurographics Workshop on Rendering.*, pages 57–68, Berlin Heidelberg New York, 1994. Springer.

[Gla95] Andrew S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995.

[Gla98] Andrew Glassner. Aperiodic tiling. *IEEE Computer Graphics and Applications*, 18(3):83–90, 1998.

[Goo76] T. W. Goodwin. *Chemistry and Biochemistry of Plant Pigments*. Academic Press, London, UK, 1976.

[GSCO06] Ran Gal, Olga Sorkine, and Daniel Cohen-Or. Feature-aware texturing. In *Proceedings of Eurographics Symposium on Rendering*, pages 297–303, 2006.

[GTS+97] Donald P. Greenberg, Kenneth E. Torrance, Peter Shirley, James Arvo, Eric Lafortune, James A. Ferwerda, Bruce Walter, Ben Trumbore, Sumanta Pattanaik, and Sing-Choong Foo. A framework for realistic image synthesis. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 477–494, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[HG83] Roy A. Hall and Donald P. Greenberg. A testbed for realistic image synthesis. 3(8):10–20, 1983.

[HJO+01] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340, New York, NY, USA, 2001. ACM Press.

[HK93] Pat Hanrahan and Wolfgang Krueger. Reflection from layered surfaces due to subsurface scattering. In *SIGGRAPH '93: Proceedings of the 20th annual con-*

ference on Computer graphics and interactive techniques, pages 165–174, New York, NY, USA, 1993. ACM Press.

[HKY+99] H. Hirayama, K. Kaneda, H. Yamashita, Y. Yamaji, and Y. Monden. Visualization of optical phenomena caused by multilayer films with complex refractive indices. In *PG '99: Proceedings of the 7th Pacific Conference on Computer Graphics and Applications*, pages 128–137, Washington, DC, USA, 1999. IEEE Computer Society.

[Hol75] John H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, Michigan, USA, 1975.

[Hoo65] R. Hooke. *Micrographia: or Some Physiological Descriptions of Minute Bodies Made by Magnifying Glasses with Observations and Inquiries Thereupon*. Royal Society, London, 1665.

[Hop99] William G. Hopkins. *Introduction to Plant Physiology*, pages 125–141. John Wiley & Sons Inc., second edition, 1999.

[IOOI05] Takashi Ijiri, Shigeru Owada, Makoto Okabe, and Takeo Igarashi. Floral diagrams and inflorescences: interactive flower modeling using botanical structural constraints. *ACM Trans. Graph.*, 24(3):720–726, 2005.

[IP00] Jean Claude Iehl and Bernard Péroche. An adaptive spectral rendering with a perceptual control. *Comput. Graph. Forum*, 19(3), 2000.

[Jen01] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.

[JRW97] D.J. Jobson, Z.U. Rahman, and G.A. Woodell. A multiscale retinex for bridging the gap between color images and the human observation of scenes. *IEEE Transactions on Image Processing*, 6(7):965–976, July 1997.

[JSTS06] Jiaya Jia, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Drag-and-drop pasting. *ACM Trans. Graph.*, 25(3):631–637, 2006.

[KCODL06] Johannes Kopf, Daniel Cohen-Or, Oliver Deussen, and Dani Lischinski. Recursive wang tiles for real-time blue noise. *ACM Trans. Graph.*, 25(3):509–518, 2006.

[KDM02] Steven L. Kilthau, Mark S. Drew, and Torsten Möller. Full search content independent block matching based on the fast fourier transform. In *Proceedings of International Conference on Image Processing 2002*, volume 1, pages 669–672, Vancouver, BC, Canada, 2002.

[KEBK05] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. *ACM Trans. Graph.*, 24(3):795–802, 2005.

[Koz95] John R. Koza. Survey of genetic algorithms and genetic programming. In *Proceedings of 1995 WESCON Conference*, pages 589–594. IEEE, 1995.

[KSE+03] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph.*, 22(3):277–286, 2003.

[KYK02] Shuichi Kinoshita, Shinya Yoshioka, and Kenji Kawagoe. Mechanisms of structural colour in the morpho butterfly: cooperation of regularity and irregularity in an iridescent scale. *Proc. R. Soc. Lond. B 266*, 269(1499):1417–1421, 2002.

[LD99] Bernd Lintermann and Oliver Deussen. Interactive modeling of plants. *IEEE Comput. Graph. Appl.*, 19(1):56–65, 1999.

[LD05] Ares Lagae and Philip Dutré. A procedural object distribution function. *ACM Trans. Graph.*, 24(4):1442–1461, 2005.

[Lea86] B. Lear. Autumn leaves. *ChemMatters*, 10:7–13, 1986.

[LG93] T. Lindeberg and J. Garding. Shape from texture from a multi-scale perspective. pages 683–691, 1993.

[LH05] Sylvain Lefebvre and Hugues Hoppe. Parallel controllable texture synthesis. *ACM Trans. Graph.*, 24(3):777–786, 2005.

[LH06] Sylvain Lefebvre and Hugues Hoppe. Appearance-space texture synthesis. *ACM Trans. Graph.*, 25(3):541–548, 2006.

[Li93] Lifeng Li. A modal analysis of lamellar diffraction gratings in conical mountings. *J. Modern Optics*, 40(4):553–573, 1993.

[LL02] Boading Liu and Broading Liu. *Theory and Practice of Uncertain Programming*. Physica-Verlag, 2002.

[LLH04] Yanxi Liu, Wen-Chieh Lin, and James Hays. Near-regular texture analysis and manipulation. *ACM Trans. Graph.*, 23(3):368–376, 2004.

[LLX+01] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Trans. Graph.*, 20(3):127–150, 2001.

[LN02] Sylvain Lefebvre and Fabrice Neyret. Synthesizing bark. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 105–116, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[Mac81] D. L. MacAdam. *Color Measurements Theme and Variations*. Springer Verlag, Berlin, Germany, 1981.

[MCK+01] S. Mochizuki, D. Cai, T. Komori, H. Kimura, and R. Hori. Virtual autumn coloring system based on biological and fractal model. In *Proceedings of the 9th Pacific Conferene on Computer Graphics and Applications*, 2001.

[Mei82] H. Meinhardt. *Models of Biological Pattern Formation*. Academic Press, London, UK, 1982.

[Mey88] Gary W. Meyer. Wavelength selection for synthetic image generation. *Comput. Vision Graph. Image Process.*, 41(1):57–79, 1988.

[MMPP03] Lars Mündermann, Peter MacMurchy, Juraj Pivovarov, and Przemyslaw Prusinkiewicz. Modeling lobed leaves. In *Proceedings of Computer Graphics International 2003 (CGI'03)*, pages 60–68, Tokyo, Japan, 09-11 July, 2003.

[MP96] Radomir Mech and Przemyslaw Prusinkiewicz. Visual models of plants interacting with their environment. In *Siggraph 96 Conference Proceedings*, pages 397–410, 1996.

[Mur03] James Dickson Murray. *Mathematical Biology, Volume II: Spatial Models and Biomedical Applications*, pages 71–81. Springer Verlag, Berlin, Germany, 2003.

[NA03] Andrew Nealen and Marc Alexa. Hybrid texture synthesis. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 97–105, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[Nas01] Kurt Nassau. *The Physics and Chemistry of Color: The Fifteen Causes of Color*. John Wiley & Sons, New York, second edition, 2001.

[NC99] Fabrice Neyret and Marie-Paule Cani. Pattern-based texturing revisited. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 235–242, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[New30] I. Newton. *Opticks, 4th edn*. William Innys, London, 1730.

[NMMK05] A. Nicoll, J. Meseth, G. Müller, and R. Klein. Fractional fourier texture masks: Guiding near-regular texture synthesis. *Computer Graphics Forum*, 24(3):569–579, September 2005.

[NTN+06] J. Farley Norman, James T. Todd, Hideko F. Norman, Anna Marie Clayton, and T. Ryan McBride. Visual discrimination of local surface structure: Slant, tilt, and curvedness. *Vision Research*, 46:1057–1069, 2006.

[NWT+05] Tuen-Young Ng, Conghua Wen, Tiow-Seng Tan, Xinyu Zhang, and Young J. Kim. Generating an ω-tile set for texture synthesis. In *Proceedings of Computer Graphics International 2005 (CGI'05)*, pages 177–184, Stone Brook, NY, USA, 2005.

[PA00] Simon Premöe and Michael Ashikhmin. Rendering natural waters. In *PG '00: Proceedings of the 8th Pacific Conference on Computer Graphics and Applications*, page 23, Washington, DC, USA, 2000. IEEE Computer Society.

[Pai00] K. J. Painter. Models for pigment pattern formation in the skin of fishes. *IMA Volumes in Maths. & App.*, 121(3):59–82, 2000.

[Pal85] Edward D. Palik. *Handbook of Optical Constants of Solids*. Academic Press, 1985.

[Pee93] Mark S. Peercy. Linear color representations for full speed spectral rendering. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 191–198, New York, NY, USA, 1993. ACM Press.

[PGB03] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, 2003.

[Pla03] Luca Plattner. *A study in biomimetics: nanometer-scale, high-efficiency, dielectric diffractive structures on the wings of butterflies and in the silicon chip factory*. PhD dissertation, School of Electronics and Computer Science, University of Southampton, 2003.

[Pla04] L. Plattner. Optical properties of the scales of *Morpho rhetenor* butterflies: theoretical and experimental investigation of the back-scattering of light in the visible spectrum. *Journal of The Royal Society Interface*, 1(1):49–59, 2004.

[PLB01] J. Plantier, S. Lelandais, and L. Boutte. A shape from texture method based on local scales extraction: precision and results. In *Proceedings of International Conference on Image Processing*, pages 421–435, Washington, DC, USA, 2001. IEEE Computer Society.

[PMJ96] J.S. Pan, F.R. McInnes, and M.A. Jack. Application of parallel genetic algorithm and property of multipleglobal optima to vq codevector index assignment for noisy channels. *Electronics Letters*, 32(4):296–297, 1996.

[Pru98] Przemyslaw Prusinkiewicz. Modeling of spatial structure and development of plants. *Scientific Horticulture*, 74:113–149, 1998.

[Pru00] Przemyslaw Prusinkiewicz. Simulation modeling of plants and plant ecosystems. *Commun. ACM*, 43(7):84–93, 2000.

[PSS99] A. J. Preetham, Peter Shirley, and Brian Smits. A practical analytic model for daylight. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 91–100, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[RAGS01] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Comput. Graph. Appl.*, 21(5):34–41, 2001.

[RCSC03] Y. Rodkaew, P. Chongstitvatana, S. Siripant, and Lursinsap C. Particle systems for plant modeling. In *PMA'03: Proceedings of Plant Growth Modeling and Applications 2003*, pages 210–217, 2003.

[RF91] Maria G. Raso and Alain Fournier. A piecewise polynomial approach to shading using spectral distributions. In *Graphics Interface '91*, pages 40–46, Toronto, Canada, 1991. Canadian Information Processing Society.

[RFL+05] Adam Runions, Martin Fuhrer, Brendan Lane, Pavol Federl, Anne-Gaëlle Rolland-Lagan, and Przemyslaw Prusinkiewicz. Modeling and visualization of leaf venation patterns. *ACM Trans. Graph.*, 24(3):702–711, 2005.

[RH00] Eraldo Ribeiro and Edwin R. Hancock. Adapting spectral scale for shape from texture. In *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part I*, volume 2, pages 261–264, London, UK, 2000. Springer-Verlag.

[RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004.

[RP97] Gilles Rougeron and Bernard Péroche. An adaptive representation of spectral data for reflectance computations. In *Proceedings of the Eurographics Workshop on Rendering Techniques '97*, pages 127–138, London, UK, 1997. Springer-Verlag.

[SCA02] Cyril Soler, Marie-Paule Cani, and Alexis Angelidis. Hierarchical pattern mapping. *ACM Trans. Graph.*, 21(3):673–680, 2002.

[SFCD99] Yinlong Sun, F. David Fracchia, Thomas W. Calvert, and Mark S. Drew. Deriving spectra from colors and rendering light interference. *IEEE Comput. Graph. Appl.*, 19(4):61–67, 1999.

[SFD99] Yinlong Sun, F. David Fracchia, and Mark S. Drew. Rendering the phenomena of volume absorption in homogeneous transparent materials. In *In 2nd Annual IASTED International Conference on Computer Graphics and Imaging (CGIM'99)*, pages 283–288, 1999.

[SFDC01] Yinlong Sun, F. David Fracchia, Mark S. Drew, and Thomas W. Calvert. A spectrally based framework for realistic image synthesis. *The Visual Computer*, 17(7):429–444, 2001.

[SG02] D. A. Sims and J. A Gamon. Relationships between leaf pigment content and spectral reflectance across a wide range of species, leaf structures and developmental stages. *Remote Sensing of Environment*, 81(2002):337–354, 2002.

[SLC+05] Hongwei Sun, Kwok-Yan Lam, Siu-Leung Chung, Weiming Dong, Ming Gu, and Jiaguang Sun. Efficient vector quantization using genetic algorithm. *Neural Comput. Appl.*, 14(3):203–211, 2005.

[SP94] M. Srinivas and Lalit M. Patnaik. Genetic algorithms: A survey. *Computer*, 27(6):17–26, 1994.

[Sta97] Jos Stam. Aperiodic texture mapping. Tech. rep. ERCIM-01/97-R046, European Research Consortium for Informatics and Mathematics (ERCIM), Sophia Antipolos, France, 1997.

[Sta99] Jos Stam. Diffraction shaders. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 101–110, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[STP93] S. E. Sandström, G. Tayeb, and R. Petit. Lossy multistep lamellar gratings in conical diffraction mountings: an exact eigenfunction solution. *J. Electromagnetic Waves and Applications*, 7:631–649, 1993.

[Sun00] Yinlong Sun. *A Spectrum-Based Framework for Realistic Image Synthesis*. PhD thesis, Simon Fraser University, Canada, July 2000.

[Sun06] Yinlong Sun. Rendering biological iridescences with rgb-based renderers. *ACM Trans. Graph.*, 25(1):100–129, 2006.

[Tho75] J. H. M. Thornley. *Mathematical Models in Plant Physiology*, pages 42–58, 107–143. Academic Press, 1975.

[TLR01] Yanghai Tsin, Yanxi Liu, and Visvanathan Ramesh. Texture replacement in real images. In *CVPR '01: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 539–544, Washington, DC, USA, 2001. IEEE Computer Society.

[TSP00] P. S. Thenkabail, R. B. Smith, and E. D. Pauw. Hyperspectral vegetation indices and their relationships with agricultural crop characteristics. *Remote Sens. Environ.*, 71:158–182, 2000.

[Tur91] Greg Turk. Generating textures on arbitrary surfaces using reaction-diffusion. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 289–298, New York, NY, USA, 1991. ACM Press.

[TZL+02] Xin Tong, Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, and Heung-Yeung Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Trans. Graph.*, 21(3):665–672, 2002.

[VRV+05] Jean Pol Vigneron, Marie Rassart, Zofia Vertesy, Krisztian Kertesz, Michael Sarrazin, Laszlo P. Biro, Damien Ertz, and Virginie Lousse. Optical structure and function of the white filamentary hair covering the edelweiss bracts. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 71(1):011906–1–011906–8, 2005.

[VS00] Pete Vukusic and Roy Sambles. Colour effects in bright butterflies. *Journal of the Society of Dyers and Colourists*, 116(12):376–380, 2000.

[VSLW99] P. Vukusic, J. R. Sambles, C. R. Lawrence, and R. J. Wootton. Quantified interference and diffraction in single morpho butterfly scales. *Proc. R. Soc. Lond. B 266*, pages 1403–1411, 1999.

[VSLW01] P. Vukusic, J. R. Sambles, C.R. Lawrence, and R.J. Wootton. Structural colour: now you see it - now you don't. *Nature*, 410:36, 2001.

[WC83] Samuel J. Williamson and Herman Z. Cummins. *Light and Color in Nature and Art*. John Wiley & Sons, New York, second edition, 1983.

[Wei04] Li-Yi Wei. Tile-based texture mapping on graphics hardware. In *HWWS '04: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 55–63, New York, NY, USA, 2004. ACM Press.

[WF06] Ryan White and David A. Forsyth. Combining cues: Shape from shading and texture. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1809–1816, Washington, DC, USA, 2006. IEEE Computer Society.

[WFR98] Marcelo Walter, Alain Fournier, and Mark Reimers. Clonal mosaic model for the synthesis of mammalian coat patterns. In *Graphics Interface*, pages 82–91, 1998.

[WL00] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[WP95] Jason Weber and Joseph Penn. Creation and rendering of realistic trees. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 119–128, New York, NY, USA, 1995. ACM Press.

[WS76] G. Wyszecki and W. Stiles. *Color Science: Concepts and Methods,Quantitative Data and Formulas*. Wiley, New York, second edition, 1976.

[WWD+05] Lifeng Wang, Wenle Wang, Julie Dorsey, Xu Yang, Baining Guo, and Heung-Yeung Shum. Real-time rendering of plant leaves. *ACM Trans. Graph.*, 24(3):712–719, 2005.

[WWYS04] Bin Wang, Wenping Wang, Huaiping Yang, and Jiaguang Sun. Efficient example-based painting and synthesis of 2d directional texture. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):266–277, 2004.

[WY04] Qing Wu and Yizhou Yu. Feature matching and deformation for texture synthesis. *ACM Trans. Graph.*, 23(3):364–367, 2004.

[ZDM06] Ning Zhou, Weiming Dong, and Xing Mei. Realistic simulation of seasonal variant maples. In *PMA '06: Proceedings of the Second International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications*, Beijing, China, 2006.

[ZDP07] Ning Zhou, Weiming Dong, and Jean-Claude Paul. Modeling and visualization of flower color patterns. Submitted to the Interference Conference of CAD/CG 2007, 2007.

[ZG02] Steve Zelinka and Michael Garland. Towards real-time texture synthesis with the jump map. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 99–104, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[ZG04] Steve Zelinka and Michael Garland. Jump map-based interactive texture synthesis. *ACM Trans. Graph.*, 23(4):930–962, 2004.

[ZGC05] Chang-Ling Zhao, Wei-Ming Guo, and Jun-Yu Chen. Formation and regulation of

flower color in higher plants (in chinese). *Chinese Bulletin of Botany*, 22(1):70–81, 2005.

[ZZV+03] Jingdan Zhang, Kun Zhou, Luiz Velho, Baining Guo, and Heung-Yeung Shum. Synthesis of progressively-variant textures on arbitrary surfaces. *ACM Trans. Graph.*, 22(3):295–302, 2003.