Reshape and Relighting for Interactive Content Creation and Manipulation

Marcio Cabral

UNIVERSITÉ DE NICE-SOPHIA ANTIPOLIS

ÉCOLE DOCTORALE STIC
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

T H È S E

pour obtenir le titre de
Docteur en Sciences
de l'Université de Nice - Sophia Antipolis
Mention : **INFORMATIQUE**

Présentée et soutenue par
Marcio Calixto CABRAL

Remodelage et Re-eclairage Pour La Création et Manipulation de Contenu Interactif

Thèse dirigée par : **George DRETTAKIS**
préparée au sein du projet REVES - INRIA Sophia Antipolis
soutenue le 03 mai 2011

Jury :

Rapporteurs :	Prof. Christophe SCHLICK	-	INRIA Bordeaux (IPARLA)
	Prof. Bernd FRÖHLICH	-	Bauhaus-U. Weimar
Directeur :	Dr. George DRETTAKIS	-	INRIA Sophia Antipolis (REVES)
Examinateurs :	Dr. Sylvain LEFEBVRE (co-Directeur Chapitre 3)	-	INRIA Nancy (ALICE)
	Dr. Pierre ALLIEZ	-	INRIA Sophia Antipolis (Geometrica)
	Prof. Olga SORKINE	-	ETH Zurich

Dedico esta tese para os amores da minha vida, Fernanda e Mila.
para meus pais,
e meu irmão.

Acknowledgments

First and foremost I would like to acknowledge and to express my gratitude to my advisor, Dr. George Drettakis. His expertise and profound knowledge on a wide range of subjects made this thesis possible. He was truly inspiring and always eager to help and teach me new things. More than an advisor, he became a close friend and helped me through three very important years of my life. Thank you!

Another source of inspiration, and to whom I would also like to express my gratitude, is Dr. Sylvain Lefebvre. He is one of the brightest persons I have ever had the chance to work with and an exceptional programmer.

I must also acknowledge the REVES team, both current and past members. In particular Nicolas Bonneel, who would go out of his way to help others in the lab and our team assistance Sophie, who took care of everything, specially for the non-French speaking students in the first year. Pierre-Yves Lafont and Emmanuelle Chapoulie must also be thanked due to their efforts in helping me out with all the administrative things (in French!). I must also thank them, along with Adrien Bousseau, for the translation of parts of this thesis to French. Gaurav, Peter, David, Manuel and Matthaus, thanks for helping me taking countless pictures of trees!

The coffee room at the lab was a great place to hang out and I am thankful for those moments with the team: Nicolas, Ares, Peter, Carles, Pierre-Yves, Gaurav, Emmanuelle, David Grelaud, Adrien David, Georgios (and Monique, Manuel, Nicolas Tsingos, Statis and Carsten at the very beginning). I have had a great time at REVES! Thank you guys!

I would like to thank the guys at CAVERNA Digital USP, specially Prof. Marcelo Zuffo, who first inspired me in pursuing a career in Computer Graphics.

I would like to thank my family, specially my parents who have always supported me and my long standing goal of doing a PhD.

Finally, I must acknowledge my beloved wife Fernanda. We got married at the very beginning of this PhD journey and she has always being there for me, as a wife and as a friend, supporting me to achieve this goal, which would have not been possible without her. And to toast our stay in France, the greatest moment of all was reserved for the final year of this thesis: our beloved daughter Mila was born, making these three past years the happiest and most important years of my life. Thank you Fernanda and Mila!

Antibes
May 3rd, 2011

Reshape and Relighting for Interactive Content Creation and Manipulation

Abstract

Over recent years, computer graphics tools and techniques have become accessible to a wider audience, allowing non-experts to create 2D and 3D digital content of their own. However, most users still lack the skills to create content that is both attractive and useful.

This thesis proposes solutions to bridge this gap, by providing tools that can be used by non-expert users to assist content creation for virtual worlds.

We first address the complexity present in current 3D modeling tools, which are targeted at experienced users. Our approach allows users to simply drag and drop vertices of an architectural model to accomplish the desired changes, while also adapting the textures. We cast the problem as a set of linear equations representing the structure of the 3D model. At run time, the equations are solved in a least squares sense, allowing the user to make changes to the 3D model while preserving overall shape and texture.

We next adapt this approach to an immersive 3D virtual environment. The user can interact with the modeling system using gestures, while immersed in a 4-wall projection system. We augment our approach with basic lighting capabilities, allowing the user to change the day and time of the year to visualize light distribution inside the architectural model.

We conclude this thesis with a solution that allows users to change the lighting in a photograph of a tree. We adopt a single scattering volume rendering approach to approximate light distribution in tree canopies. Using a few pictures at a single time of the day as input our solution enables the user to virtually change the time of day at which the input picture was taken.

Keywords: Computer Graphics, Image-based Rendering, Human Computer Interface, Interactive Mesh Editing, Linear Systems, Image-based Relighting, Global Illumination.

Remodelage et Ré-éclairage Pour La Création et Manipulation de Contenu Interactif

Resumé

Ces dernières années, les outils et techniques d'infographie sont devenus accessibles à un public plus large, permettant á des non-spécialistes de créer du contenu numérique 2D ou 3D par eux-mêmes. Cependant, la plupart des utilisateurs n'ont pas les compétences nécessaires pour créer du contenu qui soit à la fois esthétique et utile.

Dans cette thèse, nous proposons de combler cette lacune en fournissant des outils qui peuvent être employés par des utilisateurs non-experts pour aider dans la création de contenu numérique pour des mondes virtuels.

Nous examinons d'abord la complexité des outils de modélisation 3D actuels, qui sont conçus pour des utilisateurs expérimentés. Notre approche permet aux utilisateurs de simplement déplacer les sommets d'un modèle architectural pour effectuer les changements souhaités, tout en adaptant les textures du modèle. Nous avons posé le problème comme un système d'équations linéaires représentant la structure du modèle 3D. A l'exécution, les équations sont résolues selon la méthode des moindres carrés, permettant à l'utilisateur de modifier le modèle 3D tout en préservant la forme générale et la texture.

Nous adaptons ensuite cette approche à un environnement virtuel 3D immersif. L'utilisateur peut interagir avec le système de modélisation en utilisant des gestes, étant immergé dans un système de projection avec 4 murs. Nous étendons notre approche avec des capacités d'éclairage basiques, permettant à l'utilisateur de changer le jour de l'année et l'heure pour visualiser la distribution de la lumière à l'intérieur du modèle architectural.

La dernière partie de cette thèse présente une solution qui permet aux utilisateurs de modifier l'éclairage d'une photo d'un arbre. Nous adoptons une approche de rendu volumique à un rebond pour estimer la répartition de la lumière dans le feuillage des arbres. Avec quelques photos prises à un seul moment de la journée comme entrée, notre solution permet à l'utilisateur de changer l'heure de la journée à laquelle la photo d'origine à été prise, et ce avec un éclairage cohérent.

Mots clés: Infographie, Rendu á partir d'Images, Interface Homme/Machine, Modélisation Interactive, Systèmes Linéaires, Ré-éclairage à partir d'Images, Eclairage Global

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

"Quando uma forma cria beleza
tem na beleza sua propria justificativa."
"When a shape creates beauty
its own beauty justifies it."
Oscar Niemeyer

Contents

Sculpture, painting, modeling and other forms of art have existed for as long as mankind exists. Artists, in each particular field, have expressed themselves through the usage of these various techniques, which require dexterity as well as an acute eye to make the most out of each tool.

The recent advent of computer systems has introduced new tools which allows artists to express themselves using other media. In particular computer graphics has gained much attention given that it provides the ability to create virtual worlds that look real as well as augmenting real worlds with virtual objects. Personal computing brought computer graphics tools and techniques closer to a wider audience, exposing "non-artists" to the possibility of creating art of their own. However, most users still lack the required dexterity and ability to create content that is both attractive and useful to them.

To bridge this gap by providing tools that can be used by non-expert users to assist content creation for virtual worlds, we will present new solutions by exploring both assisted content creation and interaction techniques that attempt to diminish the barrier between users and current interaction devices based on 2D metaphors.

Specifically we target two different fronts: interactive editing of architectural textured meshes and relighting photographs. For the former, we first develop a general framework allowing easy editing, and then propose a solution in an immersive environment. For the latter, we concentrate on the case of tree canopies, which, as we shall see, have several interesting properties.

We begin this introduction by presenting the main motivations to our work followed by stating our main goals. We then finalize this Chapter with a brief overview of each technique that is explored and developed in this thesis.

1.1 Motivation

We want to provide users with techniques and tools that leverage their abilities for creating content that would otherwise require artistic skills. My thesis focuses on two possible directions for this: complex editing of existing content to create new models from basic pieces and; parameterization and data extraction of lighting information in photographs to allow editing.

1.1.1 Easy Interactive Modeling

It is hard for non-expert users to translate their creative ideas to create physical objects or to the commands of a digital content creation program without the required skills to put them to practice. Even if we consider, for example, the task of small scale changes to an already created 3D model, it requires a considerable effort from an expert 3D artist to achieve such modifications in existing commercial software modelers.

On the other hand, a small number of skilled people, namely artists, make available daily a large quantity of new content, ranging from photographs to 3D models, that they create and share with others often at no charge. In a ideal world, users would be able to interactively edit this content, in its numerous formats and data types, in such a way that pleases them. Users could achieve what would otherwise be impossible to them given their restricted skills. Completely new content can be created by taking what has already been developed. In such an ideal scenario, it would be possible to create completely new content by taking and editing it accordingly and putting together different pieces.

Another important issue is that the size of virtual environments is increasing at an unprecedented rate, both in terms of spatial extent and quantity of details. As a consequence, most of the difficulty in producing interactive applications is now in dealing with the graphical content. This is especially true with online environments featuring spaces of several square kilometers.

At the same time, many games and online applications let users produce additional content (referred to as *user created content*). This has fostered large communities of so called *modders*, and thousands of additional game levels, objects and characters can be found online for the most popular applications.

For instance, gamers often like to build game levels on their own. When a game seems exhausted by the user, new game levels, created by the user himself, can open

up novel possibilities in a game. Other possibilities include architectural design where users want to visualize how ideas for remodeling a house would look like. For instance, increasing the size of the living room while keeping the kitchen with the same area.

These two trends make example-based modeling approaches extremely interesting: The idea is to let users create complex environments from a set of basic elements. Such approaches have been successfully applied to avatar and character creation [EA, 2008], and single objects [Funkhouser et al., 2004, Kraevoy et al., 2007]. However it has never been applied to large scale architectural environments, despite its attractivity: Computer artists save a lot of time by producing large amounts of content from small inputs, and non-expert users can easily use such a system, as long as they are given examples - which abound online.

Unfortunately approaches for mesh editing that work well on finely tessellated, non textured object, fail to generalize to buildings and very structured environments[Sorkine et al., 2004]. The key reason is that the properties that are preserved by those methods are not the ones that must be preserved in an architectural model.

1.1.2 Editing Photographs and Textures

One can argue that casual computer users hardly use a tool for 3D modeling - but they often take photographs. Another common activity that users normally do with computer tools are photograph edits. Users often want to make changes to a photograph taken. Currently available tools for photo editing allow a myriad of operations. However, changing the lighting in a photograph is a hard task. Users can mimic this effect by altering the photograph's brightness. But subtle effects such as indirect lighting and self-shadows are hard to change or remove.

In outdoors scenes for instance, a compelling case would be to allow users to change the time of the day or the day of the year at which the picture was taken. If such a tool existed, a photograph taken in the morning in the summer could be transformed automatically to a photograph taken in the early evening in the winter. In this context, another common editing operation such as photo montage, would benefit from inserting new elements in this scene, with matching lighting. Other editing operations on a photograph such as resizing have greatly improved over the years [Avidan and Shamir, 2007, Shamir and Sorkine, 2009]. Content-aware methods aid users to apply smart resize operations on photographs, preserving areas of interest and removing areas that will not be missed otherwise.

1.1.3 Human Computer Interfaces

During the course of development of this thesis we realized that despite enhanced technical functionality, some tools would benefit from an improved interface paradigm. 3D tools are mostly used through a 2D interface metaphor such as the mouse and keyboard. Experienced and expert users, in the long run, benefit from this paradigm as they get

proficient with the tools. Newcomers to the 3D world however need to learn the 2D interface in order to interact and possibly edit the 3D world. For small scale changes and rapid prototyping, a 3D interface is better suited. We draw inspiration from a concept video created by Bruce Branit called World Builder [1] - see Figures 1.1 (a) and (b).

(a) (b)

Figure 1.1: *World Builder by Bruce Branit:* (a) shows the interface paradigm for resizing a 3D object, which is simply pinching two corners of a box and pulling them apart, to achieve (b)

In this context devices that allow for these tasks to be accomplished play an important role. Appropriate devices will allow users to communicate better with the computer while expressing themselves with accuracy. In the long run, as showcased by the *World Builder's* concept and noted by [Bowman et al., 2008], gestural interfaces will allow users to communicate seamlessly with the computer, providing true immersion to the user.

VR immersive systems provide such environments where a user can feel immersed, given its limitations. We choose to build upon this concept and transposed one of the solutions developed in this thesis to a 3D projection environment. Given its nature, that of 3D manipulation, we felt that it was appropriate to explore this venue and develop an interface that would give users a form of 3D interaction, using 3D devices, to achieve the desired editing goals. Visual realism matters to the feeling of immersion [Slater et al., 2009] but ultimately a proper, non-invasive interface completes the scenario [Bowman et al., 2008].

1.1.4 Context and Previous Work

Several methods have been proposed to address the challenges described above. Modeling from photographs [Shlyakhter et al., 2001] or from examples [Funkhouser et al., 2004] have been explored in order to ease content creation.

Editing models also present an interesting challenge, where both geometric and the underlying semantic information needs to be taken into account for modifications to be reasonable. Office furniture arrangement has been dealt with an automatic placement algorithm proposed by [Xu et al., 2002]. Although not interactive, it allowed users to direct

[1] http://www.branitvfx.com/worldbuilder/

placement of objects in the scene as a way to enforce constraints. [Gleicher, 1994] proposed a method to integrate constraints and direct manipulation for interactive editing. If semantic information is available, more complex automatic modeling and further editing operations can be performed [Lipp et al., 2008]. Although not the case for most architectural types of meshes, highly tesselated meshes benefit from other editing approaches such as Laplacian [Sorkine et al., 2004] and Poisson [Yu et al., 2004] based editing techniques. Due to their complexity, editing occurs locally within the mesh. A more recent approach [Gal et al., 2009] targeted at man-made objects identifies intelligent curves that describe the shape of the object. Editing on the object is then performed through these curves, providing a global scale change to the model, while preserving its features.

These techniques however explore editing of 3D meshes using a 2D desktop metaphor as an interface. Other approaches try to face the problem from a different perspective, paying attention to not only the technique involved but how can users effectively act on editing these meshes. [Igarashi et al., 2007] Teddy is an interface metaphor for creating 3D shapes from 2D sketches. As reported in this work, within minutes of learning the interface, users are able to create interesting 3D models. Other techniques take advantage of the large displays and their immersive nature to create novel interface metaphors that can control the level of accuracy when interacting [Peck et al., 2009] or to reach far away objects [Pierce et al., 1999, Bowman and Hodges, 1997]. Seminal work done by [Mine et al., 1997a] explores the concept of proprioception for creating a 3D interaction metaphor, positioning objects within reach of the users hands and body.

Despite recent work, [Bowman and Fröhlich, 2005] states there has not been an increase of quality and usable solutions for VR applications in Immersive environments mostly due to the lack of proper 3D UI research. We thus focus on extending our approach in 2D to a 3D metaphor proposing a novel interface for basic architectural light and design which mixes different interaction scales at the same time (see Chapter 4).

Changing illumination in photographs is a long standing goal in computer graphics. Several methods have been proposed but require a somewhat difficult capture procedure to allow relighting offline [Yu et al., 1999] or even interactively [Loscos et al., 2000]. Inserting new objects in a photograph with matching lighting have also been solved using complex capture procedures [Debevec, 1998] which requires some effort and it is not targeted at causal computer users. Recent techniques approach this problem by taking advantage of the large amount of data available online to achieve relighting by transferring illuminant information between photographs [Lalonde et al., 2009]. These data-driven approaches work well for most cases but require a considerable amount of pre-processing.

We believe that despite previous work in these areas, there is still need for more active development for tools that aid users create content seamlessly.

1.2 Goals

Our idea is to empower the average computer user and allow him to perform tasks that would otherwise be too difficult to perform manually. In this context, our goal is to

create tools that aid users at creating new content based on existing data. We target two different areas: (1) creating new textured 3D models based on existing ones and; (2) altering the lighting configuration at existing photographs.

More specifically for (1), we focus on architectural 3D models since these are widely available online and commonly used in applications such as in games and design. For (2) we choose to edit lighting in tree canopies for outdoor environments since these are commonly encountered in most outdoor scenes.

1.2.1 Interactive Geometry Editing

To address the problem of interactive modeling of architectural environments we take inspiration from previous approaches for mesh editing. Our main contribution, as discussed at length in Chapter 3, is the development of an interactive technique that automatically extracts a set of linear equations from the input that expresses their overall shape. At run time, user interaction triggers the system to be solved in a least-squares sense. The user has the freedom to drag and move vertices of the mesh while the underlying system deals with the details preserving the original characteristics of the mesh.

Textures are a crucial way of representing details in the architectural type of meshes. They contain geometric information that adds depth and decoration to a variety of environments. And such, when reshaping geometry, we have adapted our method to a similar reshape operator for textures. Our approach identifies regions of the texture that can be stretched to concentrate deformations in these areas. Details are later re-introduced, allowing a high-quality result overall.

Geometry reshape triggers the underlying texture reshape accordingly allowing it to adapt to the new size of the mesh. Our system is interactive so all these operations are executed at run-time. Due to its interactive nature, users can experiment with several possibilities for mesh editing, going back and forth, to finally settle with the desired editing choices.

Figures 1.2 (a) and (b) shows results of our interactive mesh editing tool. Figure 1.2 (a) shows an initial set of 3D meshes from various game levels. In Figure 1.2 (b) these pieces have been reshaped and plugged together to create a new game level. Note how textures change significantly between the input model and the final reshaped geometry.

1.2.2 Editing in 3D

While experimenting with our tool for mesh editing we realized that one difficulty was the translation of the 3D commands through a 2D mouse/keyboard device configuration. The ambiguity inherent in this metaphor is not suitable for interacting in a 3D application such as in architectural editing. Visualizing and understanding the true depth of the models reduces ambiguity in interaction. Moreover, it is arguably better to use 3D commands in space for this type of geometric editing.

With this in mind, we further extended our approach and designed a 3D interface to perform geometry editing while in a immersive VR environment, and in particular the 4-sided BARCO iSpace at INRIA Sophia-Antipolis. In this environment, the user is

(a)

(b)

Figure 1.2: (a) input 3D meshes from a game level; (b) our interactive mesh editing approach allows pieces to be individually reshaped and plugged together to create new environments with textures appropriately adapted.

surrounded by projection screens depicting the virtual world. Another advantage is that thanks to an optical tracker, hand and head movements from the user can be tracked in real-time.

We use this technology and allow the editing operations on the architectural mesh to be performed in 3D. Users can grab and move vertices of the mesh, by means of a flystick, just as if they were standing in front of the actual 3D model. In this system, the user has three different options for interaction: a small scale version of the model is presented in front of him; a 1:1 scale of model and; a mixed mode, where both the 1:1 scale and small scale are presented at the same time. We then evaluate user actions and performance through these three different paradigms.

Additionally, we allow users to explore lighting design. We use a simplified version of a global illumination algorithm to evaluate indirect lighting of the architectural world being edited. Users can change the day of the year and the time of the day to simulate how sun light will illuminate these models.

Figure 1.3 shows a user changing a window position in a mixed mode environment, where both the 1:1 scale version of the 3D model and a small scale are present.

1.2.3 Relighting photographs of tree canopies

To achieve these goals we combine existing algorithms with novel techniques. First, in Chapter 6 we develop a new approach which builds on volumetric modeling of trees to allow relighting of photographs of tree canopies. Using a small set of photographs taken

Figure 1.3: The user is changing the position of a window in *Mixed Mode* (more details in Chapter 4).

from a tree at a **single** time of the day, we are able to perform relighting, i.e., change the time of the day and/or day of the year at which the photograph was taken. We build an approximate volumetric proxy of the tree canopy and estimate lighting parameters at the time the input photograph was taken. Using a single scattering volumetric rendering approach with image ratios, we are able to achieve a convincing effect for tree canopy relighting which is qualitatively comparable to a ground truth photograph of a tree canopy taken at the target time. For an example of our relighting technique, see Figures 1.4 (a), (b) and (c). Figure 1.4 (a) shows the input photograph, taken at noon. Figure 1.4 (b) shows our relighting result for changing the time of the day to 18h00. Figure 1.4 (c) shows an actual photograph of the tree taken at the target time of 18h00.

(a) (b) (c)

Figure 1.4: (a) input photograph taken at noon; (b) our relighting result changing the time of the day to 18h00; (c) a ground truth photograph taken at 18h00 for comparison.

1.3 Contributions and Organization

Our goal is to contribute tools that ease content creation. As such, we consider the following to be contributions of this thesis:

- Interactive mesh editing tool targeted at architectural models: We developed an algorithm that allows interactive editing of architectural 3D meshes, taking into account both geometry and textures. Our formulation extracts a set of linear equations for both geometry and textures, allowing users to edit the mesh in real-time while the underlying system maintains coherence of the input meshes by solving the set of equations in a least-squares sense. To our knowledge, this is the first time an algorithm proposes to edit both geometry and textures concurrently and interactively.

- 3D interface for simple interactive architectural and lighting design: We developed a multi-mode immersive user interface to allow interactive textured mesh editing. We allow users to perform simple architectural design using 3D gestures by means of a joystick in a immersive projection environment. To our knowledge, this is the first approach that allows simple architectural editing with lighting design to be performed in a VR immersive environment, which is also accompanied by a comparative study for different interface configurations.

- Relighting photographs of tree canopies using as input photographs at a single time of day: We develop an algorithm that takes as input a small set of photographs from a tree at a **single** time of the day and performs relighting of the input by changing the time and the day of the year of the photograph, effectively changing the illumination of the tree canopy. Our method uses a volumetric approach to perform relighting. To our knowledge, this is first time that an algorithm is able to relight tree canopies using only a single lighting condition as input.

1.3.1 Organization

This thesis is structured in two main parts.

Part I starts with Chapter 2 which gives a brief overview of related work to our mesh editing operator and its extension to a 3D environment. We then introduce our reshaping operator in Chapter 3 and show its results. In Chapter 4 we explain how we extend our reshape operator to a 3D environment and how we devise a set of tests to evaluate the effectiveness and performance of our solution.

Part II presents our work on relighting photographs. In Chapter 5 we review literature related to tree modeling, image based relighting and volumetric algorithms. Then in Chapter 6 we introduce our relighting algorithm for tree canopies.

Chapter 7 presents our final conclusions and discussions, giving insight to future work. Finally, there is an online Appendix, located at *http://www-sop.inria.fr/members/Marcio.Cabral/thesis/* which contains additional results for our tree relighting algorithm, containing the full set of time lapse photographs taken for ground truth comparison as well as additional ground truth renderings for our synthetic trees. The appendix also contains videos showing results for Chapters 3,4 and 6.

Part I

Easy Interactive Modeling

Introduction to Part I

Our goal is to provide algorithms for users to interactively create new content without knowledge of the specificities of the underlying software. Geometry editing in this context should be as easy as dragging and dropping vertices from an initial position to the desired location. In addition, for 3D models, textures play an important role. Textures include a lot of detail that would be otherwise prohibitively expensive to describe in the geometry alone.

Thus, we will develop a new approach that allows users to simply drag and drop vertices to achieve the desired editing goal, while adapting textures and geometry accordingly. Geometry of the 3D model is described by a set of linear equations, representing their overall characteristics such as angles and surface contacts. Texture is also described by a set of linear equations, with the key difference that we identify areas that can be stretched, i.e, areas that do not posses a lot of detail. These equations are then solved in real time and interactively, while trying to accommodate user's changes to the model. What is important here is that the user does not need to deal with the intrinsics of modeling: the underlying system handles these transformations, at interactive framerates.

We further explore this system for 3D modeling in an immersive setting. We use specific interactive techniques targeted at using our system in an immersive environment, called *iSpace*, where the user is surrounded by projection screens and can use gestures to drag and drop vertices of the geometry. In this immersive context, we also allow interactive lighting design, providing a preview of the appearance of the inside of a house for different lighting conditions, such as early morning in the summer or late afternoon in the winter. This is a new research direction and, to our knowledge, automatic and interactive mesh editing has not been explored before in such an immersive context.

In the following Chapter we introduce related work that has inspired our developments. We then delve into the intrinsics of the proposed method to show how to automatically change geometry plus textures to accommodate the user's changes to the 3D model. We finalize by showing how this system can be used in an immersive context, showing results in an *iSpace* like environment.

CHAPTER 2

Previous Work for Part I

Contents

2.1 Interactive Geometry Editing

Interactive mesh editing includes ideas from several areas. The most closely related areas to our work are procedural modeling, modeling by example, mesh and constrained editing, texture and image resizing, and room connectivity through portals. For each of them, we present research most related to our work.

Procedural modeling Several approaches have been proposed for automatic geometry synthesis. L-Systems build geometry from a rule set. They have been applied to plants [Prusinkiewicz and Lindenmayer, 1990], cities [Parish and Müller, 2001], buildings [Müller et al., 2006] and facades [Müller et al., 2007].

[Prusinkiewicz and Lindenmayer, 1990] explores mathematical models that define processes and structures, from birth to death, of plants and their interaction with the ecosystem. The authors showed impressive results using L-systems to generate strings that are interpreted with the LOGO turtle paradigm.

[Parish and Müller, 2001] uses L-Systems to model a full city. The authors extend L-Systems to take into account local constraints and global goals, replacing rules with generic templates at each expansion step. This allows the modification of parameters in external functions, reducing the complexity of rule creation.

[Müller et al., 2006] introduced CGA Shape, a grammar with production rules that iteratively adds details to buildings. The modeling process first creates a volumetric structure of the building, which is called a mass model. The modeling process continues by adding structured details to façades and then windows. Model annotation is specified during the modeling process, which aids in creating variety and variation, for instance, when creating a full city. The method however depends on the input configuration, for instance an arbitrary building footprint read from a GIS database, and it can produce inconsistent configurations that are not plausible. As with most rule-based procedural modeling, there is an intrinsic difficulty in learning the language, mostly faced by users without a background related to computer science, as noted by the authors.

[Müller et al., 2007] is a procedural façade modeling system which takes as input a single aerial image of a building façade and enhances it by matching 2D features with a database of existing 3D models representing several types of windows and doors. The system automatically identify repeating features, both horizontally and vertically, such as windows and doors for a building façade. The method uses the Mutual Information statistical model, which compares intensities of nearby regions of the input image to detect repeating elements, grouping them in a hierarchical order within the façade. It works for low resolution input textures but it fails if the façade structure is not symmetric, with prominent details and variation across floors.

Similarly, geometric languages let the user write programs generating complex shapes from simple operations [Berndt et al., 2005]. While such approaches often provide very impressive results, the main drawback is the level of expertise they require: The rules have to describe how every single geometric primitive is to be placed in the scene.

[Lipp et al., 2008] attempts to leverage the intrinsic difficulty of procedural modeling and grammar editing with a visual tool. Despite impressive results with visual interfaces, understanding of rules and grammars is still necessary, to a certain extent, to create/modify initial models.

These methods produce impressive results; however they all require manual labeling of 3D models which makes them unsuitable for our approach. We favor the use of 3D models available online which may not contain such information. We believe that this is a frequent use-case, especially as more and more ready-made model libraries become available.

Modeling by example [Merrell, 2007] proposed an example-based scene synthesis method producing impressive results. The method is inspired by traditional texture synthesis and it can be seen as an extension to 3D of the traditional 2D texture synthesis. The example is given as a set of building blocks aligned on a regular 3D grid, which evidently have to be very carefully modeled. As noted by the author, example models which are not carefully constructed will reproduce exactly the input model, given the tight constraints present on the input example. By reproducing the neighboring relationships of the input blocks in the output, the algorithm automatically generates larger randomized environments. The method was recently adapted to handle arbitrary inputs [Merrell and Manocha, 2008].

Other approaches have been proposed to model from examples, in particular by assembling mesh pieces. [Funkhouser et al., 2004] let the user slice parts from objects and assemble them in new ways. The process is interactive and intuitive: it allows the user to draw strokes to segment parts of 3D models. Strokes are used as input to search for a best cut on the mesh that satisfies some criteria such as cutting only through smaller edges, choosing edges closer to the center of the stroke, etc. Once parts have been segmented, they can be assembled together by using a two-par process: placement and attachment. Placement is achieved using a voxel version of the ICP (Iterative Closest Point) algorithm [Besl and McKay, 1992]. The ICP algorithm will converge to a locally optimal solution, which minimizes the sum of the squared distances between the mod-

els. The voxel version of the ICP does not need an initial guess and is guaranteed to give a globally optimal solution. Attachment uses a simple heuristic to find corresponding edges and create fillet edges connecting both independent parts, creating a watertight and smooth connection. Despite impressive results, the system is limited to connecting segmented parts of matching regions, for instance, animal heads or house ceilings. The system does not handle connection of completely different parts, for an example, as cited by the authors, a bell in a church does not match a bell on a fire engine. The segmentation system also does not handle well strokes that are too close to the boundary of the mesh and it does not handle occlusions (for mesh parts that are not visible to the user) although the authors provide a remedy to this problem using a "laser cut" segmentation, which cuts through the geometry. The results produced are interesting and it allows users to produce a large variety of new models.

[Kraevoy et al., 2007] follow a similar approach. Unlike [Funkhouser et al., 2004], the proposed method automates alignment and segmentation, which avoids putting together segments that differ significantly. The automatic segmentation aims at providing perceptually meaningful decomposition of input meshes, which can be both natural or man-made objects. The heuristics for segmenting the input meshes rely solely on geometry features such as convexity and angles, which limits segmentation of more semantic parts such a cheek or a forehead on a human face. The system was tested using a prototype interface called *shuffler* which automatically selects possible segments for a given model part chosen by the user, giving a naive user a powerful tool to produce impressive results.

Both methods [Kraevoy et al., 2007] [Funkhouser et al., 2004] are targeted at creating objects and would be difficult to adapt for entire environments. A similar approach is used by [Zhou et al., 2006] at a much finer scale to synthesize mesoscale structures. Patches of geometrical details are carefully stitched together to cover a surface. This approach draws inspiration from regular 2D texture synthesis - it carries the same limitations of earlier approaches such as repetition. The algorithm heavily depends on the underlying geometry and the local parameterization of surface patches which presents a problem for regions with high curvature such as objects with high genus or complex models. Similarly to some 2D texture synthesis methods, if the 3D swatch is not tileable, perfectly aligned synthesis and matching cannot be achieved.

More recently, data driven approaches have been developed to ease 3D modeling. [Merrell et al., 2010] introduced a data driven approach to model architectural houses from high level input. The method produces plausible results from architectural high level input and user's constraints. The optimization method however takes at least a few seconds for the simpler models, making it unsuitable in its current form to interactive techniques.

Mesh editing Mesh editing techniques provide interactive tools to deform a mesh while retaining its overall appearance, for instance such as Poisson mesh editing [Yu et al., 2004] and Laplacian surface editing [Sorkine et al., 2004]

[Yu et al., 2004] introduces a strong theoretical formulation to mesh editing using the Poisson equation. The formulation allows mesh editing based on gradient field ma-

nipulation and boundary condition editing. This technique manipulates the gradients of each vertex of the mesh and deforms the reconstructed surface by solving a least-squares system. The system is formulated as a parameterization of the vertices of the mesh into a discretized Poisson equation with Dirichlet boundary conditions. The method allows large-scale and local deformation of the mesh while preserving features. The authors also develop a powerful interface capable of manipulating boundary conditions to achieve interactive mesh editing in a intuitive manner to users.

Alternatively, Laplacian surface editing [Sorkine et al., 2004] represents geometric details of a mesh by means of their distances between vertices and the average of all their respective neighbors. This is known as a Laplacian coordinate. The key contribution of this work is the ability to make Laplacian coordinates invariant to rotation and isotropic scaling in addition to translation. A transformation T which encodes linear transformations of isotropic scale and translation of a vertex is introduced into the Laplacian coordinate implicit representation of the mesh. When solving in a least squares sense, T is computed and allows for scaling and rotation, which Laplacian coordinates cannot handle. The bulk of processing occurs when factoring the matrices, while solving for a new user vertex position request is fast. Due to performance constraints, the system works locally, on a limited, well-defined area of the mesh. Nevertheless, it can handle interactively segmented areas of a mesh with at least 10000 vertices.

In related work, a method for model resizing has been recently proposed [Kraevoy et al., 2008]. The proposed method allows axis-aligned non-uniform scaling of 3D models. Vulnerable parts of the mesh are identified - they will receive most of the stretching during deformation. These areas are selected based on metrics such as slippage [Gelfand and Guibas, 2004] and surface normal curvature. Deformation on the 3D model occurs indirectly, by applying changes to a 3D grid which encloses the full model. Each grid cell has scaling properties defined by the metrics of the underlying geometry. These properties are expressed in a system of linear equations and express locally how each cell changes. This is the first step; this system of equations is solved using a preconditioned MINRES solver [Toledo et al., 2003]. Another set of linear equations restricts the global motion of the cells, keeping the grid cells aligned to their axis, and prevents undesirable sheering of the model. This global system of equations is solved using a conjugate gradient solver [Toledo et al., 2003]. The proposed method performs well for most man made shapes. However, memory and computation time implied by the 3D grid used would be a big handicap for our particular case since it allows only axis-aligned reshaping.

Most of these approaches work very well as long as the mesh is smooth and finely tessellated. In particular, highly tessellated meshes often depict organic shapes, where they achieve impressive results and allow interactive changes to a mesh through user interaction, while preserving the overall original shape. However, this does not hold for architectural pieces, where the tessellation is often irregular and sharp edges are common. Our approach is inspired by these works and it provides a new formulation better suited to our needs.

Constrained editing In earlier work, [Gleicher, 1992] outlined the benefit of mixing direct manipulation with constraint solvers. Spatial relationships between objects are inferred from user manipulations and later maintained by the system. The ability to continuously update the model while the user interacts with it is called Differential Constraints by the authors. To maintain constraints during this dragging operation performed by the user, the authors treat the model as a differential equation and solve a sparse-linear systems to maintain the non-linear constraints. In comparison to our approach, we disturb the model slightly, subdividing each user interaction into small steps and solve for each one of this changes. Gleicher notes that this rapid visual feedback is essential. It allows the user to explore under-constrained models permitting user experimentation and a better understanding of how the model behaves when a particular motion is applied. The author developed a system called *Briar* to test the proposed techniques.

Similarly, [Xu et al., 2002] helped the layout of many objects in a scene by guiding user manipulation using constraints. The objects can be placed automatically. Objects are placed in the scene in a 2D manner: 3D objects are reduced to 2D polygons by projecting their convex hull on the ground plane. This produces simple and convex polygons. With these conditions, the 2D spatial planning problem can be solved using the classic theory of Minkowski sums and differences. Objects are also organized semantically, for instance, with attributes as "can support another object". Finally the system developed also integrates a pseudo-physics engine, disallowing object interpenetration and automatically placing objects in a stable position. A scene layout containing 300 objects can be accomplished in approximately 10 minutes using the proposed system. Alternatively the user can guide the process or suggest modifications by restricting the positioning of an object.

We follow a similar trend proposed by these two papers: we let the user manipulate a scene while the system automatically updates vertex positions and textures through constraints. We take advantage of the visual feedback, as noted by [Gleicher, 1992] to provide the user interactive editing abilities to explore model variation. Also, similar to the Snap-Dragging technique by Gleicher, we allow the user to interactively add constraints to the 3D model, integrating them automatically into the set of linear equations and changing the behavior of future mesh editing operations.

Texture and image resizing There has been much recent work on texture and image resizing. The closest approach to ours has been proposed by [Wang et al., 2008]. A grid is used to deform an image while preserving gradients. This method could be adapted to our needs, although our emphasis is more on structure versus detail rather than salience, as targeted by the authors. The grid is overlaid on top of the original image; regions with homogenous content are stretched and regions with details, considered to be important, are simply scaled. Important (or "attractive pixels" as suggested by the authors) are identified by a significance map, which is a composition of two measures: image gradient and saliency. Saliency identifies pixels that are different from its surroundings (in different scales). Like previous methods, it does not preserve straight lines or prominent lines that are not aligned to the xy-axis of the overlaid quad mesh.

Seam carving [Avidan and Shamir, 2007] minimizes energy to find the appropriate

image seams to remove (or add),effecting content-aware image resizing. Seams are optimal paths in a image, either horizontally or vertically, that minimizes its cost for a given energy function. Reducing the size of an image is achieved by removing successive optimal seams. To increase the size of an image, optimal $k-$seams are found first and then replicated using the average of their neighboring pixels: this avoids repetition and stretching artifacts. The authors experimented with several cost functions such as L_1 and L_2, saliency map and the Harris-corners measure to show that no single function performs better across all images. Although all functions produces similar results, they found that a simple energy function that minimizes the differences between neighboring pixels in both the horizontal and vertical directions performed well for most cases. These costs functions, do not identify visual cues and important structured areas, such as human faces, and thus the algorithm fails to preserve these areas.

[Tai et al., 2008] use texture synthesis from example [Wei and Levoy, 2000] to recover details in stretched image areas when a 2D texture is applied to 3D models. Usually this 2D mapping to the 3D geometry produces regions with highly distorted textured appearance. These distorted areas are identified and amended using an adapted version of the graph-cut texture synthesis technique [Kwatra et al., 2003] that takes into account texture flow. Flow is identified manually by user strokes. Texture synthesis is applied to a high-frequency version of the stretched area to reduce illumination effects in the final result. The resulting high frequency synthesis is then combined with the original low frequency texture using a standard Poisson image blending technique. Like most texture synthesis techniques, this approach works best for near-regular and stochastic textured regions.

These ideas are related to our approach to reintroduce details to stretched texture areas. Our method extracts tiles from the stochastic textured regions. These tiles contain details that are re-introduced in the stretched areas (see following Chapter, Section 3.4.5). If the stretched area is bigger than the tile, a modulo operation allows the tile details to be repeated across the textured region. The bigger the stretch, the more the re-introduced details will appear. The key difference is that our approach must allow for interactive feedback. We call this approach *detail sliding*.

Interiors as connected rooms Representing scenes – especially interiors – as a set of rooms connected together is very convenient for fast on-line visibility determination [Teller and Séquin, 1991]. [Lefebvre and Hornus, 2003] proposed a way to obtain such a decomposition automatically from a binary partitioning of the scene. The many small convex cells are merged into larger ones, following a rendering cost-driven heuristic. [Haumont et al., 2003] position portals at strangulations of the volume distance field defined by the distance to the walls. This distance field D in 3D serves as an adaptation to classic 2D image segmentation algorithm called watershed, which a flooding process starts at places of local minimum height and identifies where valleys merge. The merging area is replaced by a damp which is the location of the portal.

In contrast to the above approaches, we consider geometry and texture reshape together. However, we do explore room connectivity using pre-defined portals, which are elements that identifies openings in a 3D model. This simplification allows a sim-

ple alternative for the propagation of texture and geometry reshape to other models, as explained in Section 3.5.

2.2 Interactive 3D Manipulation

Interactively changing geometry is inherently a 3D task. The methods previously described however are targeted at using a 2D desktop metaphor for interaction. This interface paradigm imposes restrictions on the activities performed by users of such systems, limiting the full potential of such applications. Recent work on surface editing and modeling has focused on how to provide a better interface for users to express their skills [Igarashi et al., 2007]. We extend our interactive geometry editing approach to an immersive environment, where the user is surrounded by stereo projection screens, providing users the ability to use 3D gestures to effectively edit geometry. Additionally, we introduce interactive lighting design for simple architectural 3D models. This is a novel task that takes advantage of the immersive nature of a 3D projection environment. In the rest of this section, we describe work in this area that is relevant and has inspired our work.

[Ball et al., 2007] show that physical navigation improves user performance and interaction for perception of data in visualization. Additionally, in a recent publication, [Peck et al., 2009] presented a multi-scale interaction for tasks such as selection and manipulation, which adapts itself depending on the distance between the user and the visualization device.

We draw inspiration from these ideas for interaction targeted at architectural editing. We focus on building a system that allows users to design a simple house composed of separate rooms. We take advantage of the immersive setting and provide the users a way to explore the designed house from different viewpoints including a 1:1 scale where the user can position himself and navigate inside the house. In this context useful operations are changing the height and widget of a room; adding windows and doors to a room; changing the size of windows and doors and; adding more rooms to complete a full house.

These operations are available to the user through widgets. These widgets only appear near a surface when the user gets close to them, avoiding unnecessary clutter in the virtual environment. Widgets are positioned where the user can easily perform co-located gestures to achieve the desired editing operation. Our definition of co-located gestures means that the user can, without discomfort, use his hands to reach the model in front of him.

Depending on the scale of the object with respect to the user, changes to the model are either likely to be local, directed toward a specific detail of a specific part of the 3D model (for instance, changing the size of a single step in staircase) or global (for instance, changing the size of corridor, or of a roof top).

We use a form of multi-scale interaction [Peck et al., 2009] to allow users to interact with the system using different modes. Each mode has a different scale and allows the use of a different viewpoint for manipulating the objects in the environment. We believe

that fatigue is also reduced by using a multi-scale approach for interaction, as user's gestures are confined to a smaller 3D space. The available modes for interaction are based on exocentric and egocentric paradigms and are called Table Mode, which follows the exocentric paradigm; Immersive Mode, which follows the egocentric paradigm and a mode which is a composition of the two previous modes, called Mixed Mode. These modes will be describe in detail in Section 4.3.

In architectural design, users need tools for interaction which are simple yet provide enough flexibility to achieve the desired goal. Simple and generic manipulation techniques such as Go-go [Poupyrev et al., 1996] may be limited for architectural design, since they are not practical when dealing with a large number of objects [Chen and Bowman, 2009]. Architectural design and geometric modeling have traditionally been privileged applications for virtual reality systems [Deisinger et al., 2000, Bullinger et al., 2010, Leigh and Johnson, 1996]. Our approach draws inspiration from these and others but addresses the problem from a different perspective. We restrict users to a small number of available actions depending on their position with respect to the 3D environment. This will unclutter the environment displayed. We also provide a combination of different viewpoints, at different scales, for interaction, allowing a more precise manipulation, depending on the task at hand.

The closest previous work is the World-in-Miniature (WIM) system [Stoakley et al., 1995]. They presented a miniature version mixed with a scale 1:1 model (equivalent to our "Mixed Mode"), but for simple manipulations of viewpoint and of the virtual scene, using an HMD. The viewpoint can be interactively changed by rotating a tennis ball on the user's non-dominant hand. The user's dominant hand holds another device, used to interact with objects in the scene. These objects are placed "on top" of the tennis ball device so the user can reach them and move them around. Looking at the tennis ball gives the user an aerial viewpoint of the whole Virtual World.

In contrast, we allow the user to actually change the shape and size of objects in the scene automatically. As we shall see in Chapter 3, our system provides automatic assistance in the context of a "quick prototype" modeling or *conceptual design* application [Anderson et al., 2003]. More sophisticated modeling/design operations can thus be performed. In addition, we study lighting design in this context, using a simple "one-bounce" global illumination approximation to evaluate the consequences of light source changes in an immersive setting.

By targeting specific developments to the universal 3D tasks of manipulation, selection, system control, navigation and symbolic input [Bowman et al., 2002a] we can focus on providing a resourceful experience for architectural environment design. Our study is related to several aspects of manipulation of objects in large scale and immersive displays; these are usually accomplished with some degree of physical navigation and interaction.

The idea of automatically scaling an object and placing it within reach in front of the user has been proposed by [Mine et al., 1997b] and makes use of the concept of proprioception. [Pierce et al., 1999] make use of this concept in their Voodoo Dolls technique where a copy of an object that is far away is brought close to the user's hand for interac-

tion. In this thesis we will build on these ideas for interaction, as discussed in Chapter 4

We use a simple form of bimanual interaction for resizing (similar to that used on 2D touch-sensitive devices). [Malik et al., 2005] propose a bimanual touchpad for interacting with large displays from a distance. In asymmetric interactions, the non-dominant hand coarsely controls the region of the display that maps to the touchpad, while the dominant hand does the fine-scale interaction in that region. In symmetric interactions, the region is locked and both hands interact inside it. [Owen et al., 2005] study the faster performance of bimanual manipulation, especially for cognitively demanding tasks. [Moscovich and Hughes, 2006] show that multi-finger touchpad interaction allows for simultaneous control of several object properties, such as translation, rotation, and scaling. In previous work, bimanual interfaces have been used in immersive settings [Balakrishnan and Kurtenbach, 1999]. We try to follow ideas which were initially observed by [Guiard, 1987] and others ([Cutler et al., 1997], [Zeleznik et al., 1997]) regarding the usage of both hands in our resizing gesture.

There has been a lot of previous work on hand and fingers gesture recognition, e.g. [Laviola et al., 2004, Keskin et al., 2003]. Most of them describe in detail software and/or hardware for actually recognizing hand's gestures. To overcome the problem of previous input devices for reading noise free data from hand motion, alternative interaction methods appeared such as using only discreet data from pinching of fingers [Bowman et al., 2002b, Bowman et al., 2001, Laviola, 1999].

In our system, described in Chapter 4, the user may switch between an immersive, scale 1:1 view and a world-in-miniature view, which implies a viewpoint change. [Ware and Osborne, 1990] proposed a technique called Eyeball in Hand in which the user directly controls the viewpoint with a device that directly maps its position and orientation. Our approach adopts the use of widgets to disambiguate rotation in all axes and provide the user more precise control. Go-Go [Poupyrev et al., 1996] and PRISM [Frees and Kessler, 2005] switch between direct (1:1) and scaled interaction, either for large-scale or precise manipulation, based on the user's intentions which are determined by the velocity of the hand. The HOMER system [Bowman and Hodges, 1997] attaches a virtual hand to an object selected by a light ray selection, allowing the user to manipulate far objects with no extra work.

[Lucas et al., 2005] introduce alternative 3D resizing widgets, including the straightforward generalization of the familiar 2D resizing widgets, the Pointer Orientation based Resize Technique (PORT), and the Gaze-Hand resize technique. Users prefer the familiar widgets, but after a training period they perform faster with the novel widgets.

[Buchmann et al., 2004] present a finger tracking and haptic feedback setup that allows users to manipulate virtual objects naturally in a HMD AR environment. [Grossman et al., 2004] describe multi-finger gestures for interaction with volumetric displays where hand motion is restricted to the outside of the display's enclosure.

2.3 Conclusion and Discussions

Previous work on geometry editing and manipulation has provided solutions based on differential coordinates that work well on highly tessellated meshes [Yu et al., 2004, Sorkine et al., 2004]. While interactive, these methods are not suitable for architectural meshes, which are composed of a few polygons and mostly straight angles between edges. On our case, we find novel formulations that express the shape and connectivity for this class of meshes, composed of only a few vertices. This class of meshes is composed of game levels, which for performance reasons is mostly composed of a smaller number of meshes and a lot of detail is encoded on the textured faces. We call this set of meshes architectural meshes. We do however use the same ideas as [Yu et al., 2004, Sorkine et al., 2004] for solving our linear system expressions at an interactive rate by pre-factoring the equations only when needed and doing the back-substitution at every interaction from the user.

Our method also couples deformation of geometry and textures. The reshaping that occurs at each face triggers the deformation of the underlying texture accordingly. Although we could use more resilient methods for texture stretching [Avidan and Shamir, 2007] we needed to keep our system running at interactive rates, in order to provide the user with realtime feedback of the results of his actions. Additionally, game level textures (or architectural types of textures) encode details such as straight lines and other types of decorations, which are not guaranteed to be preserved by these methods [Wang et al., 2008, Avidan and Shamir, 2007]. Newer methods exist that identify straight lines and try to preserve them when stretching a texture [Barnes et al., 2009, Laffont et al., 2010, Chen et al., 2010]. Although we could use them, we opted for a simpler, faster method that could be used coupled with our geometry reshape. Our texture stretching will be explained in Section 3.4.

To a certain extent, all these methods for shape deformation work well, depending on the input model. In our tests however, we identified that one of the main problems for users trying to create new content is to overcome the interface barrier. Learning and mastering a software interface is crucial to achieve the desired results and make the most out of a 3D tool. We address this issue by taking our reshaping operator from a 2D to a 3D metaphor. We believe that reshaping of 3D meshes can be better understood in a 3D environment. We explore this idea and develop a solution that allows our reshape operator to be used interactively in a 3D Virtual Reality immersive environment. We further extend our tool and also allow lighting design to be explored in a architectural context.

In the following Chapter we introduce the details of our reshaping operator. We begin by a brief motivation Section. We then describe how geometry reshape works in this architectural context. Then we introduce the coupled texture reshape and how we can connect different pieces of mesh together to create a novel 3D world or game level. Then in Chapter 4 we bring our reshaping operator to the 3D immersive environment, showing how novel interaction techniques can be used to aid users edit geometry using 3D gestures.

CHAPTER 3

User Assisted Architectural Geometry Editing

Contents

In this Chapter we will show how to interactively deform architectural geometry, coupled with textures. We show results on sets of 3D models both found on the web and developed in-house. Our method easily allows novel game levels and 3D models to be created from existing content.

3.1 Reshaping of Architectural Geometry

In order to ease the creation of large environments such as game levels, it is typical to prepare a set of basic building-blocks that are later combined together. However, these building-blocks have to be very carefully prepared to ensure proper combination. Moreover, too much repetition of the same elements will quickly become visible to the user. Modeling large architectural environments is a difficult task due to the intricate nature of

these models and the complex dependencies between the structures represented. Moreover, textures are an essential part of architectural models. While the number of geometric primitives is usually relatively low (i.e., many walls are flat surfaces), textures actually contain many detailed architectural elements.

We present an approach for modeling architectural scenes by reshaping and combining existing textured models, where the manipulation of the geometry and texture are tightly coupled. We make a set of assumptions to achieve this goal which we consider to be reasonable. In particular, for geometry, preserving angles such as floor orientation or vertical walls is of key importance. We thus allow the user to interactively modify lengths of edges, while constraining angles. Our texture reshaping solution introduces a measure of directional autosimilarity, to focus stretching in areas of stochastic content and to preserve details in such areas. We show results on several challenging models, and show two applications: Building complex road structures from simple initial pieces and creating complex game-levels from an existing game based on pre-existing model pieces.

We first introduce a coupled approach to geometry and texture reshaping of structured, architectural models. Our approach lets the user interactively compose large architectural environments by combining pieces cut out of example scenes. Pieces can be connected together and moved around through simple drag and drop, while geometry and texture are updated on-the-fly. The geometry of each piece is also modifiable by simple drag of vertices, the rest of the scene automatically updated to accommodate the changes. Our approach relies on a reshaping operator targeted at architectural pieces, taking both geometry and textures into account.

In Section 3.3 we explain how to define the constraints to maintain edge directions and other desirable properties and how we efficiently solve for *geometry reshape*. In Section 3.4, we introduce *directional autosimilarity* and show how it is used to *reshape texture*. Our approach preserves structured parts of the texture while re-introducing detail in stretched regions. We present examples and applications in Section 3.5 and discuss limitations in Section 3.6.

3.2 Motivation

The cost of developing modern interactive applications such as games is often dominated by the creation of a large number of detailed textured models. For many such scenes, such as typical "game levels", these models are often architectural, or more generally man-made, structures: In this thesis we will be focusing our attention to this class of models. Traditionally, such assets are created by trained artists, who create the models and textures for each scene. While the tools used have improved in the past few years, this remains a tedious and painstaking manual process.

A different approach to create such content is procedural or grammar-based modeling [Prusinkiewicz and Lindenmayer, 1990, Müller et al., 2007]. While these methods hold great promise and can be very powerful, they require a "programmer-like" approach to modeling, making them hard to use, with a steep learning curve for typical

modelers/artists.

Figure 3.1: Examples of our approach used to build road structures based on a small number of initial blocks. We show the building blocks (sides) and two example constructions made from these blocks in a few minutes, as shown in the accompanying video (located in the online Appendix *http://www-sop.inria.fr/members/Marcio.Cabral/thesis/*). Notice how pieces deform and adapt, and how texture replicates but also preserves detail in cases of stretching.

Our solution allows the user to interactively modify or reshape *textured geometry*, since in the interactive applications we focus on a large part of the detail is usually incorporated in textures. Conceptually, our solution lies between the two methods discussed above. In our approach, both the geometry and the texture adapt to these modifications in an intuitive manner. This opens the way to easily creating large varieties of models from small sets of pre-existing model "pieces", as can be seen in Figure 3.1. An additional motivation for our approach is that appropriate model pieces are becoming widely available as communities of modelers (or "modders") create and distribute them on a massive scale [EA, 2008].

Our main design choice is to provide interactive feedback to the user while modifying textured geometry. Our approach is thus based on the definition of appropriate constraints and a fast least-squares solution. The interactivity of our approach gives sufficient flexibility to enable repeated tailoring of the constraints, thus allowing the user to obtain the desired result. In contrast to grammar-based approaches, which imply knowledge of model semantics, we do not assume any high-level knowledge of the model, e.g., that one part of the model is a door and the other a window. However, since we concentrate on architectural/man-made structures for interactive applications, the meshes do have an inherent "expected behavior" which we will seek to preserve. We thus base our reshaping operator on the key insight that angles are most important in keeping the aspect of a room or structure. Obvious examples are the horizontal orientation of the floor and the vertical orientation of walls. By preserving angles we avoid that a wall becomes tilted in a weird angle or that a corridor gets narrower at one end. This constraint allows reshaping to happen without these undesirable side effects. Given this choice, the main "degree of freedom" for the user will be the ability to make edges (of walls etc.) longer or shorter, without changing the angles. At the same time we restrict deformation for *small* edges, since they typically correspond to finer details, which we want to preserve. For instance, decorations around a window or a door are likely to be very small edges if

compared to the length of a room. We want these edges to remain unchanged or receive less deformation if we enlarge this wall.

We have designed and implemented a complete system to achieve these goals. Our main contributions are thus:

- A novel approach for deforming and reshaping architectural meshes, based on separating angular and length constraints. We propose a linear formulation of the problem, solved as a least square minimization. Our approach allows interactive geometry reshape with intuitive results.
- A reshaping tool tightly coupling geometry and texture. While the user manipulates the geometry, texture features are updated so as to maintain their visual appearance while following the deformations; to our knowledge such coupling has not been done before.
- The use of directional autosimilarity to identify regions of a texture which can be appropriately deformed, while keeping structured parts rigid during interactive textured geometry reshape.
- An interactive method to re-introduce texture detail in stretched regions, based on detail extraction, tiling and a realtime rendering solution.

We have implemented the above ideas in an interactive system. As we will show in our results and applications (Section 3.5), our new method provides an interactive approach to complex reshaping of textured models (see Figure 3.1).

3.3 Geometry reshape

Our goal is to reshape an architectural model while retaining its characteristic features. As explained above, we consider that preserving angles is the most important constraint to impose for the class of models under consideration.

In our formulation, vertices are either *variables*, in which case the solution to our system determines their position, or they are *constrained*. Constrained vertices can be manipulated by the user - in which case their position is attached to the mouse ("handles") - or they remain fixed. In Figure 3.2 and the video [1] constrained vertices are shown in green.
Our approach has three major steps:

- The input model is loaded and organized into a simple constraint graph (Section 3.3.1).
- A set of constraints is defined capturing the relationships between vertices, walls and edges (Section 3.3.2). In addition to preserving angles, we maintain short edge length, vertex/face contacts and avoid edge flips.

[1] located here *http://www-sop.inria.fr/members/Marcio.Cabral/thesis/*

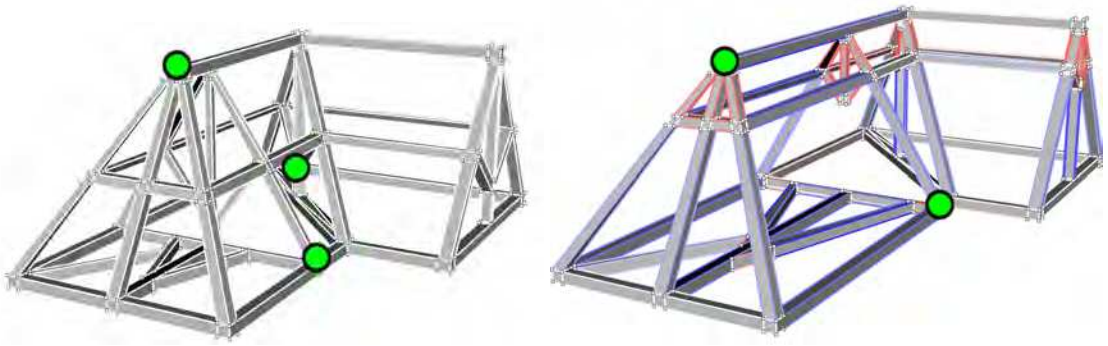


Figure 3.2: *Left:* The original model. *Right:* The final result. Elongated edges are highlighted in blue and compressed edges in red. This motion is achieved by moving the handle up (green point in the center).

- A solver recomputes vertex positions from the user controlled handles, while attempting to preserve the constraints (Section 3.3.3).

The two first steps are a pre-process. In contrast, the solver is used at run-time, during user manipulation of the model. To enable interactivity, we rely on a simple solver and allow it to fail or refuse user input if this leads to degeneracies (ie., collapsing edges or collisions). We argue this behavior is reasonable since the user has full freedom to assign new handles and guide the solver in avoiding degeneracies, thus obtaining the desired result. Nevertheless, we propose simple mechanisms to help the user in this task (see Section 3.3.4).

3.3.1 Input and graph construction

We first load the model and create a corresponding *constraint graph*. We assume that the input is a textured mesh, in the form of an indexed face set. We expect triangles to be grouped in textured surfaces, which we refer to as *surfaces*. Triangles within a same surface share vertices; a shared edge implies that vertices share 3D and UV (texture) coordinates. The nodes of the constraint graph are the vertices of the model. Note however that two co-located vertices in different textured surfaces will share a same graph node.

We distinguish three types of edges within a textured surface. *Contour edges* are used by a single triangle in the textured surfaces, while *angle edges* are shared by two non co-planar triangles. Both edge types are added to the graph. Lastly, *flat edges*, shared by two co-planar triangles, are ignored. In addition, textured surface contours must be well formed: i.e., a contour is formed by following the ring of adjacent contour edges. We also support holes.

Finally, we identify connected components of the graph. In the examples we show here, the number of connected components is typically low (most often around 3 or 4).

3.3.2 Constraints

We use the constraint graph to express the properties to be preserved whenever handles are being moved. We thus formulate a system of equations expressing constraints that must be either strictly enforced or minimized. In what follows, we will present "strict" and "soft" constraints. Table 3.2 and table 3.3 provide a summary. We will explain how these are actually used in Section 3.3.3.

Notations In what follows v_i is a vertex and E_{ki} an edge between vertices v_i and v_k. N_i is the set of neighboring vertices of vertex i and $|N_i|$ the size of this set. We designate variables using a "tilde" symbol. Hence, $\tilde{v}_i$ is the unknown next position of vertex i, while v_i simply refers to its initial position. We note $u_{ki} = \frac{v_i - v_k}{||v_i - v_k||}$ the normalized direction of edge E_{ki}, while $l_{ki} = ||v_i - v_k||$ is its length. Please see Figure 3.3 for more details.

Note that the edge direction u_{ki} and length l_{ki} are fixed with respect to variables since they are computed on the initial graph; they are thus constants in the system being solved (Section 3.3.3). Finally $v_{T_i^j}$ is the i-th vertex of triangle j, and n_{T^j} the normal to triangle j. Please refer to Table 3.1 for a summarized list of notations.

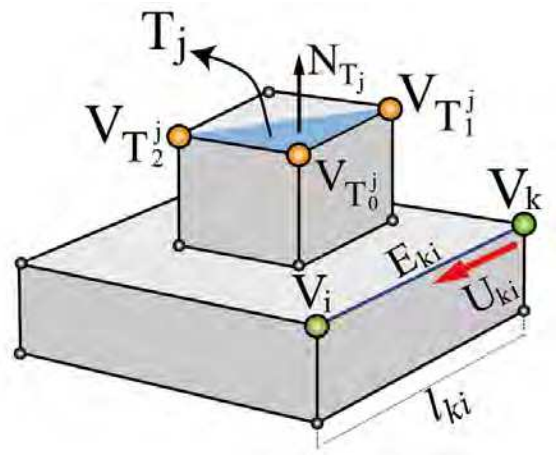


Figure 3.3: Notations used.

Edge direction constraints The most important constraint is to preserve angles between planar faces. Rather than working directly on faces, we equivalently preserve edge directions. For each vertex i we derive the following equation:

$$|N_i|\tilde{v}_i - \sum_{k=0}^{|N_i|-1} (\tilde{v}_k + (u_{ki} \cdot (\tilde{v}_i - \tilde{v}_k))u_{ki}) = 0 \tag{3.1}$$

Intuitively, this simply states that from any neighboring vertex k we can come back to vertex i by adding the appropriate length in the direction of E_{ki}. Note that u_{ki} is constant and computed on the initial mesh, while $\tilde{v}_i$ and $\tilde{v}_k$ are variables.

An important property of this equation is that it does not restrict the edge length but only the alignment of the vertices. However, vertices are free to move so the solver might

Notations	
v_i	initial position of vertex
E_{ki}	edge between vertices v_i and v_k
N_i	set of neighboring vertices of vertex i
$\|N_i\|$	size of the set N_i
$\tilde{v}_i$	vertices as variables
$u_{ki} = \frac{v_i - v_k}{\|v_i - v_k\|}$	normalized direction of edge E_{ki}
$l_{ki} = \|v_i - v_k\|$	length of normalized edge u_{ki}
$v_{T_i^j}$	i-th vertex of triangle j
n_{T^j}	normal to triangle j

Table 3.1: Notations used in this Chapter.

have to compromise and change the direction of some edges (see Figure 3.5(right)).

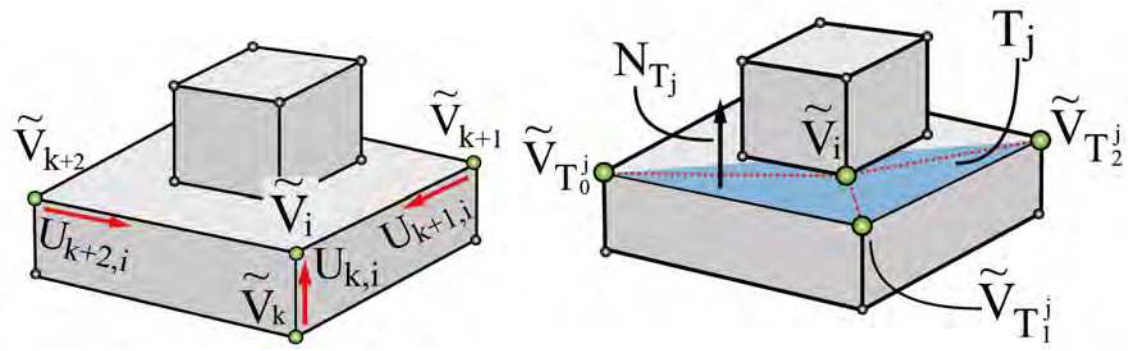


Figure 3.4: Edge direction constraints (left) and contact constraints (right).

Edge length preservation It is also desirable for the edges to keep their original lengths: The result of our geometry reshape will thus be as close as possible to the initial mesh.

We express this with an *edge stress* term S. We thus seek to minimize the following term for each edge:

$$S_{ki} = w_{ki} \ (u_{ki} \cdot (\tilde{v}_i - \tilde{v}_k) - l_{ki}) \tag{3.2}$$

The scalar term w_{ki} controls how well the edge length must be preserved, or edge *stiffness*. We typically give a higher importance to small edges compared to large edges, and consider edges shorter than a user defined threshold as rigid.

We compute w_{ki} as:

$$w_{ki} = w_{small} + (w_{long} - w_{small}) \left(\frac{l_{ki} - l_{min}}{l_{max} - l_{min}} \right) \tag{3.3}$$

where l_{max} is the largest edge length, l_{min} the threshold below which edges are consid-

ered rigid, and w_{small}, w_{long} control the overall edge stiffness. We use $w_{small} = 10^{-3}$ and $w_{long} = 10^{-5}$. Note that w_{small} is larger than w_{long} to make small edges more rigid. For rigid edges, we set $w_{ki} = 1$

We illustrate direction constraints and edge length preservation in Figure 3.2. Our approach preserves angles and the length of short edges, while allowing large edges to be either shortened or lengthened.

Additionally, all edge lengths must remain positive during reshaping, which is expressed with the following strict constraint per-edge:

$$u_{ki} \cdot (\tilde{v}_i - \tilde{v}_k) > 0 \tag{3.4}$$

Contacts Often structures lie on other surfaces: Pillars, doors, windows, etc. Using only edge direction constraints, we would not be able to capture these relationships.

Within a connected component, coplanarity is already captured by edge direction constraints. However, two disconnected components do not share triangles: Contact relationships are not captured by the previous equations. We check whether vertices of one component are on a triangle of another component. When this happens we add one "strict" constraint for co-planarity, simply using the vertex and the first vertex of the triangle containing it:

$$(\tilde{v}_i - \tilde{v}_{T_0^j}) \cdot n_{T^j} = 0 \tag{3.5}$$

where n_{T^j} is the normal to triangle T^j, $\tilde{v}_i$ and $\tilde{v}_{T_0^j}$ belong to different connected components. We also add three "soft" constraints to encourage the vertex to stay at the same distance from the triangle vertices:

$$\left((\tilde{v}_i - \tilde{v}_{T_k^j}) - (v_i - v_{T_k^j})\right) = 0,\ k\ =\ 0..2 \tag{3.6}$$

These are mandatory otherwise the system is under-constrained, since co-planarity does not tells us "where" the vertex should be on the plane. Please see Figure 3.4 for more details.

Edge groups Architectural models contain many implicit constraints. For instance, the height of doors is typically the same throughout a building. In general, inferring this type of constraint from input geometry is a difficult problem and depends on the semantics of the model.

We do not infer semantic information; instead we use some easily identifiable properties of the model, notably groups of *similar* edges. This provides correct default behavior for most cases, and it may be switched off by the user at any time. Specifically, we define an *edge group* as a set of edges having similar direction, similar length, and being spatially close to each others. This is achieved using a simple clustering approach.

For each edge group we add constraints stating that edges must keep similar length. The first edge of the group is used as reference. For each other edge we write the equation stating that its length must be equal to the length of the first edge, using a *group error*

G_E:

$$G_E = w_g \left(\left(\sum_{E_{ji} \in G} u_{ji} \cdot (\tilde{v}_i - \tilde{v}_j) \right) - |G| \; u_{10} \cdot (\tilde{v_0} - \tilde{v_1}) \right) \tag{3.7}$$

We use v_1, v_0 to denote the vertices of the first edge of group G, and u_{10} for its direction. The edge between v_j, v_i (E_{ji}) is within the same group. The group has $|G|$ edges. We use $w_g = 0.1$. An example of the effect of edge groups is illustrated Figure 3.5.

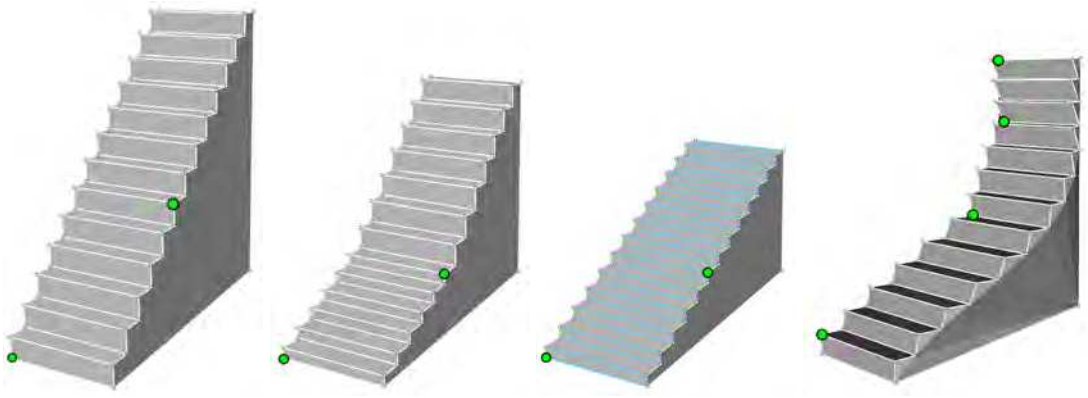


Figure 3.5: *Left:* A model of stairs. *Middle-left:* Result obtained if we do not use edge groups and use the handles shown in green. Unwanted distortion appears. *Middle-right:* Using our edge group approach, we achieve uniform reshape of the stairs. *Right:* The solver changed edge directions in order to achieve the motion required by the user.

Edge direction	$\lvert N_i \rvert \tilde{v}_i - \sum_{k=0}^{\lvert N_i \rvert - 1} (\tilde{v_k} + (u_{ki} \cdot (\tilde{v}_i - \tilde{v_k}))u_{ki}) = 0$
Edge > 0	$u_{ki} \cdot (\tilde{v}_i - \tilde{v_k}) > 0$
Co-planarity	$(\tilde{v}_i - \tilde{v}_{T_0^j}) \cdot n_{T^j} = 0$

Table 3.2: Summary of desired strict constraints.

Edge length	$w_{ki} \;\; (u_{ki} \cdot (\tilde{v}_i - \tilde{v_k}) - l_{ki})$
Contact positions	$\left((\tilde{v}_i - \tilde{v}_{T_k^j}) - (v_i - v_{T_k^j}) \right), \; k \; = \; 0..2$
Edge groups	$w_g \;\; \left(\left(\sum_{E_{ji} \in G} u_{ji} \cdot (\tilde{v}_i - \tilde{v}_j) \right) - \lvert G \rvert \; u_{10} \cdot (\tilde{v_0} - \tilde{v_1}) \right)$

Table 3.3: Summary of soft constraints to be minimized.

3.3.3 Solver

As discussed above, we have three types of constraints: inequality, equalities and terms to that should be minimized. For the first, an accurate solution would require linear programming, thus sacrificing interactivity. We discuss our alternative solution in Section 3.3.4. For the remaining constraints a typical way to solve is to describe the problem

as an least squares minimization and solve with efficient linear system solvers enabling interactivity. We can write this more formally as:

$$x' = argmin_x ||Ax - b||^2 \tag{3.8}$$

where x is the vector of positions $\tilde{v}_i$ of all the vertices in the model except for the handle(s), whose position is given by user input and the fixed vertices which are not affected by the solution. A is the matrix defined by the constraint equations.

A typical way to enforce equality constraints in a least square optimization is to include them in the system with a large weighting term [Loan, 1985]. This can lead to small inaccuracies, which are tolerable in our context.

In this system, we weight the minimization constraints with appropriate values w_i. In particular we use weight $w_g = 10^{-2}$ for Eq. 3.7, $w_c = 10^{-5}$ for Eq. 3.6; for Eq. 3.2 weighting is embedded in w_{ki}. In practice, these weights work across most of our meshes. Only l_{min} – fixing the edge length under which $w_{ki} = 1$ (Section 3.3.2) – needs to be adapted since it strongly impacts the reshaping behavior.

We compute the normal equations, pre-factor the sparse matrix $A^T A$ and solve very efficiently at run-time, using the Cholesky factorization of the TAUCS library [Toledo et al., 2003]. Such approaches have been used for interactive mesh manipulation [Botsch et al., 2005, Botsch and Sorkine, 2008].

3.3.4 Edge flips and User Constraints

To deal with the non negative edge inequality (Eq. 3.4), we exploit the interactive aspect of the user manipulations: As the user drags handles only small motions occur. After every step we verify that no edge is collapsing. If an edge becomes too small we artificially increase the length l_{ki} (Eq. 3.2), making it less likely to collapse. Since this only affects the constants in the system, it does not affect the interactive performance of our approach. If an edge does collapse we refuse the last motion and rollback to the previous position.

Note that linear programming *could* be used to enforce this inequality, for instance by incrementally updating constraints during interactive manipulation [Borning et al., 1997]. In practice interactive feedback makes it easy to detect and avoid degeneracies during manipulation: It seemed unnecessary to resort on more complex solvers in our context.

User-defined additional constraints In some cases (e.g., Figure 3.7(left)), it may be necessary to add additional constraints. Our system provides a simple mechanism to link two vertices, resulting in an additional constraint. An example is shown in Figure 3.7(right).

3.3.5 Limitations

One of the main limitations is that we ignore surface interpenetration, which can result in internal structures going through walls.

The behavior of the reshape depends on the chosen handles, and an unfortunate choice may result in undesired motions. Thankfully, given the interactive nature of our

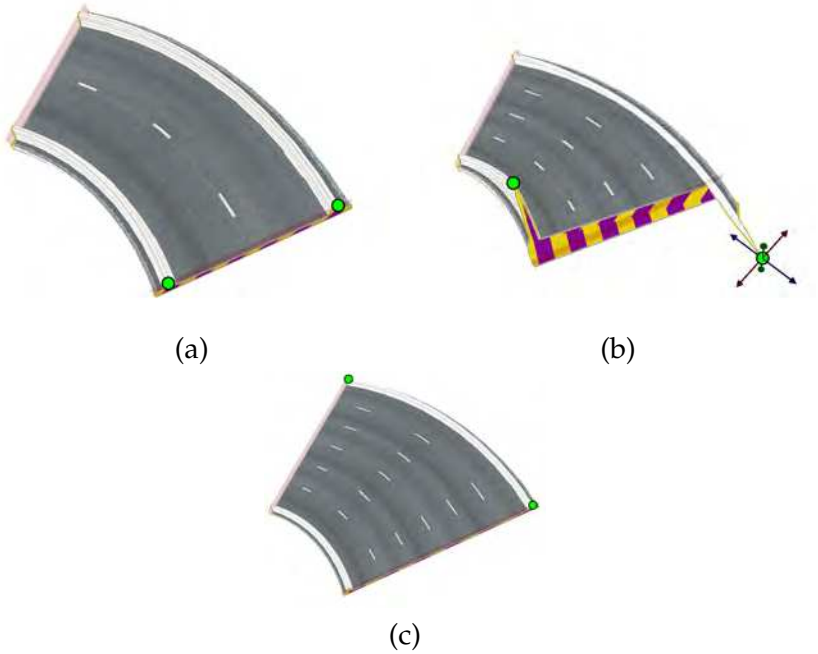


Figure 3.6: *(a) and (b):* The choice of handles results in undesired motions. *(c):* Thanks to interactive feedback the user easily chooses a better set of handles and achieves the desired reshape.

approach the user can quickly correct for such cases, as illustrated in Figure 3.6. Note that the system behaves reasonably even if the user ask for deformations where edge direction cannot be preserved, as illustrated in Figure 3.5-*Right*.

Many of these limitations could be addressed by adding constraints; however, we cannot infer all these constraints since they often depend on model semantics.

3.4 Texture reshape

The vast majority of interactive applications enhance the appearance of objects with texture mapping. But surprisingly most existing geometry editing approaches do not provide special treatment of textures during deformation. For single-material models texture synthesis from example methods could be used to automatically obtain a new texture map [Wei and Levoy, 2000]. However, on architectural models textures are often much richer, contain many architectural elements: A door frame, various decorative elements and other wall details. It is therefore very important to retain and preserve the appearance of this information when the user changes the geometry of a mesh, even though the only information we have are the texture pixels.

We thus propose a new approach to resize texture maps targeted at architectural environments, inspired by ideas from image warping [Wang et al., 2008], image com-

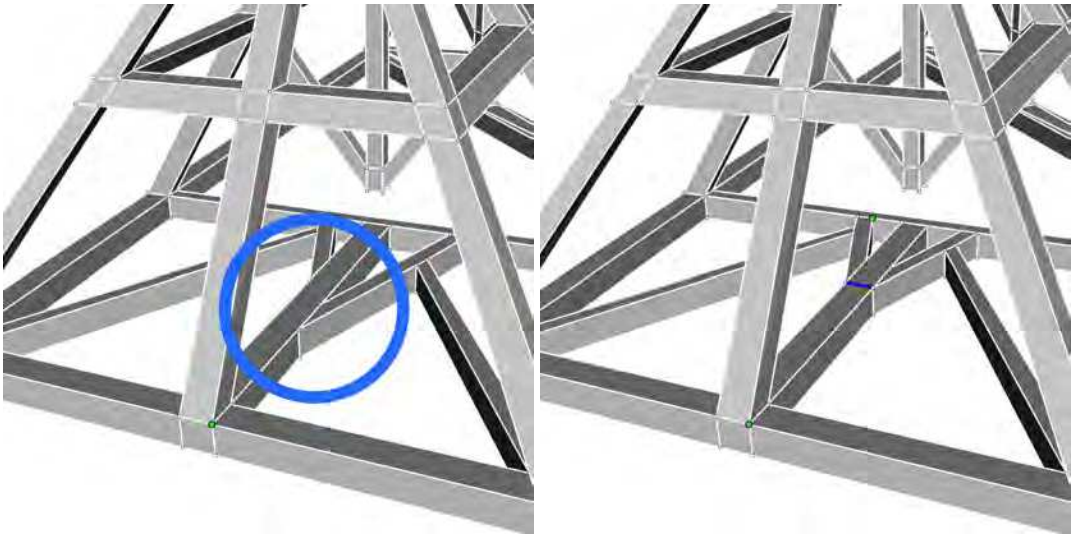


Figure 3.7: *Left:* parts of the roof frame (in the blue circle) should move together; however they do not share edges or vertices. *Right:* The user can add an a link between appropriate vertices, resulting in appropriate deformation.

pletion [Drori et al., 2003] and texture synthesis from example [Efros and Leung, 1999]. It is designed to perform efficiently during interactive manipulation of the geometry, while providing high quality results. Our work is most similar to the approach of [Wang et al., 2008] in that it deforms the texture to concentrate stretch in some particular areas. However we provide a different formulation based on the new notion of *directional autosimilarity* (Section 3.4.2).

In the following we use *texture map* to designate the image mapped onto a surface and *stochastic texture* to designate areas of the image having homogeneous content.

3.4.1 Overview

Consider the texture in Figure 3.8(a); if we enlarge the geometry with no special treatment, we get the result of Figure 3.8(b)-*Left*. We want to get the result shown in Figure 3.8(b)-*Right*, where structured parts of the texture are preserved, and stretching is focused on the stochastic texture regions.

To do this we use *directional autosimilarity*, which measures whether a texture region remains similar to itself if slightly translated along a given direction. We typically compute this measure along the two main image axes. Our autosimilarity measure often takes large values *across* edges and low values in the direction *parallel* to edges (see Figure 3.9). In areas of stochastic content it exhibits a low value - compared to edges - in all directions. This provides an effective measure to detect regions of similar homogeneous content.

We encode this measure in an autosimilarity map (see Figure 3.9). We then warp the texture using a grid, shown in Figure 3.10(a). The goal is to first replicate edge-length changes from geometry reshape to texture space, and then to deform the grid guided by

(a) (b)

Figure 3.8: (a) The original textured polygon. (b) *Left:* The polygon has been stretched without reshape. (b) *Right:* The desired result, where features are preserved.

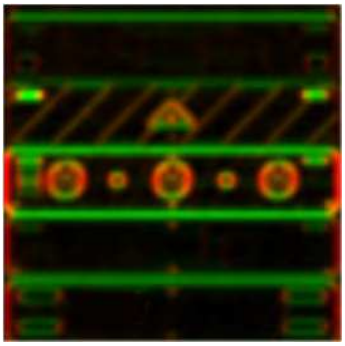
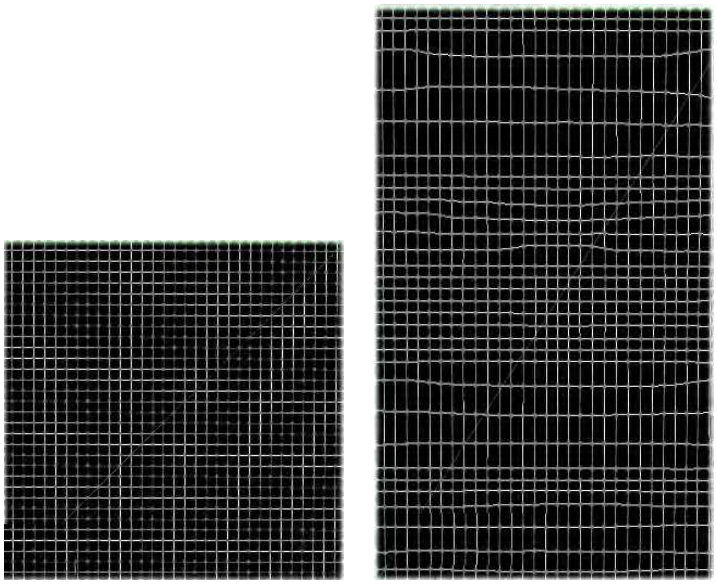


Figure 3.9: Directional autosimilarity map.

the autosimilarity measure, giving the result shown in Figure 3.10(b). We use the result to generate a map of distorted texture coordinates for the stretched polygon.

As we can see in Figure 3.13(a), details in stretched regions are now blurred. We want to reintroduce detail by finding and using appropriate regions in the original texture. We do this by segmenting the texture into rigid/stochastic regions (Figure 3.12(a) and then extracting *detail tiles* Figure 3.12(b). These are regions of stochastic content which can be tiled to re-introduce detail into stretched regions. We introduce an appropriate rendering technique that can use this information, giving the final result shown in Figure 3.13(b).



(a) (b)

Figure 3.10: (a) The grid on the original texture (Figure 3.8(a)). (b) The deformed grid, representing Figure 3.8(b)-*Right*. Note how rigid features are preserved.

3.4.2 Directional texture autosimilarity

Architectural texture maps often contain a mix of structural elements and areas of homogeneous, stochastic texture content. For instance, in the image of a brick wall, the separation between the bricks can be considered as structure, while the interior of each brick is a texture in the statistical sense [Wei and Levoy, 2000]. That is, for a given pixel neighborhood in these areas other, similar pixel neighborhoods can be found in its vicinity. Since we know these textured areas are easier for us to reproduce, our goal will be to detect them so as to focus stretch onto them.

When the geometry is deforming, the edges of the textured polygon will change length (Section 3.3.2). Often these changes will be anisotropic, for instance when a wall is widened while keeping its height. The structural elements contained in textures are often parallel or orthogonal to the ground plane - a plinth, a door frame or brick walls are good examples. These structural elements are likely to be very auto-similar in a given direction along which they can be considered as a texture. This implies that stretch is not likely to be visible if applied in this same direction - however it must be prevented in other directions.

In order to exploit these degrees of freedom, we introduce the notion of *directional autosimilarity*. We start from a given set of directions, typically the vertical and horizontal texture space axes. For each direction we compute an error map indicating whether each pixel location can be considered as an autosimilar texture *in the given direction*. This is done by comparing small neighborhoods along a 1D line centered on the pixel and oriented parallel to the current direction.

More formally, we define an error at a given pixel p expressing how far we are from autosimilarity:

$$T_{\vec{dir}}(p) = \frac{1}{2\Delta} \sum_{\substack{q=p-\Delta\vec{dir} \\ q\neq p}}^{p+\Delta\vec{dir}} ||N(q) - N(p)||^2 \tag{3.9}$$

where $\vec{dir}$ is the considered direction, Δ controls how far around the pixel we search and $N(p)$ is a pixel neighborhood around p. We typically use 5×5 neighborhoods and a value of $\Delta = 10$. The result for two directions can be seen in Figure 3.8(d), encoded as red and green.

We next use this information to deform the texture map and focus stretch in textured areas.

3.4.3 Deformation grid and Constraints

We now exploit directional autosimilarity maps to deform the texture map. We achieve this using a *deformation grid* overlaid on the texture map. We intersect the grid with the contour of the surface polygon in texture space, using a DDA approach for robustness. This is illustrated Figure 3.10. The spacing of the grid is chosen to be as large as possible while still preserving the features of the texture map. We typically use a spacing of $\frac{1}{32}$ in normalized texture space.

We call *nodes* the vertices of the grid. We distinguish three types of nodes: corner nodes, denoted c_i, boundary nodes, which lie on the edge of the polygon boundary but are not corners, denoted b_i, and interior nodes (Figure 3.10). We use g_i to denote any kind of grid node (interior, boundary or corner).

We deform this grid in two steps, first solving for the shape of the textured polygon (i.e., the corner nodes) and then for the boundary and interior nodes.

Grid corner constraints First, in order to determine the new shape of the polygon in texture space we replicate the changes in length of the world space polygon edges. We deform the texture space polygon using a gradient-based approach [Botsch and Sorkine, 2008]. Second, as we shall see, for efficiency we impose that boundary edge directions are maintained. Finally, we fix the position of one corner node to remove the translational degree of freedom of the system.

We use the same convention as for geometry reshape where $\tilde{c_i}$ denotes the (variable) position of corner i during reshape. For gradient-based deformation of the triangles we consider each corner c_i, $i = 0..2$ of each triangle T in the textured surface. We define the set of $N_T(c_i)$ of neighboring corners c_k, with $|N_T(c_i)|$ the size of this set. We thus have:

$$|N_T(c_i)|\tilde{c_i} - \sum_{c_k \in N_T(c_i)} [\tilde{c_k} + r_{ki}\,(c_i - c_k)] = 0 \tag{3.10}$$

where r_{ki} is the length change ratio of edge ki from the geometry reshape:

$$r_{ki} = \frac{|v_k - v_i|_{current}}{|v_k - v_i|_{initial}} \tag{3.11}$$

The interior node system, defined in the following section, depends on the direction of the polygon contour edges. To avoid having to re-factor the interior system during interaction, we constrain the edges of the contour to maintain their direction, using a 2D version of Eq. (3.1), replacing v_i by c_i.

Interior grid node constraints The first constraint concerns boundary nodes b_k, contained in the edge defined by corners c_i, c_j. We define $N_{c_ic_j}$ to be the normal direction to the edge defined by c_i, c_j. We require that the b_k remain on the edge, by imposing the following constraint:

$$\tilde{b_k} \cdot N_{c_jc_i} \;=\; \tilde{c_i} \cdot N_{c_jc_i} \tag{3.12}$$

Note that thanks to the edge direction constraint described above, normal $N_{c_jc_i}$ will never change during interactive manipulation.

For interior nodes we want to achieve a deformation which will preserve rigid regions and concentrate stretching in stochastic regions. For each edge defined by grid nodes g_i, g_j and for each deformation direction $\vec{dir}$ (there are typically two), we define energy e_{ij} to be minimized:

$$e_{ij} \;=\; \left((\tilde{g_i} \;-\; \tilde{g_j}) \cdot \vec{dir} \;-\; (g_i \;-\; g_j) \cdot \vec{dir}\right) s_{ij} \tag{3.13}$$

We define the stiffness weight s_{ij} using the autosimiliarity error map $T_{\vec{dir}}$ in direction $\vec{dir}$ as:

$$s_{ij} = s_{min} + T_{\vec{dir}}\left(\frac{v_i + v_j}{2}\right) \;\left(1 + s_{align}\left(1 - |(v_j - v_i) \cdot \vec{dir}|\right)\right) \tag{3.14}$$

where T is accessed with a 2D coordinate in texture map space ($\frac{v_i+v_j}{2}$ represents the proper texture coordinate that access its respective value in the error map, between interior nodes v_i and v_j), s_{min} is the stiffness of texture areas, s_{align} controls how much orthogonal alignment must be preserved. Note that we equalize luminance in all textures, and we assume that lighting information is not included.

3.4.4 Online Reshape Solver

To achieve online texture reshape we use an approach analogous to that used for geometry (Section 3.3.3). We express all constraints as a linear system and weight equations with respect to their importance. We use a direct least square solver (sparse Cholesky factorization [Toledo et al., 2003]). For fast interactive manipulation we pre-factor the matrices for both the contour and interior system. During interactive manipulation, only the vector b needs to be recomputed and $A^T b$ updated, where A contains the constraint equations.

An example is shown in Figure 3.8 where the textured surface has been stretched

significantly. For efficient storage and display, we manipulate and store *uv* coordinates rather than the texture itself. The deformation is thus coded in these coordinates, which are rendered into a render target and stored as a texture of the same resolution as the original texture map. We call this the *distortion map*. During rendering we thus use one level of indirection to access the original texture.

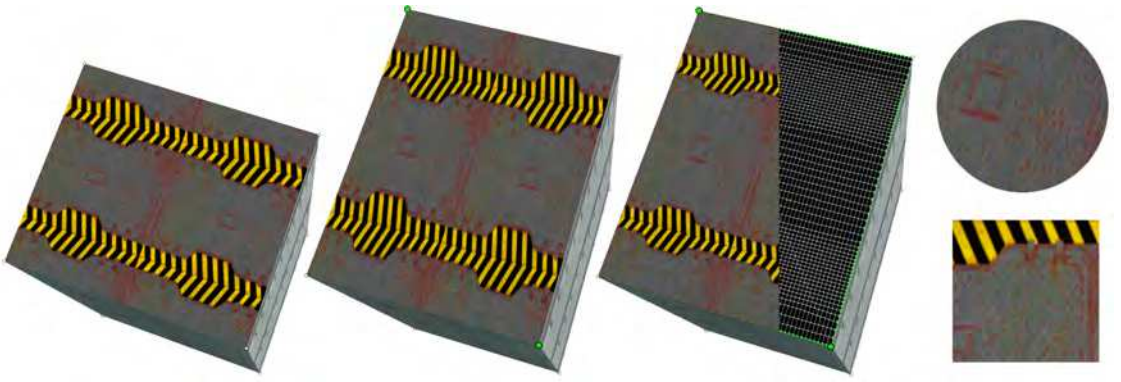


Figure 3.11: *Left:* the deformed surface without texture reshape *Middle:* our texture reshape focuses stretch in stochastic areas, preserving structural elements. *Right:* the top circle shows a close-up of the reshaped region, part of the original texture is shown below. Notice how fine details are stretched.

Regions where stretching has occurred will of course be blurred, since the texture hardware interpolation will be used. This is illustrated in Figure 3.8(b). To alleviate this problem, we present a new approach to reintroduce details for the reshaped texture.

For toroidal textures we want to have an integer number of tiles covering the polygon. We solve for the corners, then extract a bounding square in texture space. This allows us to compute the number of repetitions in UV and the amount of distortion for a single tile. This should not be confused with the detail tiles described below. We then solve as usual for the interior nodes. Windows on the right of Figure 3.15 are repeated in this manner.

3.4.5 Reintroducing detail

Stretching high-frequency texture regions results in undesirable distortion artifacts. To avoid this issue, we separate texture detail from the image and identify each texture region of the texture map. For each region, we extract a representative rectangular tile from the detail layer. When stretch occurs we only apply the deformation to the base image, and reintroduce details. However, instead of stretching the details we simply repeat the representative tile, hence preserving the frequency content. As shown Figure 3.8(b)-*Right* this significantly improves the quality of the result at little cost. The following paragraphs explain this approach in more detail.

The first step in recovering detail is to identify which regions within the texture can be used as a model to reintroduce details. We first perform a binary segmentation of the texture into stochastic and structural areas following our autosimilarity measure. We

then identify stochastic regions and "grow" a rectangular tile within each such region. These tiles will be used to reintroduce detail during rendering. Figures 3.12 and 3.13 illustrates this process.

(a) (b)

Figure 3.12: (a) Texture segmentation into rigid/stochastic regions. (b) Detail tiles extracted from original texture (color rectangles)

(a) (b)

Figure 3.13: (a) Stretched texture: inset shows blurred detail. (b) Stretched texture: inset shows re-introduced detail, extracted from tiles. (Please zoom to see details)

Texture segmentation and Detail extraction To segment, we first merge the directional autosimilarity error maps (typically we have two, one along each axis), keeping the largest error for each pixel. This gives us a new map measuring how rigid each pixel is in the worst case direction. We apply a binary segmentation [Boykov and Jolly, 2001] on this map. After experimentally setting the parameters of the segmentation we use the same ones on all textures of our dataset. This works well in practice since our textures all have equalized luminance and do not contain lighting information.

The segmentation separates the texture in disconnected regions (see Figure 3.12(a)). We assume that each region corresponds to a stochastic texture with locality and stationary properties [Wei and Levoy, 2000] - note that while this is not to be expected in general, it works well in our cases since most architectural textures depict regions of homogeneous materials separated by structural elements.

Within each region we seek to extract a rectangular tile representative of the details, with the goal of using this to re-introduce detail when stretching occurs. Starting from a random seed we grow the rectangular tile in a "spiral-like" manner. We perform several iterations and keep the largest rectangle as our tile (Figure 3.12(b)). This naive search could be improved using more involved algorithms [Daniels et al., 1997]. Another example of this pipeline can be seen in Figure 3.14.

For each texture, we finally create a *tilemap* which indicates in each pixel the tile ID associated with its region. The tile ID indexes a small table which contains the upper and lower corner of each tile. If a pixel belongs to a rigid region, the tile ID is set to a special value (for example 0).

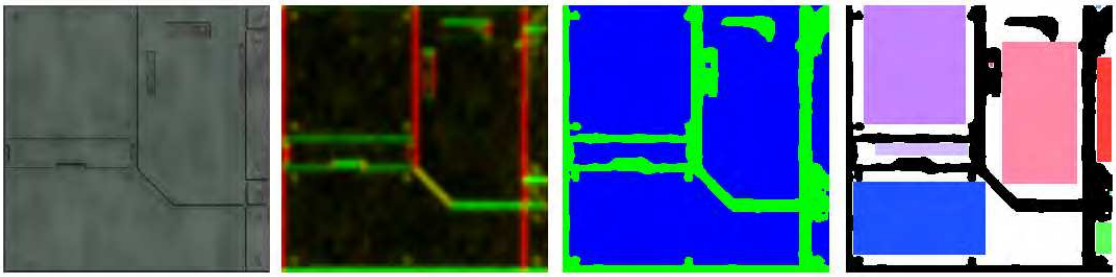


Figure 3.14: From left to right: original texture, directional autosimilarity error (red for the u direction and green for the v direction), resulting segmentation, the detail tile locations (each tile is shown with a different color). Texture Copyright ©Id Software, all rights reserved.

Online Detail sliding After solving the system for the polygon, we obtain new texture coordinates for each vertex. During rendering of the final scene, these per-vertex texture coordinates are interpolated by the rasterizer and passed over to the fragment shader. We use these coordinates to access the deformation map (see Section 3.4.4), retrieving the coordinates computed by the deformation grid. Directly accessing the original texture with these coordinates produces blurred details.

Instead, we perform a lookup in the tilemap to determine whether the pixel has an associated tile. If yes, we perform a lookup in the texture from the tile, using a modulo

operation to let the tile cycle in heavily stretched regions.

Directly using colors from the tiles would produce artifacts at the boundary of stretched regions. Instead, we separate the original image into a base and a detail layer using a bilateral filter [Tomasi and Manduchi, 1998]. We only use the tiles on the detail layer, the base layer being stretched as previously. However in the base layer stretched areas contain no detail.

Finally, to avoid modifying the original texture when no stretch is applied we gracefully transition from the original detail layer to the tiled details using a measure of the local stretch. This is easily computed from the deformation grid.
The shader pseudo-code below summarizes these operations:

```
UV = distortionMap(p); // p contains the distorted coordinates
tile = tilemap(UV); // UV accesses tilemap & original texture
if( tile != 0 ) { // pixel isnt rigid
    detailTile = lookup(tileMap, p modulo tileSize);
    color = base(UV) + lerp(detail(UV), detailTile, stretch);
}
else
    color = base(UV) + detail(UV);
```

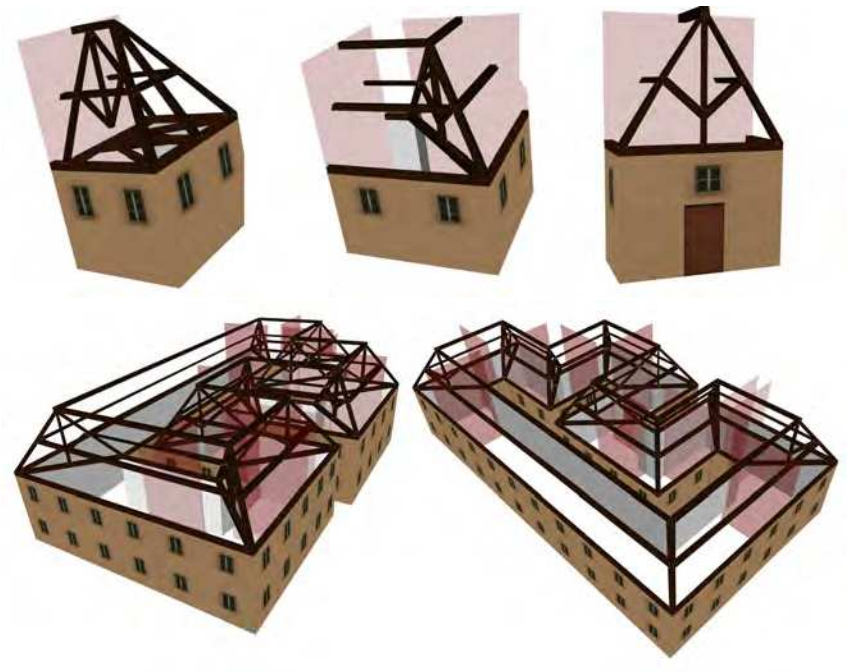


Figure 3.15: An example of houses with a complex roof frame using three pieces shown on the first row. We show two views of two different variants of the resulting house.

The result of this operation is that detail is added in the stretched regions. In addition,

the modulo operation during the lookup of the tilemap will result in tiles being repeated when necessary. We illustrate these operations in Figures 3.13 and 3.12.

3.5 Results and Applications

Piece	vars	nnz A	nnz A^tA	nodes	edges	eqns	factor	solve
	114	2136	4284	38	60	217	4 ms	1 ms
	1314	30294	60936	438	760	3874	2155 ms	10 ms
	258	5604	13068	86	176	517	44 ms	1 ms
	909	15588	30681	303	530	1555	175 ms	3 ms

Table 3.4: Performance measure for some of the meshes utilized for tests. *nnz* stands for 'number of non-zero entries', *vars* for variables and *eqns* for equations. All results shown here were achieved using a Dual Xeon 1.67GHz machine with 8GB RAM.

The geometry (Section 3.3) and texture (Section 3.4) reshaping algorithms constitute the core of our approach. We next present present three examples using a "mesh piece modeling" application. To truly appreciate our results, please see the accompanying video located in the online Appendix *http://www-sop.inria.fr/members/Marcio.Cabral/thesis/*.

Texture and geometry reshape can be used together to form a powerful tool for the edition and creation of architectural scenes. The user can create complex models based on a small set of initial pieces, which are either specifically built for this purpose, or are easily available (e.g., the pieces of a game level in our examples).

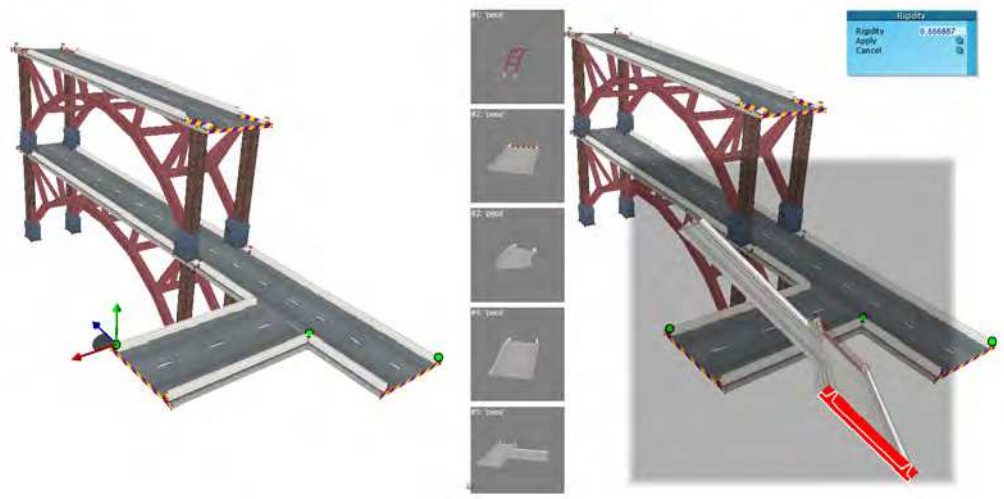
We assume that each piece has "portals" associated at each extremity, and that portals are compatible, i.e., have the same number of vertices. This is for example the case with the game level pieces we extracted.

The user deforms and connects pieces to achieve the desired result, for example as in Figure 3.1 for roads, or Figure 3.17 for game levels. When modifications are made on a given piece, deformations and reshape are appropriately propagated along the chain of pieces. We can limit the propagation length during interactive manipulation; when the user releases the handles, the propagation is completed. We also cache the matrix factorizations to accelerate updates. Please see the video for example usage.

Mesh pieces and Interactive tool

We have developed an interactive tool providing a simple, yet intuitive interface to assemble and reshape pieces. The interface allows the user to connect pieces (by their portals) and reshape them. To connect two pieces, the user simply clicks on a piece portal, which brings up a list of possible pieces with matching portals that can be connected

to the selected piece (left hand side of the interface - see Figure 3.16(b)). Reshaping is possible by clicking on any vertex: when one vertex is selected as a handle, an axis-widget is shown, allowing the user to move the vertex (see Figure 3.16(a)). Changes made to this vertex propagate through all pieces affected. Besides moving vertices, the user can also add constraints between edges (rigidity values can be added interactively in a pop-up menu - see Figure 3.16(b) top right).



(a) (b)

Figure 3.16: Our interactive tool allowing the user to assemble various pieces into complex models.

Applications: House/Road Building and Doom Levels We first show an example of house building in Figure 3.15. We then show an example with road blocks in Figure 3.1. As we can see, quite complex road structures, with bridges, many-lane highways, overpasses etc. can be built from a small number of initial pieces. Figure 3.17, shows game level pieces and the construction of a new modified level using our approach.

In these examples, mesh pieces had between 200-1400 vertices (450-4000 equations), creating the matrix takes from 40-800ms (unoptimized), and factorization takes between 20ms and 2.5s (for the bridge piece). For textures factorization takes 20-330ms with a maximum of almost 5000 equations.

3.6 Discussions

We have already discussed limitations specific to geometry reshape in Section 3.3.5. In terms of other limitations, we currently assume that the models are well formed in textured indexed face sets. We have found this to be a reasonable assumption for example with the game levels of "Doom" we tested here. For the custom-built pieces we also show here, no particular modifications were required in the modeler's workflow to pro-

Figure 3.17: Examples of our Doom level construction. We show the level building blocks and two views of a final construction made in tens of seconds. Note that these pieces do not trivially connect by construction. Textures and models from the game Doom III™. ©Id Software, all rights reserved.

duce models we could use. Nonetheless, it may be desirable to re-use model pieces which have some inconsistencies in terms of connectivity information, for instance data from 3D scanners. We did some initial experimentation with such pieces, and we found that a simple voxelization approach to determine corners was sufficient to extract corners and connectivity. Evidently, a general robust solution is a very difficult problem with strong links to mesh repair techniques [Ju, 2004]. Since our method is primarily designed to be applied to models created by 3D modeling software we do not expect this to be be a major limitation.

Another limitation is our assumption that each connected component of texture detail contains one tile. Evidently this may not be the case; A multi-label segmentation should be performed to identify this and extract the appropriate tiles.

Finally, we currently assume that portals of pieces are "compatible". Correctly treating incompatible portals is a hard problem; a more general "geometry matching" approach needs to be developed. While methods to treat general meshes exist (e.g., [Funkhouser et al., 2004]), they are not necessarily adapted to architectural pieces. We

are actively pursuing this direction of research.

Adding feedback between texture rigidity constraints and geometric reshape should be relatively straightforward to add. This would be particularly helpful in cases when we cannot identify detail tiles for examples.

3.7 Conclusion

To conclude, we have introduced a new approach which simultaneously treats geometry and texture reshape of architectural or structured man-made models. Our new algorithm enables modeling by adapting and varying existing pieces, and chaining them together, achieving rapid construction of complex models.

Our main contributions are twofold: (i) A novel approach to deformation and reshape of architectural meshes, by separating angular and length constraints. We propose an efficient, interactive solution which we incorporate into a reshaping tool tightly coupling geometry and texture modifications. (ii) The use of directional autosimilarity identifying regions of a texture to deform during interactive reshape, while keeping structured parts rigid. We use this to present an interactive method to re-introduce texture detail in stretched regions, using detail extraction, tiling and a realtime rendering solution.

We have also demonstrated the power and utility of the approach in several examples: complex parts with conflicting deformation requirements, road and bridge construction based on a small set and complex new "Doom" (game) levels based on a set of existing pieces. We believe our approach is a key step towards powerful content creation tools for end users of virtual environments.

CHAPTER 4

3D Interaction For Geometry Editing

Contents

In the previous Chapter, we described a system for interactive mesh editing with automatic texture deformation. It provides an intuitive approach for mesh editing to naive users. We observed however, that its 2D interface was restrictive and hard to use (although easier to use than commercial modeling packages). Precise 3D selection and manipulation of vertices is desired (over 2D manipulation) to achieve editing goals due to its one-to-one mapping. Additionally, viewpoint changes in the 2D desktop interface are mimicked with a combination of mouse motion/mouse buttons events, which disorients the user from the target goal.

In this Chapter we extend our approach to try to solve these issues. We explore 3D interaction applied in an immersive setting. To evaluate our assumptions, we propose three different immersive settings for architectural editing and perform an initial pilot user study. The initial user study results that indicates that users appreciated the immersive nature of the system, and found interaction to be natural and pleasant. We finalize this Chapter with concluding remarks and directions for future work that can be explored.

4.1 Geometry editing in a 3D environment

Creating and manipulating 3D models is one of the hardest tasks in virtual environments. Traditional 3D modeling packages [Autodesk Maya, 2010, Blender, 2010] have a very high learning curve, and are inappropriate for fast conceptual design by lay users, especially in immersive settings. In the previous Chapter we presented our effort on providing an intuitive and easy-to-use interfaces to build simple 3D models. Recently several other research [Zeleznik et al., 1996, Igarashi et al., 2007] and commercial [Google Sketchup, 2010] efforts have also concentrated on this topic. These results and products are an indication of the widespread interest for such functionality. Intuitively, the creation of 3D models should be an ideal candidate for true 3D interfaces, since the content is inherently 3D. Such considerations have inspired futuristic artistic works [Branit, 2009] in which a 3D gestural interface is used to create a complex and visually rich environment.

Figure 4.1: A screenshot from the artistic work called "World Builder" by Bruce Branit [Branit, 2009]. In his vision, a person can build an entire city using simple hand gestures such as the one shown in this image.

Based on this intuition, in this part of the thesis we intend to study direct manipulation for basic 3D modeling in an immersive setting, in the context of conceptual or initial design for architecture.

Our 3D modeling system fills this gap of conceptual design which allows users to manipulate geometry and texture directly, and automatically updates the model to respond to these changes.

In addition to geometry and texture, we are interested in the manipulation of lighting, which is an important element in modeling and design, especially for architectural models.

The last element of interest is the interaction paradigm and in particular the "view" or "mode" in which the user performs the modeling task: egocentric (1:1 scale) immersive mode, exocentric mode or a combination of both (similar to world-in-miniature [Stoakley et al., 1995]) - see Figure 4.2. These modes are described in more detail in Section 4.3.3.

The goals of this study are to create a prototype system in an immersive setting, which supports casual modeling of textured geometry and lighting, and to investigate the relative merits of the different view modes with a pilot user study.

4.2 Related Work

Figure 4.2: A view of our system in "mixed mode" where a miniature and immersive version are combined.

In architectural design, users need tools for interaction which are simple yet provide enough flexibility to achieve the desired goal. Simple and generic manipulation techniques such as Go-Go [Poupyrev et al., 1996] may be limited for architectural design, since they are not practical when dealing with a large number of objects [Chen and Bowman, 2009]. Architectural design and geometric modeling have traditionally been privileged applications for virtual reality systems [Deisinger et al., 2000, Bullinger et al., 2010, Leigh and Johnson, 1996]. Our approach draws inspiration from these and others but addresses the problem from a different perspective. The closest previous work is the World-in-Miniature (WIM) system [Stoakley et al., 1995]. They presented a miniature version mixed with a scale 1:1 model (equivalent to our "mixed mode"), but for simple manipulations of viewpoint and of the virtual scene. The virtual world is shown to the user using an HMD system. Each hand holds a different device to interact with the Virtual World. The user's dominant hand holds a tennis ball adapted with a tracking sensor plus two buttons. This device is used for fine grained interaction with the scene, such as changing the position of a chair. The user's non-dominant hand holds a clipboard. The surface of the clipboard represents the floor of the virtual scene. When the user looks at the clipboard, he sees an aerial view of a miniature version of the Virtual World. This division of labor follows Guiard's [Guiard, 1987] guidelines for two-handed interaction. We also provide two-handed interaction, however, equally dividing labor between both hands so that they work in cooperation in a coordinated fashion [Guiard, 1987]. In contrast to [Stoakley et al., 1995], our system provides automatic assistance in the context of a "quick prototype" modeling or *conceptual design* application [Anderson et al., 2003]. More sophisticated modeling/design operations can thus be performed. In addition, we study lighting design in this context, using a simple "one-bounce" global illumination approximation to evaluate the consequences of light source changes in an immersive setting.

By targeting specific developments to the universal 3D tasks of manipulation, selection, system control, navigation and symbolic input [Bowman et al., 2002a] we can focus on providing a resourceful experience for architectural environment design. Our study is related to several aspects of manipulation of objects in large scale and immersive displays; these are usually accomplished with some degree of physical navigation and interaction.

The idea of automatically scaling an object and placing it within reach in front of the user has been proposed by [Mine et al., 1997b] and makes use of the concept of proprioception. [Pierce et al., 1999] make use of this concept in their Voodoo Dolls technique where a copy of an object that is far away is brought close to the user's hand for interaction. Our approach tries to build on these ideas for interaction.

We use a simple form of bimanual interaction for resizing (similar to that used on 2D touch-sensitive devices). In previous work, bimanual interfaces have been used in immersive settings [Balakrishnan and Kurtenbach, 1999]. We try to follow ideas which were initially observed by [Guiard, 1987] and others ([Cutler et al., 1997], [Zeleznik et al., 1997]) regarding the usage of both hands in our resizing gesture, which uses coordinated manipulation tasks (for example, our resizing gesture).

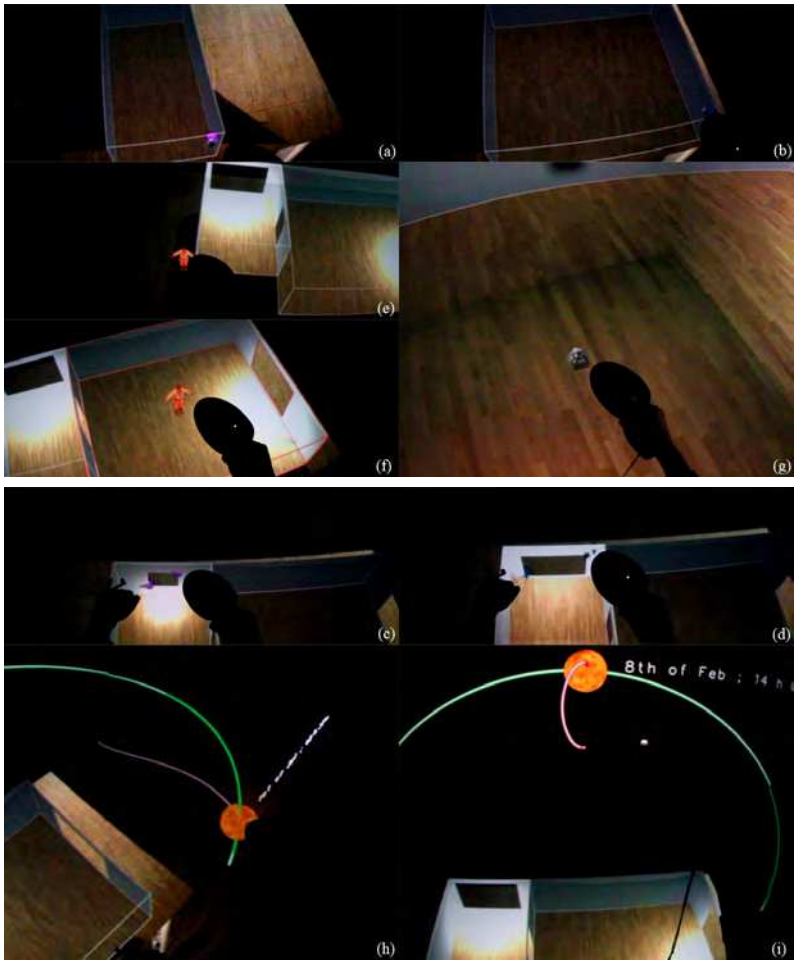


Figure 4.3: (a) The wall widget can be grabbed, resulting in a resized wall (b). (c) A window widget allows move and resize (bimanual) shown in (d). (e) The user selects the "human avatar" on the left and drags it to a location in the room (f). The user is then transported to that point in 1:1 scale (g). (h) The sun widget allows manipulation of the location of the sun (i), resulting in an update of one-bounce illumination.

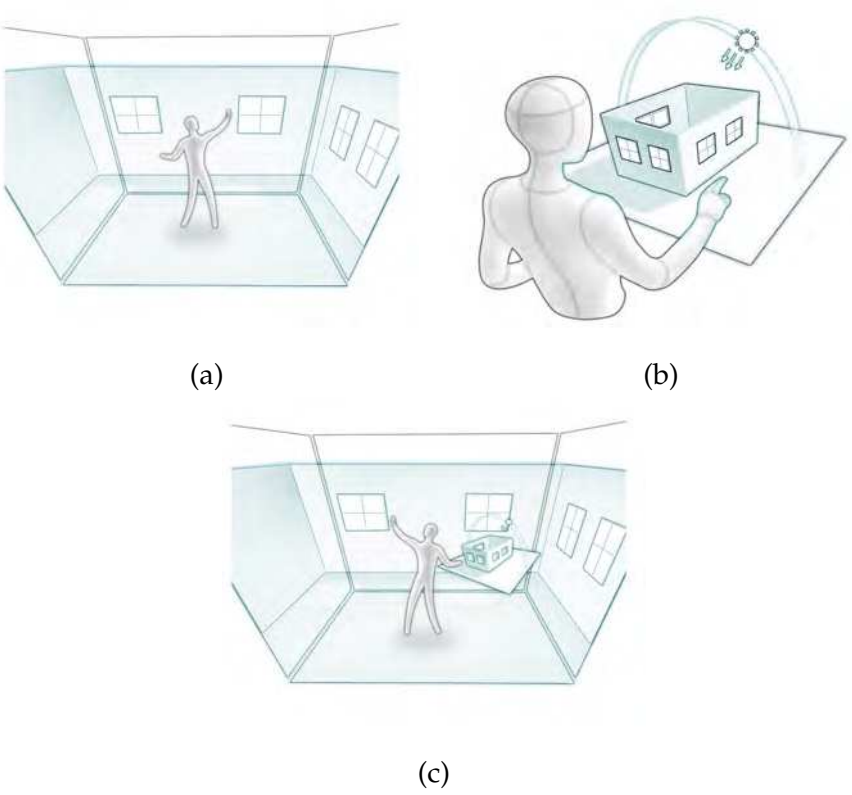
In our system, the user may switch between an immersive, scale 1:1 view and a world-in-miniature view, which implies a viewpoint change. Ware and Osborne proposed a technique [Ware and Osborne, 1990] called Eyeball in Hand in which the user directly controls the viewpoint with a device that directly maps its position and orientation. Our approach adopts the use of widgets to disambiguate rotation in all axes and provide the user more precise control. Go-Go [Poupyrev et al., 1996] and PRISM [Frees and Kessler, 2005] switch between direct (1:1) and scaled interaction, either for large-scale or precise manipulation, based on the user's intentions which are determined by the velocity of the hand. The HOMER system [Bowman and Hodges, 1997] attaches a virtual hand to an object selected by a light ray selection, allowing the user to manipulate far objects with no extra work.

[Lucas et al., 2005] introduce alternative 3D resizing widgets, including the straight-

forward generalization of the familiar 2D resizing widgets, the Pointer Orientation based Resize Technique (PORT), and the Gaze-Hand resize technique. Users prefer the familiar widgets, but after a training period they perform faster with the novel widgets.

4.3 Prototype System Description

We designed and implemented a system for 3D immersive modeling aimed at initial prototyping or conceptual design, including modification of textured geometry and lighting. The system supports three different interaction modes, notably *table* (see Figure 4.4(b)), *immersive* (see Figure 4.4(a)) (1:1 scale, fully immersive) and *mixed* (see Figure 4.4(c)), which is a combination of the former two modes.



(a) (b)

(c)

Figure 4.4: Sketches representing each one of the interaction modes: in *Immersive Mode* (a) the user is placed inside the confines of the 3D model; in *Table Mode* (b) the user stands in front of a virtual table to interact with a small scale version of the 3D environment; and finally *Mixed Mode* (c) is a combination of the previous modes: the user is immersed inside the 3D environment but a small scale version of it is also displayed, allowing him to interact and perceive changes in both models simultaneously.

The underlying modeling system is the one developed in Chapter 3, in which the user "fixes" a set of vertices, and then manipulates one or more "variable" vertices. The system solves an optimization and updates the rest of the model accordingly. In addition

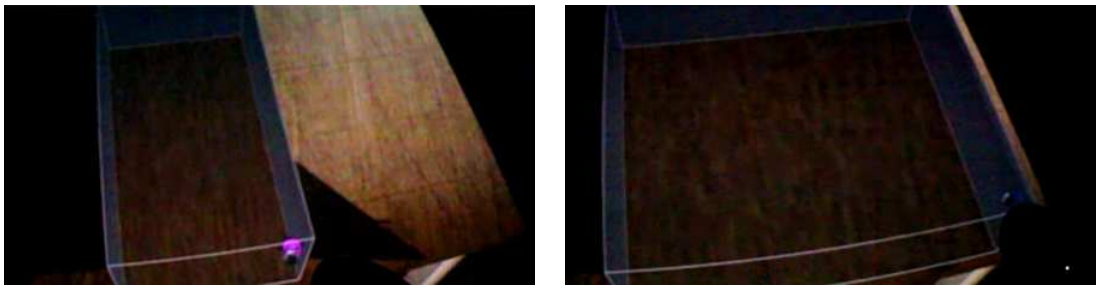
we have added new functionality for lighting and the control of sun position, which we describe later. We have modified the system to solve the specific task of building a simple house from a set of pre-built rooms which can be assembled.

The user can add doors and windows which can subsequently be manipulated. Doors and windows are modeled as textured faces which have the same texture of the underlying wall. Resizing windows and doors works the same as with the rest of the geometry, using the reshape operator describe in Chapter 3.

Different "rooms" can be assembled, which communicate through portals (our doors) allowing the propagation of the mesh modifications. Users interact using both hands, in a co-located manner, based solely on simple actions that we call "grab-move-release": the user grabs "something" (a widget, an avatar); then moves it to a particular position in the 3D space to achieve an action; to finally release it, confirming that action. We take advantage of human nature to perform coordinate two-handed interaction [Guiard, 1987] and allow simple gestures using two hands for resizing windows: the user grabs any two-corners of a window and move their hands, to interactively resize the window.

(a) (b)

Figure 4.5: Resizing a wall in *Mixed Mode* (a) The wall widget can be grabbed, resulting in a resized wall (b).



(a) (b)

Figure 4.6: Resizing a wall in *Table Mode* (a) The wall widget can be grabbed, resulting in a resized wall (b).

For our prototype, we introduce a layer on top of our system that knows which one of

(a) (b)

Figure 4.7: Resizing a window in *Immersive Mode* (a) Any combination of the four window widgets, placed respectively at each corner, can be grabbed and translated, resulting in a resized window (b).

(a) (b)

Figure 4.8: Resizing a window in *Table Mode* (a) Any combination of the four window widgets, placed respectively at each corner, can be grabbed and translated, resulting in a resized window (b).

the vertices are variables and which ones are fixed. This layer contains pre-defined sets of vertices representing the objects that can be edited. These objects are windows, doors and walls. We identify these objects manually since our models are not annotated. This is done as a pre-processing step for each different piece type. Each piece is a room composed of four walls, one floor, one ceiling and four portals, which are initially disabled. These portals can become either windows or walls at run time.

The system is interactive, and thus geometry and textures of 3D models, as well as lighting, are updated on the fly. The system receives input from the user through the use of a bi-manual interface, which is currently implemented using an ART [1] wand (dominant hand), called *flystick*, and a hand-tracking device (non-dominant hand). The user is head-tracked and all interaction is performed in a BARCO *iSpace* 3.2x3.2x2.4m

[1] http://www.ar-tracking.de/

(a) (b)

Figure 4.9: If the user is outside the 3D environment, walls and windows are rendered with transparency, only showing edges of windows and/or doors (a),(b).

(a) (b)

Figure 4.10: Moving a window in *Mixed Mode* (a) Any one of the four window widgets, placed respectively at each corner, can be grabbed and translated, resulting in a new position for the window (b). The advantage of the *Mixed Mode* is that the user can experience editing transformations immersed in the virtual room.

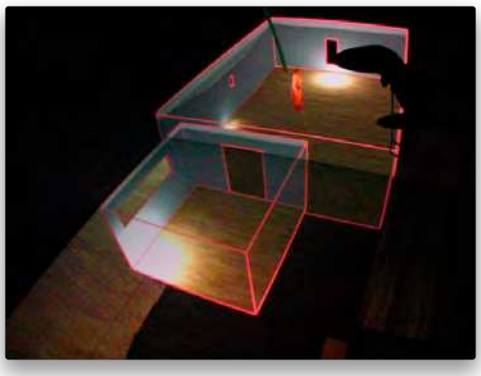
4-wall system [2]. The system used in our developments can be seen in Figure 4.12.

4.3.1 Widgets

The user can interact with the environment through a set of "selectable widgets". The list of available widgets along with their pictorial representation is shown in Table 4.1.

[2]http://www.barco.com/en/virtualreality/product/732.

(a)



(b)

(c)

Figure 4.11: The human avatar widget can be grabbed (a) and dragged inside (b) a particular room, transporting the user to that room by changing the interaction mode from *Table Mode* to *Immersive Mode* (c).

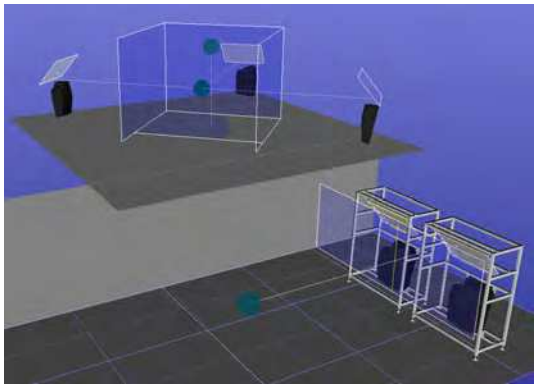


Figure 4.12: The virtual reality *iSpace* used in our experiments.

Widgets	Icon	Widgets	Icon
(a) Move Wall, Move Window Widget		(b) Sun Widget	
(c) Add Door Widget		(d) Add Window Widget	
(e) Go Into Immersive Mode Widget			

Table 4.1: List of available *widgets* for user interaction.

Each widget allows the user to perform a unique action. For instance, a "move widget" (see Table 4.1(a)) allows the user to grab, then move or resize the wall or window. Figures 4.5(a) and (b) shows an example of a wall resize in mixed mode. Figures 4.7(a) and (b) shows an example of a window resize in immersive mode.

A “human avatar" (see Table 4.1(e)) can be dragged and dropped inside a particular room, changing the viewpoint so that the user is transported to immersive mode inside that room. A button-press is used to go back. Figures 4.11(a) and (b) exemplies this process.

The viewpoint change is performed with a smooth transition, interpolating the scale of the scene similar to [McCrae et al., 2009]. We however, use a unique interpolation speed given the simplicity of our architectural scene.

4.3.2 Lighting Design

A special widget is provided for lighting design: the user can move an orange colored and textured sphere (see Table 4.1(b)) representing the sun over two trajectories (latitude and longitude). The date and time are displayed in 3D next to the sun widget; when these are changed the lighting is updated in the scene (see Figures 4.3 (h) and (i)).

To simulate natural lighting inside the room, we use a simple global illumination algorithm. Sun rays entering through a window will directly illuminate a polygonal area of the interior of the room. We call this polygonal patch P (see Figure 4.13). Other surfaces of the room, for instance, the opposite wall facing the window will receive light indirectly from this patch. Global Illumination algorithms attempt to solve this complex light interaction problem. However, computing these light interactions precisely is computationally expensive and is prohibitive for our application.

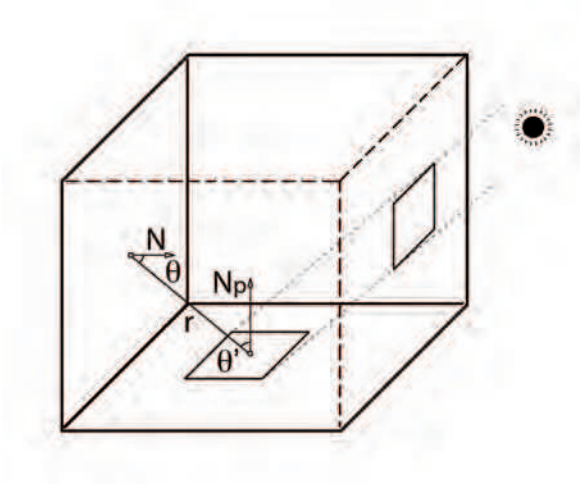


Figure 4.13: Light rays enter the room through a window, illuminating its interior with both direct and indirect lighting. We use a simplified one-bounce global illumination approximation to provide an approximate and visually acceptable result at interactive frame rates.

Empirically we have found that a simplified light interaction computation sufficed for our purposes of conceptual lighting design. A simplified one-bounce illumination model is used: a directional light source illuminates the scene through the windows in the room created by the user. The center of P is used as if it was a secondary light source, illuminating the other surfaces of the room, using a point to point differential form-factor [Greenberg et al., 1986]:

$$F_{x,P} = \int_{y \in P} \frac{\cos\theta \cos\theta'}{\pi r^2} V(x,P) dy \tag{4.1}$$

For a sun patch P, we integrate over all surface points x in the room to find the form factor $F_{x,P}$. r is the distance between each surface point x and the center of P. θ is the angle between the surface normal at the surface point x and r. Respectively, θ' is the angle between the surface normal of P and r. Please refer to Figure 4.13 for more details.

This is a simple approximation, but it provides an effective first impression of global illumination changes due to modifications in lighting position.

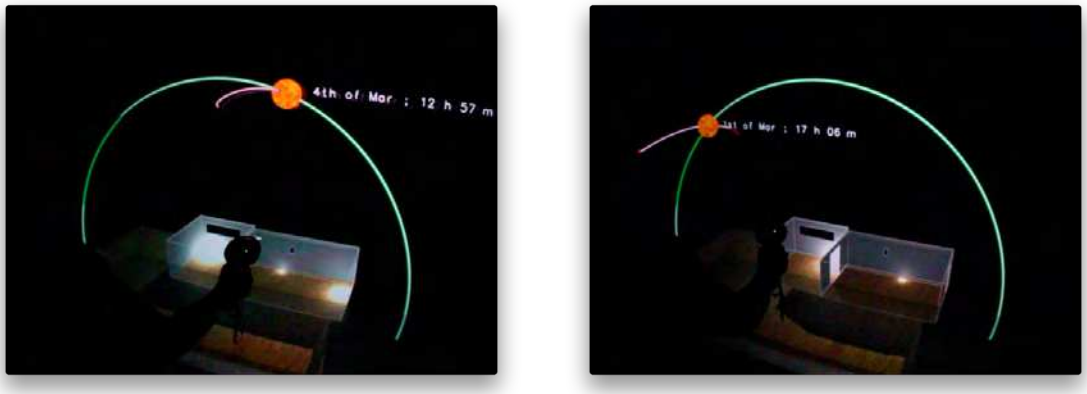
Since we do only one bounce and assume that all points are visible to the sun patch P, the form factor is computed per pixel in the GPU using the following formula:

$$F_{x,P} = \frac{\cos\theta \cos\theta'}{\pi r^2 \cdot A_P} \tag{4.2}$$

where A_P is the area of the projected sun patch P on the surface of the room.

In Figure 4.14 we can see how lighting changes inside a room when the user moves the sun. The overall illumination of the room changes accordingly. This simple approximation proves to be visually effective for quick lighting design in real time.

4.3.3 Interaction Modes



(a) (b)

Figure 4.14: (a) The sun widget can be grabbed and moved to a different location (b), changing the illumination of the room.

Actions can be executed in three different editing modes. All three modes take advantage of the immersive nature of a iSpace system, putting the user within reach of all tools for prototyping an architectural model. All modes allow co-located interaction based on the widgets described above.

Table Mode The first mode is based on a table top design, with building blocks and tools available for interaction: the user stands in front of a table with the building blocks available for usage (Figure 4.15(a)). The user begins the architectural design by choosing a pre-designed 3D model e.g., a room, corridor, etc. Room selection is provided to the user with a floating rotation menu with options (see Figure 4.15(b)), displayed to his right and within reach (co-located paradigm). As the user waves his hand over the menu, it rotates and displays all available options; the user presses the wand button to choose the appropriate room (please see video located in the online Appendix *http://www-sop.inria.fr/members/Marcio.Cabral/thesis/*).

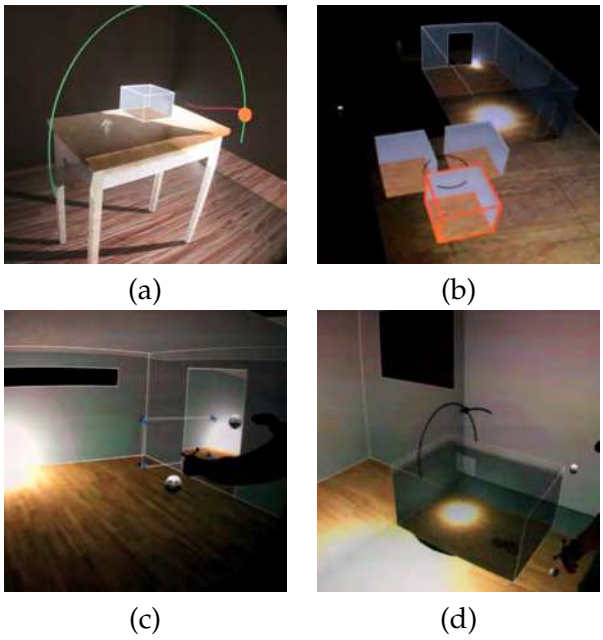
Table (and mixed) mode resembles the WIM paradigm, with the addition of the underlying table which gives the user stability and a "real location" to position its prototype. Since the user observes the world from an external perspective, this is an exocentric view.

Immersive Mode The second editing mode is egocentric: the world is displayed at a 1:1 scale and the user is correctly immersed in the architectural model (Figure 4.15(c)).

In this second mode the user has the same editing tools available as in the previous mode. Interaction occurs in the same manner as in table mode using the widgets. One of the challenges in this particular mode is navigation: the user is free to walk around in the

confined space of the iSpace system to explore and edit the model. This also translates to a 1:1 scale translation inside the virtual environment. However, it is often the case that the VR environment is bigger than the actual physical iSpace room. To overcome this limitation, the user can “fly through" the environment using the wand.

Mixed mode The third editing mode is a combination of the two previous editing modes. The user is positioned at a 1:1 scale inside the 3D model. While in this mode, the user is able to visualize a small scale version of the model in front of him (see Figure 4.15(c)). This approach is similar to the one described in [Stoakley et al., 1995]. The main difference is that the miniature model is placed in a fixed position, always displayed in the center of the physical *iSpace* projection system.



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

Figure 4.15: (a) Table mode, (b) Room rotational menu (c) Immersive mode (d) Mixed mode

4.4 User studies and Evaluation

The integration of the prototype modeling system described in the previous Chapter 3 and a simple lighting control interface into an immersive environment allows us to study the relative effectiveness of the different modes (table, immersive and mixed) for several different tasks.

The study has three parts: a training session, then a first set of specific tasks and finally a more open task in which the user is asked to construct a three room house. The specific tasks are: resize a wall, add/move/resize a window, and move the sun. For placement and resize tasks, the user is presented with a target guide in wireframe which allows objective error in the task to be measured. For the sun task, the user is presented

	Table	Immersive	Mixed
Add window	✔	✔	✔
Move window	✔	✔	✔
Resize window	✔	✔	✔
Resize wall	✔	✔	✔
Add door	✔	✔	✔
Add room	✔	✗	✔
Move sun	✔	✗	✔

Table 4.2: Available actions for each interaction mode.

with a square having a target intensity, and is asked to move the sun to match the target intensity. Figure 4.16 (a) shows the target wireframe in Table mode; in Figure 4.16 (b) the user is performing the test, trying to match the wireframe intensity to the one in the front room wall below.

Table 4.2 gives an overview of which actions can be accomplished in a particular mode. The participants are also presented with a questionnaire with subjective questions after the experiment.

4.5 Experimental procedure

A total of 8 participants completed the study in the *iSpace* of our institute, all male with ages varying from 25 to 40. All reported normal vision except for one subject who had stereo deficiency. Although we did not test the vision of the subjects, 3D glasses can be worn over corrective glasses and the subjects were instructed to do so.

We did not test handedness but we instructed people to hold the wand in their dominant hand. We will now detail the experimental procedures.

4.5.1 Training Session

The experiment starts with a training session so the participant can learn the interface.

The participant is introduced to the interface by an experimenter who guides the user through the process. The participant starts in table mode, with a simple room created and is then guided through each action available in each mode. The participant is then asked to perform the action until the experimenter is satisfied that the knowledge of the action has been acquired. The training session is extensive to minimize learning effects during the follow up part of the experiment.

The training session takes between 10-15 minutes. The detailed procedure for the training is described in Algorithm 4.1.

The training session can be extended until the experimenter is satisfied that the user knows how to accomplish all tasks.

Algorithm 4.1 Training Session Detailed Procedure

```
for all modes {table, immersive, mixed} do
    put the user in the mode
    start by showing a simple room
    for all actions available in the current mode do
        give detailed step-by-step instructions
        let the user perform the action
        correct and retry if necessary
    end for
end for
```

4.5.2 Objective Specific Study

After a short break, the participant is asked to perform the set of tests involving the actions listed in Table 4.2. The window and sun tests are performed for each of Table, Immersive and Mixed mode; the wall resize test is only performed in Table and Mixed mode. For all add/move/resize tests, we record time to achieve the task and error compared to the wireframe target. Completion time is also recorded for the sun test and the error in color between the target square and the corresponding square on the wall is computed (see Figure 4.16, second row).

Users performed the tasks in the following order, executing all tasks for a particular mode before proceeding to the next mode.

1.Table Mode: resize room, move/resize window three times, sun task

2.Immersive Mode: move/resize window three times

3.Mixed Mode: resize room, move/resize window three times, sun task

The Objective Specific Study session took between 10-30 minutes. The detailed procedure is described in Algorithm 4.2.

4.5.3 Objective Open Study

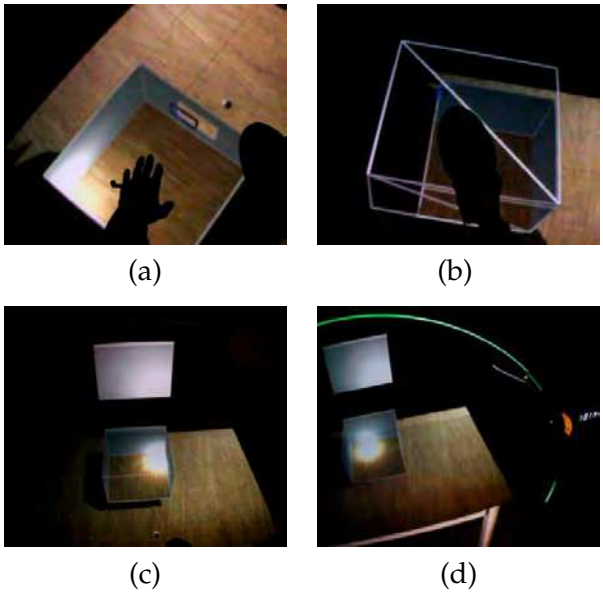
For the open task, we want to measure speed, accuracy, and behavior in an open-ended task with a specific goal. The participants were instructed to construct a three room house within a time limit of 5 minutes and to make sure that all rooms have enough sunlight in winter but not too much in summer. Participants could use any combination of editing operations, in any mode (or combination of modes) they had learned from the previous tasks.

4.5.4 Subjective Questionnaire

At the end of each study a questionnaire is presented to the participant. The post questionnaire covered the following topics:

Algorithm 4.2 Objective Specific Study

for all modes {table, immersive, mixed} **do**
  for all actions available in the current mode **do**
    start by showing a simple room
    give a specific task:
    task 1: resize room to match the constraints shown
    for $i = 1$ **to** 3 **do**
      task 2: move / resize window
    end for
    task 3: match the intensity of the target wireframe shown to the intensity of the front wall; in order to accomplish this, move the sun and/or window until the goal has been achieved
    let the user perform the task
  end for
end for



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

Figure 4.16: (a) Target wireframe for a window resize and (b) a room resize task. (c) Sun target in table mode (start of test) (d) Sun target at the end of the test.

Satisfaction How pleasant is each mode? How much is your creativity hindered by each mode? How pleasant is the iSpace?

Comfort How tiring (mentally and physically) is it to use the system?

Presence Did you feel that you were really "there" in the virtual world while using the different modes? Did you feel that you were directly manipulating a physical object or just a representation? How aware were you of the control devices and the display system?

The responses to the questionnaire were all positive about the system overall, finding the experience to be interesting and engaging. There was a small preference for table mode (4 subjects). Overall users appreciated the immersive nature of the system, and found interaction to be natural and pleasant.

4.6 Experimental Results

Tasks in the immersive mode can take slightly longer (e.g. approximately 30 seconds longer than the 42 seconds it takes on average for each window manipulation task in the other modes, two-sample t-test with unequal variances $p < 0.05$) if the user first needs to walk, virtually or physically, to get closer to the object. Some participants reported that it was difficult to reach the corners of the large windows when they were standing too far away. On the other hand resizing the windows from a suitable location was a little more accurate in immersive mode than in the other modes.

Timings for the task of room resize are equivalent both in Table and Mixed modes (73 $\pm$ 28s versus 54 $\pm$ 27s, $p > 0.05$). This is also the case for the sun positioning task (283 $\pm$ 17s versus 149 $\pm$ 8s, $p > 0.05$).

In Table mode, completing the first window resize task is marginally slower than for subsequent windows resize tasks (second and third window) because of a learning effect. Timings of first window versus second/third window resize in Table mode: 55 $\pm$ 28s versus 32 +/- 28s, $p > 0.05$.

Again, we observe the same pattern in Immersive mode: completing the first window resize task is marginally slower than the window resize task for the third window because of a learning effect. Timings of first versus third window resize in Immersive mode: 70 $\pm$ 55s vs. 53 $\pm$ 54s, $p > 0.05$.

In immersive mode, the second window resize task obliged the user to walk towards it using the wand's joystick. This caused the second window resize task to perform marginally slower than for the third window (97 $\pm$ 76s versus 53 $\pm$ 54s, $p > 0.05$ so nonsignificant difference).

Table 4.3 shows detailed timings for the Objective Specific Study session.

We measure accuracy for the Add / Move window task as the average positional error of the 4 corners of the window in virtual space. The virtual space matches the real space of the CAVE system, which is about 2.5 meters tall and 2x2 meters in length. Room resize task accuracy is proportional to the positional error of the corner of the room and is computed using the following formulae:

$$\frac{\parallel extent_u - extent_t \parallel}{t_{height}} \tag{4.3}$$

where $extent_u$ is a 3-d vector containing the user's choice for the room's height, depth and length; $extent_t$ is a 3-d vector of the room's target extent and; t_{height} is the target room's height.

The match criteria for the Sun task was a subjective measure: a comparison between brightness of two texture patches. Accuracy values for the Sun task are not included in

Table 4.4 because different window positions and sun positions can yield similar brightness.

Room resizing accuracy in both Mixed and Table mode are equivalent (0.081 $\pm$ 0.073m versus 0.076 $\pm$ 0.076m, $p > 0.05$). On the other hand, for window move/resize accuracy, we do not observe a learning effect for the window resize task for all modes. For room resize, accuracy is measured similarly to the window resize: as the average positional error of the 4 corners of the room.

In Immersive mode, completing the second window resize task has marginally larger error because the virtual window is too close to the real projection screen if the subject did not advance far enough with the joystick (0.104 $\pm$ 0.085m versus 0.050 $\pm$ 0.033m, $p > 0.05$).

Table 4.4 shows accuracy measurements for the Objective Specific Study session.

The Sun is harder to select in Table mode than in Mixed mode: 7.25 misses $\pm$ 10.32 for Table mode versus 2 misses $\pm$ 2.83, $p > 0.05$. Users perform better in Mixed mode, possibly because of a learning effect.

The widgets and the selection sphere that floated just in front of the wand cast a shadow on nearby surfaces. For widgets in the house, participants could use these shadows as a depth cue to guide them to the correct position. Most participants initially had some difficulty selecting the sun widget because there was no shadow to guide them and they had to rely on the stereoscopic depth cue.

During the 5 minutes of the open task, participants switched between modes on average 13 times (average 12.86 $\pm$ standard deviation 3.72), suggesting that they preferred to perform some operations in specific modes. Most editing was performed in table mode, while immersive or mixed mode were used mainly for inspection and even for fun. Participants used the undo functionality on average only twice (average 1.86 times $\pm$ standard deviation 1.95) since any move or resize operations could easily be corrected manually. From Table 4.4 we can see that mixed mode is globally more accurate than table mode. From informal observation, we noted that most interaction was performed using the table in mixed mode. This may imply some advantage of mixed mode for accuracy, but requires further investigation.

	Table	Immersive	Mixed
Add Move Window 1	54.81s	52.57s	47.06s
Add Move Window 2	28.74s	96.82s	48.18s
Add Move Window 3	35.09s	52.57s	41.15s
Room Resize	73.19s	✗	54.01s
Move sun	283.38s	✗	149.43s

Table 4.3: Objective Specific Study timings

	Table	Immersive	Mixed
Add Move Window 1	0.218	0.046	0.027
Add Move Window 2	0.146	0.025	0.093
Add Move Window 3	0.214	0.054	0.101
Room Resize	0.081	✗	0.076

Table 4.4: Objective Specific Study accuracy

4.7 Discussions and Conclusion

The results of the pilot study tend to indicate that none of the modes we used demonstrates clear superiority for the tasks proposed. As a result, all modes can potentially be used, depending on user preference, level of immersion required etc. The results also indicate that measured performance times are similar in table and mixed mode. This is interesting since in mixed mode the effects of lighting on the overall environment can be appreciated much better than in table mode only. Given the slight preference for table mode, it may be the case that mixed mode can be used as the "default view", but the user can "switch off" immersive view when they wish to concentrate on a specific task in table mode. Additional study is required to answer this question, with more complex tasks.

In conclusion, we have presented a first immersive system which allows simple conceptual design for textured geometric models and basic lighting design. We implemented and tested this system in an immersive room with a four-sided display, using three different modes: table, immersive (1:1 scale) and mixed, which is a combination of the first two (WIM). When asked to perform a more open task, all modes were used, and novice users all mentioned that they found the experience rewarding and interesting.

We are interested in pursuing these ideas further in the future, and notably investigating more complex modeling and lighting tasks. It is particularly interesting to investigate the use of sophisticated global illumination algorithms in the context of lighting design, and the importance of different levels of realism, in the spirit of [Slater et al., 2009]. We are also planning on using a bimanual gesture-based interface based on the finger tracking device [3] provided by ART. Ultimately, the long term goal of our research using the high-end iSpace is to develop appropriate interface paradigms for involved 3D conceptual design tasks even for low-end stereo display systems with low-resolution tracking (e.g., webcam based). We plan to investigate such paradigms and in particular test how our various hypotheses and designs bare out when we degrade the quality of the various system elements (display surfaces, tracking, stereo).

[3] http://www.ar-tracking.de/Fingertracking.54.0.html

Part II

Easy Image Relighting for Trees

Introduction to Part II

In the second part of this thesis we continue to explore techniques that aid users to create new content from existing data.

Photographs are probably the most important source used for content creation. Photographs are used as a basis for texturing geometric models for "from scratch" content creation. They are also the basis for image-based modeling and rendering. However, photographs suffer from a major limitation, which is the fact that lighting and shadows are already present in the image. It is thus hard to reuse the images in any context other than a lighting setup which is exactly the same as that in which the photograph was taken.

Relighting images is thus a very interesting and promising research direction. In the general case, the problem is of course very hard, since it is first necessary to remove shadows and lighting. Previous solutions depend heavily on detailed reconstruction of geometry and careful measurements of reflectance properties. Interestingly, the case of trees canopies is a good starting point, since the geometric nature of tree canopies has an intuitive equivalent to a semi-opaque volume. As we shall see, we will build on this intuition to develop our solution.

Ultimately, our goal is to provide a relighting tool for tree canopies which is image based and can be achieved with a small set of photographs taken at a **single** time of the day and does not rely on geometric information as input. In the following Chapter, we present previous work related to our approach. And in Chapter 6 we introduce how this can be achieved using a single scattering volume rendering approach.

CHAPTER 5

Previous Work for Part II

Contents

The goal of changing the lighting conditions of a photograph, which we call *relighting* from now on has been the focus of much past research in Computer Graphics. Initial approaches were based on constructing an approximate model of the scene, estimating reflectance parameters and then performing a relighting calculation sometimes based on global illumination computations (e.g., [Yu and Malik, 1998, Yu et al., 1999, Sato et al., 1999, Loscos et al., 2000, Boivin and Gagalowicz, 2001]).

[Yu and Malik, 1998] merges rendering and image-based techniques to produce novel renderings at different lighting conditions than those of the input data. To do this, the authors introduce the concept of "pseudo-brdf", which are exactly the same as the real BRDF if the original BRDF does not vary with wavelength. BRDF stands for the bi-directional reflectance distribution function and it models how light interacts with a given opaque surface. It is a four dimensional function which takes as parameters the incoming and outgoing light direction which are defined with respect to the surface normal. It returns a ratio between the incoming irradiance and the outgoing radiance. It was first introduced by [Nicodemus, 1965] and later widely used in computer graphics with several approximations. The authors recover two sets of pseudo-BRDF's, one for the sun and one for the sky. With this information, they generate renderings of architectural buildings under novel illumination from the input photographs (approximately 100 photographs). The results are qualitative comparable to ground truth photos taken at the target synthetic render time. The method however requires a relatively high number of images to work. This is due to the process of recovering reflectance of the building facades; it requires the separation of the sun and sky lighting, which can be done by taking photographs from the same face, at different angles and different times of a day, when a surface does not receive sun light directly.

[Yu et al., 1999] introduces a new technique that is able to estimate reflectance properties of a real scene and allows re-rendering of it under new lighting conditions. The algorithm introduced is called inverse global illumination; it captures a set of HDR photographs from a scene with known illumination and geometry to compute radiance maps for each one of the surfaces present in the scene. A low parameter reflectance model is used to solve an inverted radiosity equation to radiance maps. Radiance maps

are images that encode a large dynamic range, allowing both for very bright and darker points to be represented correctly. Using the known geometry and recovered radiance maps, the scene can be render under novel lighting conditions. Ground truth comparisons with novel lighting conditions rendered, show the system successfully achieves its goal. The input data capture process is time consuming however. In addition, the requirement of geometry, even though estimated by the input photographs, restricts the applicability of this method.

Concurrently with this approach, a similar method [Loscos et al., 2000] was developed that allowed interactive relighting of scenes. It require a smaller number of photographs and thus a simpler capture procedure for radiance estimation. It is restricted however to a single viewpoint, that is from the input photograph. It also requires that at least one view of a particular textured surface has no shadow. Nonetheless, the use of efficient global illumination algorithms allowed interactive updates with changing illumination and the mix of real and synthetic objects.

[Sato et al., 1999] proposes a method for superimposing virtual objects onto a real scene. Their method estimates the 3D geometry from a pair of fisheye images of the scene. The geometry estimation begins by identifying key points in the scene. These points are direct sources of illuminations such as a fluorescent light or a external window. Since the stereo pair of the fisheye cameras are calibrated using the Tsai's method [Tsai, 1986], these points are recovered in the scene geometry and a mapping between real and virtual objects can be stablished. The rest of the scene is approximated by first recovering a 2D Delaunay triangular mesh over these key points. Using these correspondences and the connectivity found, a 3D mesh can be recovered. Radiance distribution is estimated by establishing a relationship between the image irradiance from the fisheye photo and radiance incident on a scene object. This relationship can be established because the design of the fisheye lens allows the computation of the direction of the incoming light onto a particular point on the image plane. The estimated radiance allows rendering of scenes with superimposed virtual objects while taking into account shadows and shading, both in indoors and outdoors environments.

[Boivin and Gagalowicz, 2001] presents an approach that estimates surface reflectance properties of a scene from a single image. The authors iteratively and hierarchically compute surface properties to try to fit the Ward BRDF parameterization [Ward, 1992]. The scene is modeled and registered manually using Autodesk Maya. The illumination for which the scene was photographed is also known. With the recovered properties, the scene can be rendered under different lighting conditions. Textured surfaces however do not work well for this technique since shadows and specular highlights cannot be correctly identified.

The environments used in these approaches did not involve complex geometries such as trees, since they are hard to reconstruct from images. An interesting recent solution to image-based relighting taps into the sheer volume of images and lighting conditions available on the web [Lalonde et al., 2009]. While very promising, it has similar limitations concerning required geometry reconstruction in the case of trees.

Lighting in tree canopies has been previously studied in computer graphics and other

fields. [Soler et al., 2003] describe a hierarchical instantiation scheme for radiosity rendering of vegetation that takes advantage of self-similarities that exists in plants; their solution is on entirely synthetic models, with exact geometry and materials available. In the field of agronomy, de [de Castro and Fetcher, 1998] model lighting in a tree canopy by analyzing illumination arriving at individual grid cells overlaid on a vegetation. They validate this model with real measurements on a coarse grid and in an "artificial" plantation. While this model has some similarities to our approach, their focus is on producing average illumination values at much larger scales, suitable for agronomy. As a result this approach cannot be directly applied for image relighting. In addition, we assume that the medium is isotropic and that we have single scattering.

The work of [Boulanger et al., 2008] utilizes a volumetric approach to approximate visibility in a tree canopy. However, their approach is a tree rendering method, which relies on actual geometry and normals to compute visibility and create renderings. Our method is quite different from this approach as it does not depend on actual tree canopy geometry as input - this information is hard capture and estimate if the intent is to use a small set of photographs (approximately 12 photos) as input.

There has also been a significant amount of recent work on tree modeling from images. Most approaches (e.g., [Shlyakhter et al., 2001, Neubert et al., 2007, Tan et al., 2008]) use the images to guide the generation of a completely synthetic model; an alternative is to create an image-based volumetric representation using a voxel grid with billboards [Reche et al., 2004]. This approach first builds a volumetric proxy by taking a set of photographs around the tree. These photographs are then calibrated to recover camera parameters. For each view, a matte separating the background (world) from the foreground(tree) are identified. A density value for each voxel is estimated using the image mattes; the process is repeated until a convergence threshold is achieved. This process effectively estimates a volumetric proxy of the tree by either removing voxels that are unseen by any of the image matters and increasing the density of voxels that are seen by one or more mattes.

While it would be possible to use the resulting purely synthetic tree models for relighting in the spirit of [Debevec, 1998, Boivin and Gagalowicz, 2001, Debevec, 2002], it would be necessary to extract leaf reflectance parameters from the images. We will build on this method to create a volumetric representation of trees in the input images to perform relighting. While some methods exist for large scale reflectance capture in radiometry [Rahman et al., 1993] it is unclear whether these could be used here. Even if reflectance were available, the final result would not completely match the image since the resulting synthetic models are not usually pixel accurate. Thus our goal of relighting the tree in the actual photograph would not be achieved.

Our relighting formulation is based on ratios of images that depict different environment conditions to compute lighting at a target time. These methods have been extensively used in the literature for solving inverse lighting problems [Marschner and Greenberg, 1997, Liu et al., 2001, Shashua and Riklin-Raviv, 2001, Stoschek, 2000]. Marschner [Marschner and Greenberg, 1997] was the first to introduce image ratio multiplication targeted at relighting. Given as input a rough 3D model ap-

proximating the scene, the camera and basis lighting, the authors construct low frequency renderings of the output photograph. These renderings are computed at both lighting conditions. The ratio of these two images is then multiplied to the original photograph, pixel by pixel, giving the final result. High frequency details are not recovered by the model but are present in the original input photograph.

A recent trend in relighting research is based on complex capture setups, such as the Light Stage [Debevec et al., 2000] and followup work, e.g., [Peers et al., 2007]. These results have produced impressive results of stunning realism, and have been extensively used in film production. Our approach is orthogonal to such capture setups, since our goal is to have a simple capture workflow using a simple digital camera. In addition, most such laboratory setups [Debevec et al., 2000] are inappropriate for capturing trees, due to size and transportability issues.

5.1 A participating media lighting model

We will be using a participating media lighting model. For a comprehensive survey, please see [Cerezo et al., 2005]; we describe here the basic concepts that we will be using in our work. Our notation is inspired from this survey.

As energy travels and crosses a participating medium, three different processes can occur, according to [Cerezo et al., 2005]: absorption, scattering and emission. For our particular case, we make the assumption that light going through the canopy of a tree undergoes only scattering and absorption. We observe that a tree canopy is qualitatively similar to a participating medium since it is composed of a very large number of leaves. These leaves play the role of particles. Our relighting method is based on this idea and it allow us to make an estimate of lighting parameters for different lighting conditions, effectively allowing tree relighting.

If we consider that leaves in a tree have a random orientation, we can consider scattering to be isotropic. Furthermore, as we will show later in Section 6.1.2, results from synthetics renderings indicates that indirect lighting is qualitatively similar and comparable to single scattering.

The mathematical formulation that allows us to express the phenomena of energy reduction as it travels through a medium, or the transport equation, is given as [Cerezo et al., 2005]

$$dL(x) = -k_t(x)L(x)dx \tag{5.1}$$

where $k_t = k_a + k_s$ is called the extinction coefficient (k_a absorption and k_s scattering). We estimate these quantities for our volumetric rendering formulation in Section 6.2 using image mattes extracted from photographs around the tree. Beer's law states that the solution of the differential equation 5.1 is given as

$$L(x) = L(x_0)\tau(x_0, x) \tag{5.2}$$

where

$$\tau(x_0, x) = e^{-\int_{x_0}^{x} k_t(u)du} \tag{5.3}$$

Note that 5.2 does not take into account in-scattering. For the particular case of single-scattering, the integral transport equation can be simplified as follows

$$L(x) = \int_{x_0}^{x} \tau(u,x) k_t(u) L_{ss}(u) du \tag{5.4}$$

In Section 6.2.1 we show in more details how Equation 5.2 can be used to effectively allow tree relighting using a participating medium lighting model.

5.2 Conclusion

Image-based relighting is a difficult task. In particular for trees, where geometric reconstruction is hard, this task presents several challenges such as recovering an approximate geometric proxy and estimating the lighting that each leaf receives. Relighting is then performed based on the estimated information.

Interestingly however, the nature of tree canopies also allows us to simplify computations. Intuitively we can then avoid the computation of local indirect lighting, since it is one or more orders of magnitude lower than the sun and the sky lighting.

In the following Chapter we will introduce a new solution for relighting canopies of trees based on the assumption that ultimately, tree canopies at a distance, can be interpreted as a volumetric entity. As such, we can reliably estimate enough lighting, visibility and geometric information to perform relighting using a Single Scattering Volumetric Rendering technique. To our knowledge, our method is the first to achieve this goal.

CHAPTER 6

Image Based Tree Relighting

Contents

Trees are a very compelling and important component of many outdoor images. Matting photos of trees is common practice, e.g., in photomontages for urban planning projects or in other image editing applications. One important problem in such contexts is that the lighting conditions of the tree do not correspond to the lighting of the final image (see Figures 6.2(a)-(b)). We introduce a novel approach which allows us to take a small set of photos of a tree at a *single* time of day, and then relight the tree canopy with a different, target lighting condition (see Figure 6.1(b)-(d) and Figures 6.21, 6.22, 6.28).

Relighting photographs is a long-standing goal of computer graphics. In many previous methods (e.g., [Yu and Malik, 1998, Yu et al., 1999, Masselus et al., 2003, Boivin and Gagalowicz, 2001, Loscos et al., 2000, Debevec, 2002]) geometric and photometric reconstruction and/or capture are required, followed by different kinds of inverse (global) illumination computations. More recently, relighting research has concentrated on complex, often expensive, capture setups (e.g., [Debevec et al., 2000, Peers et al., 2007]) typically involving multiple light sources and cameras.

It is unclear how tree canopies can be relit with such methods. Despite recent work on image-based tree reconstruction [Neubert et al., 2007, Tan et al., 2008], pixel-accurate

geometric reconstruction of tree canopies is very hard. Even if a synthetic tree were to be reconstructed from images, it is unlikely that photometric, geometric and lighting calibration would be sufficiently accurate to convincingly mix the synthetic tree with the photograph. On the other hand, solutions based on laboratory capture setups are inappropriate for trees. To our knowledge, no previous method has been proposed that allows relighting of tree canopy photos, using a *single* lighting condition as input.

Given the above, our goals are as follows:

1. Allow relighting of a given tree canopy from a given input photograph;
2. Provide an *image-based* method, i.e., that only takes photographs as input. No geometry or material information is required for the tree canopy;
3. Require photographs to be taken *only at a single time of day*; no input is thus required for the target relit time.

These goals have the advantage of allowing us to relight at any target time, given input photographs at a single time of day. However, a consistent analytic model must exist to represent illumination at the input time (which is captured), and at the target time (for which no data is required). As a result, methods based e.g., on light probes, which would require capturing lighting both at input and target times, are incompatible with our goals.

We do not deal with relighting of the rest of the photograph, such as shadow removal and relighting of parts of the scene other than the canopy. In future work, our results could potentially be integrated with one of the previous approaches mentioned above to handle this more general problem.

The proposed tree canopy relighting method is based on a volumetric single-scattering participating media approximation; it is image-based and does not use or generate leaf geometry or normals to achieve relighting; in this sense it should not be confused with a traditional rendering algorithm.

We build our approach based on an intuitive argument of qualitative similarity between volumetric single scattering and the effect of lighting distribution in a tree canopy. We use synthetic renderings of trees to illustrate and validate this principle. This intuition is then used to estimate a per pixel radiance value for the input photograph and then estimate an equivalent value at the target time, allowing us to achieve relighting.

Our goal is to provide a convincing relighting algorithm for tree canopies, and as such our approximations do not constitute a physically accurate model. Nonetheless, as we shall see, our assumptions provide satisfactory results for relighting. In particular, we have three main contributions:

- We present an analysis of lighting in a tree canopy, starting from first principles, and using physically-based synthetic renderings to guide our study. Synthetic trees are used only to generate renderings for analysis, and do not constitute part of our contribution. The analysis indicates that in terms of overall lighting behavior we can assume diffuse reflectance for leaves, and approximate lighting behavior in a tree canopy using a single-scattering isotropic participating media model.

(a) input lighting condition

(b) relit image

(c) ground truth

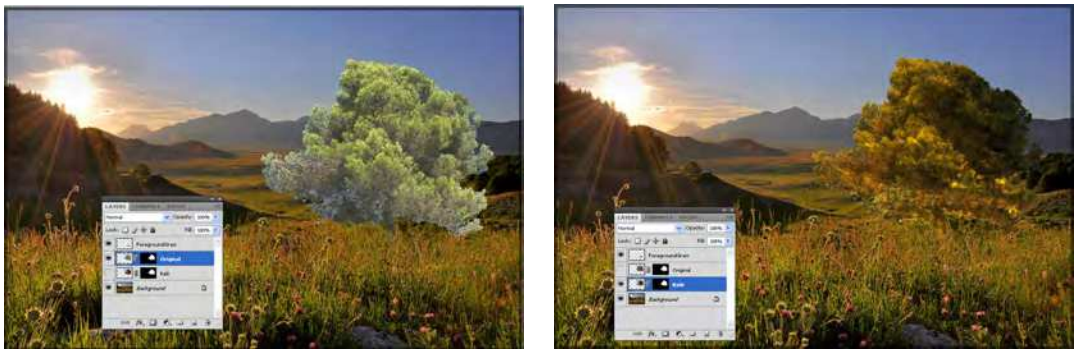
(d) relit image

(e) ground truth

Figure 6.1: (a) One of the input images taken at noon. (b)-(d) Canopies relit with our approach using only photos taken at noon, and the respective ground truth photographs (c)-(e) at the target relighting times (taken for purposes of comparison only, and not used by the algorithm). Please note that we only use (a) as input, together with a set of images taken at the same time around the tree. The target lighting conditions (i.e., (b) and (d)) are generated automatically by our method.

- Based on this analysis, we develop a relighting approach based on a volumetric lighting model, and an analytical sky/sun model [Preetham et al., 1999]. This is achieved by computing approximate representations for irradiance both at the time of the input photograph and that of the desired lighting condition, and relighting using their ratio.

- Finally, we develop a fast relighting algorithm using spherical harmonics to com-



(a) (b)

Figure 6.2: (a) A snapshot of an image editing program interface, with the layer containing a tree, and the target photo. Inserting the tree directly in the image is unsatisfactory, clearly revealing the lighting inconsistency. (b) Using our approach, the relit tree fits much better with the target lighting condition, in particular for shadowing in the canopy.

bine the effect of transmittance from the sky sampling directions and the path to the eye.

We present validation results on synthetic trees (Figures 6.12, 6.13, 6.14, 6.15, 6.16, 6.17, 6.18, 6.19), for which exact ground truth exists, and for (sparse) time-lapse sequences of three different real trees (see Figure 6.1(a)-(e) and Figures 6.21, 6.22 and 6.28)

We next start with a careful analysis of tree canopy relighting. We use synthetic global illumination renderings to study lighting in tree canopies, which leads us to adopt a volumetric rendering model (Sec. 6.1). Based on this analysis, we develop a volumetric tree canopy relighting method (Sec. 6.2) and an efficient relighting algorithm based on spherical harmonics, which were first introduced by Sloan et al. [Sloan et al., 2002] in the context of pre-computed radiance transfer. More recently, Jansen et al. [Jansen and Bavoil, 2010] introduced Fourier opacity mapping, in which absorption is projected along one ray onto a 1D Fourier basis to speed up the integral computation. In our case however, we project the directional (2D) distribution of transmissivities onto a sphere to allow for faster integration. Our method is explained in more detail in Section 6.3.

6.1 An Analysis of Tree Canopy Lighting

As discussed above, no previous method is able to achieve the goal of relighting single lighting condition photographs of tree canopies. This is due to the inherent complexity of tree canopy geometry and illumination. The analysis presented in this section shows that a number of simplifying assumptions can be made concerning lighting of tree canopies, providing us with the qualitative intuition required for our fast and efficient relighting algorithm

In particular, using theoretical arguments and synthetic simulation, we will show that a diffuse reflectance assumption and a single scattering volumetric lighting approximation (Section 6.2) are qualitatively acceptable, enabling the use of a ratio-based relighting technique, in the spirit of [Marschner and Greenberg, 1997, Liu et al., 2001, Shashua and Riklin-Raviv, 2001, Stoschek, 2000].

In the following, we will be using synthetic images to provide intuition about lighting, and to validate our approximations and assumptions.

Note however that the geometry and normals of these trees are never used in any way to generate the results of our method; using synthetic trees is just a convenient way to generate images while controlling various parameters and thus to perform validation of our various hypotheses.

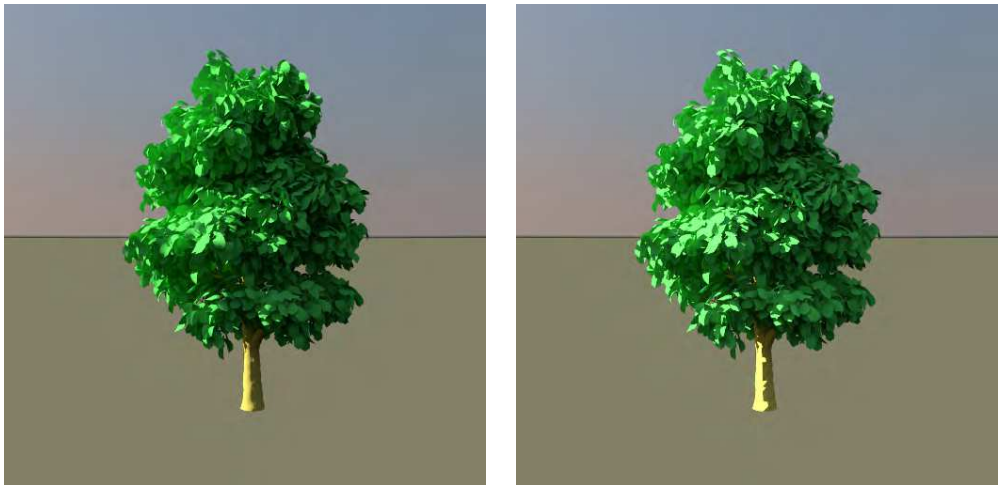
These synthetic images are rendered using PBRT [Pharr and Humphreys, 2004], a physically-based rendering system. For these images we used photon-mapping (5M photons) with final gather (128 samples/pixel and 128 directional samples), as a low-noise global illumination solution to best match reality. Note that such "reference" image requires several hours to compute. We use diffuse reflectance for the leaves, which are also slightly translucent for Figure 6.12. The sun is modeled as a directional light source and the sky is modeled using the Preetham model [Preetham et al., 1999].

6.1.1 Canopy Lighting: diffuse assumption

Consider an image of a tree canopy (e.g., Figure 6.3(a)); in what follows we will only be referring to pixels which are on the tree canopy itself. Without loss of generality, we assume that for a given pixel x we have a single corresponding 3D point x_l on a leaf in the real tree. The reflectance of x_l is $\rho(\omega_i, \omega_o)$, where ω_i is the incoming direction and ω_o is the outgoing direction. We assume two light sources, the sun, with radiance L_{sun} and the sky, which is a hemispherical source. We can sample the sky in a given direction ω, giving radiance $L_{sky}(\omega)$. We also denote $L_{ind}(\omega)$ the radiance due to indirect illumination, and θ the angle of ω with the surface normal; Ω is the positive hemisphere of directions. The radiance arriving at pixel x from point x_l, is thus:

$$\begin{aligned} L(x) = \int_\Omega & \big(\rho(\omega,\omega_o)L_{sky}(\omega,x)cos(\theta) \\ & + \rho(\omega,\omega_o)L_{ind}(\omega,x)cos(\theta)\big)d\omega \\ & + \rho(\omega_{sun},\omega_o)L_{sun}(x)cos(\theta_{sun}) \end{aligned} \tag{6.1}$$

Our first assumption is that we can assume leaf reflectance ρ to be diffuse. This may seem to be too strong a simplification; however, there are two reasons which justify this choice. First, if we assume that the leaves are randomly oriented in a uniform distribution, in a manner analogous to microfacets for surfaces, at a given scale, where each pixel captures one or more leaves, the canopy can be seen as a diffuse surface. Second, as noted by [Boulanger et al., 2008], more than half the total radiance in an image of a tree comes from the sky (which is an hemispherical source) and indirect light. In addition,



(a) (b)

Figure 6.3: (a) A globally illuminated image of a tree. (b) the same tree, in which the dependency on the cosine to the normal is ignored in all lighting computations.

the direct sunlight, while contributing a high percentage of radiance (about half), only affects a small number of pixels (see also Figure 6.4).

These observations indicate that specular effects, at a certain scale, do not always have a very strong effect on the image of a tree. For leaves with a strong preferred orientation and high specularity, these assumptions no longer hold, and this could adversely affect the quality of our results.

To further examine the effect of these assumptions on the performance of our approach, we varied the reflectance parameters of leaves in synthetic data sets, to have high specularity and then high translucency and studied the effect on the performance of our algorithm. Details of these studies are presented in Section 6.4.2.

If we assume that ρ is diffuse, we can simplify Eq. (6.1) by dropping dependencies on angle θ. Since we are integrating over the hemisphere Ω, we are now treating irradiance values, which we note with the symbol E. Radiance received at pixel x thus becomes:

$$L(x) \;=\; \rho(E_{sky}(x) + E_{sun}(x) + E_{ind}(x)) \;=\; \rho E \tag{6.2}$$

where E is the total irradiance received at x_1, and $E_{sky}(x)$, $E_{sun}(x)$, $E_{ind}(x)$ are the irradiance due to sky, sun and indirect illumination respectively. It is important to note that we could still derive Eq. (6.2) without dropping dependencies on angle: we do so to emphasize the fact that we do not have any information on normals in our volumetric approximation. For clarity, $E_{sky}(x)$, $E_{sun}(x)$, $E_{ind}(x)$ will be called E_{sky}, E_{sun}, E_{ind} from now on.

One way to evaluate the validity of discarding the cosine term in the volumetric model assumption is to examine the importance, with respect to lighting, of normals compared to visibility for tree canopies. To do this, we compute an image of a tree

Notations	
x	image pixel
x_l	corresponding 3D point on a leaf in the tree
$\rho(\omega_i, \omega_o)$	leaf reflectance at x_l
ω_i	incoming direction
ω_o	outgoing direction
L_{sun}	sun radiance
L_{sky}	sky radiance
$L_{sky}(\omega)$	sky radiance sampled at direction ω
$L_{ind}(\omega)$	radiance due to indirect illumination
θ	the angle of ω with the surface normal
Ω	positive hemisphere of directions ω
$E_{sky}(x)$	irradiance arriving at x_l due to the sky
$E_{sun}(x)$	irradiance arriving at x_l due to the sun
$E_{ind}(x)$	irradiance arriving at x_l due to the indirect illumination
$L^{vol}_{sun}(x)$	approximated radiance due to the sun arriving at x using our single scattering approach
$L^{vol}_{sky}(x)$	approximated radiance due to the sky arriving at x using our single scattering approach
$\tau(u, \omega_{sun})$	transmittance along the sun direction ω_{sun} to the point u

Table 6.1: Notations used in this Chapter

(shown in Figure 6.3(a)) using standard PBRT rendering. We then compute an image of the same tree with the same lighting parameters, but we ignore the cosine with the normal in the computation of lighting (Eq. (6.1)) (see Figure 6.3(b)). As we can see the difference in the images is minimal, except for leaves which receive direct light.

6.1.2 Volumetric approximation

A tree canopy is made up of a very large number of leaves. As such, at an appropriate scale, there is a qualitative similarity to a participating medium, in which the leaves have the roles of particles. Our method is based on this idea, which will allow us to approximate irradiance at given lighting conditions, thus enabling tree relighting. As we shall see in Section 6.3, this will allow us to approximate E_{sky}, E_{sun} and E_{ind} from Eq. (6.2) using an approximation of $L(x)$, shown in Eq. (6.12).

To investigate this qualitative analogy, we will analyze the behavior of light by separating out the various components of Eq. (6.1). In Figure 6.4(a) we show an image of a tree with global illumination. We then show the various components of illumination, notably direct sunlight Figure 6.4(b), direct skylight Figure 6.4(c) and indirect illumination Figure 6.4(d) (from both sun and skylight), Figure 6.4(e) indirect due only to skylight and Figure 6.4(f) indirect due only to sun light. Since we will be comparing to a participating media model, whose phase function will be monochromatic, we set leaf reflectance equal to (1,1,1). As we shall see later (Section 6.2), reflectance cancels out in our computations.

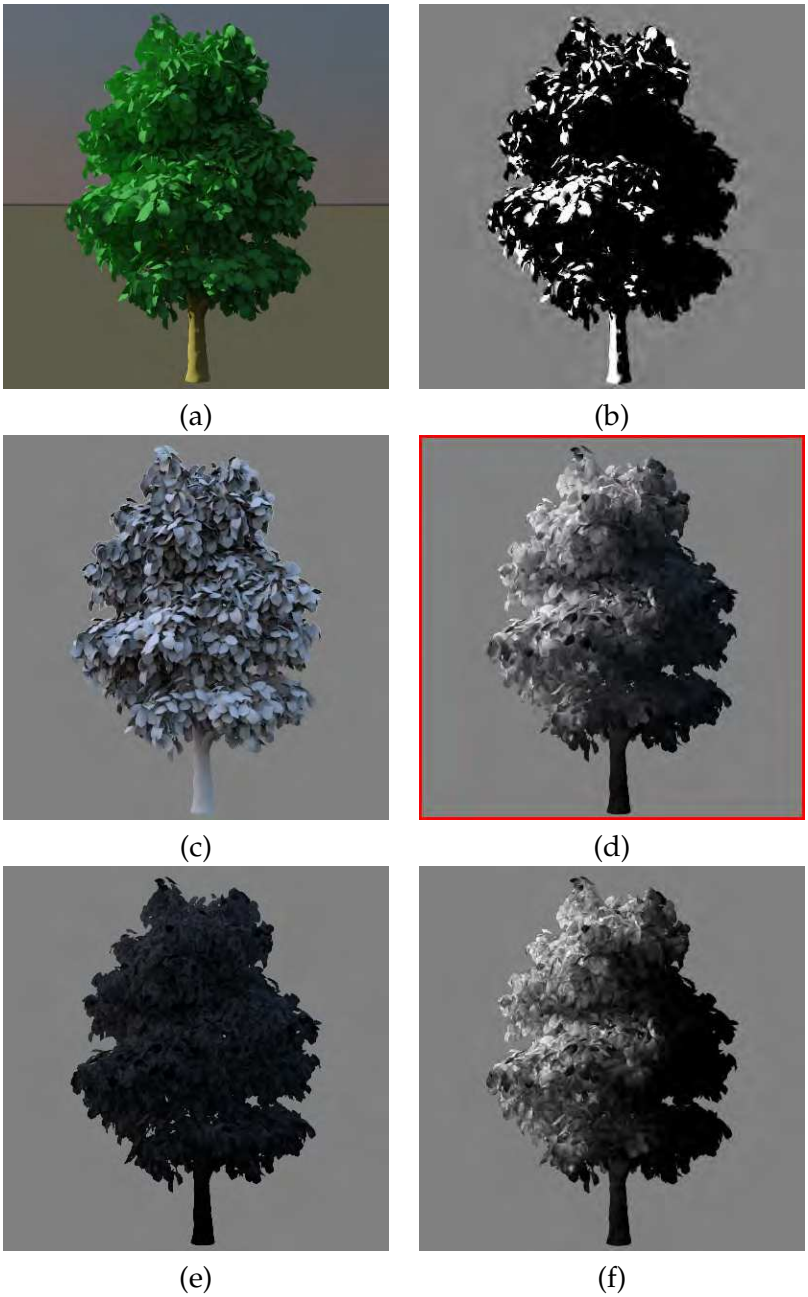
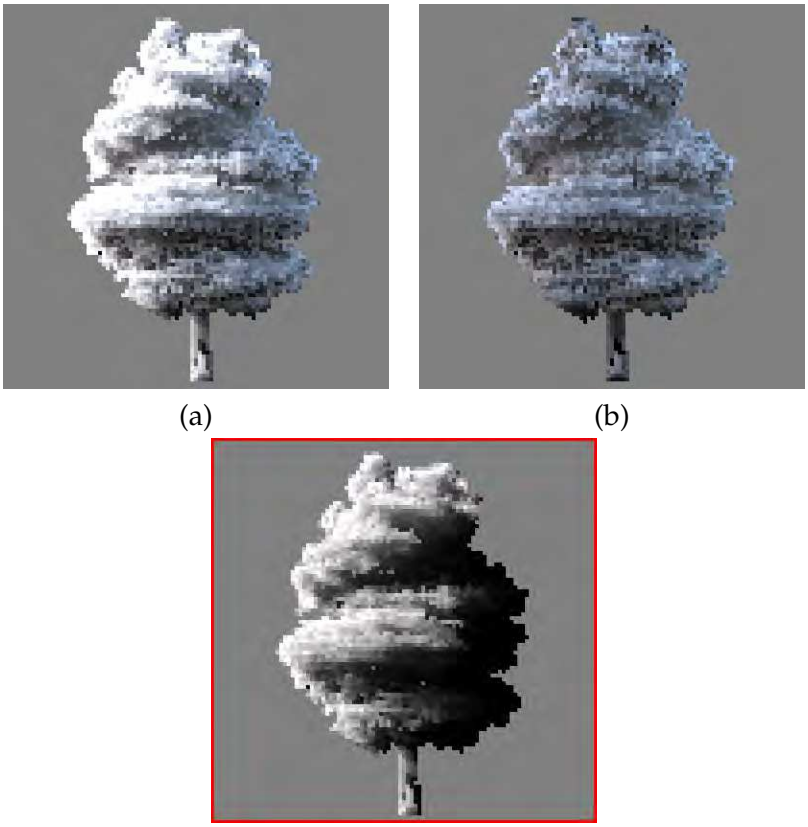


Figure 6.4: (a) A globally illuminated tree, using photon mapping, computed with PBRT. Images (b)-(f) are computed with leaf reflectance equal to (1,1,1) to allow comparison with our participating media model. (b) Direct sunlight. (c) Direct skylight. (d) Indirect illumination. (e) Indirect illumination (sky only) (f) Indirect illumination (sun only)

As we can see in Figure 6.4(b), direct lighting from the sun only affects a small number of pixels since it is only present on unoccluded front-facing leaves (with respect to the sun). Skylight (Figure 6.4(c)), while present everywhere, contributes less to the overall



(a) (b)

(c)

Figure 6.5: (a) The image $\tilde{E}$ computed with our single scattering model. (b) Skylight only component and (c) Sunlight only component.

illumination of the tree canopy. Indirect illumination (Figure 6.4(d)) represents a significant portion of overall illumination. As expected, the overall aspect of indirect light, especially due to the sun, is similar to that of scattering within a volume. As we can see in Figure 6.4(e) and (f) this is mainly due to indirect reflections of sunlight.

We use this qualitative observation as the intuition to derive our volumetric model, which allows us to develop an effective tree canopy relighting approach.

We will be using a standard participating media lighting model, described in Section 6.2.1. In particular, we shall see that we can compute an image of irradiance for a volumetric representation of the tree canopy. We use [Reche et al., 2004] (see Section 6.2.1 for details) to construct a volume, using exact "mattes" separating pixels of the canopy from those in the background.

The isotropic assumption is valid if the orientation of leaves is considered random within the canopy. As the synthetic results show, a single scattering approximation is qualitatively similar to the behavior of indirect lighting in a tree canopy.

As we shall see later (Section 6.2) our relighting does not depend on reflectance; we thus use a constant monochromatic phase function with unit value.

We show a rendering of our volumetric lighting model (Section 6.2.1) in Figure 6.5(a). We also show the effect of the sky and sun separately in (b) and (c). If we compare Figure 6.4(d) (all indirect) and 6.5 (c) (sunlight component), outlined in red, we see similar overall distribution of light. As we can see, first order scattering in the volumetric model provides an approximation of indirect light. This provides a strong indication that it is in effect unnecessary to consider multiple scattering.

In addition, Figure 6.4(d) and Figure 6.5(c) (highlighted in red) show that our model captures the overall behavior of skylight.

The above comparisons are a qualitative indication that the participating media approximation we propose provides a good approximation for the overall distribution of lighting within a tree. We make no claims about the accuracy of the absolute illumination levels, nor physical fidelity from our model; however in the context of our algorithm for tree canopy relighting, these approximations appear to work well (see synthetic validation and results on real images in Figures 6.12, 6.13-6.28, as well as the accompanying video - *http://www-sop.inria.fr/members/Marcio.Cabral/thesis/*).

6.2 A Participating Media Approach for Tree Relighting

In this section, we first discuss our relighting approach, and then present our participating media lighting model.

To achieve tree canopy relighting, we use a method based on image ratios ([Marschner and Greenberg, 1997, Liu et al., 2001, Shashua and Riklin-Raviv, 2001, Stoschek, 2000]). Assume that we have two images of the same tree canopy, at two different times of day I_{in} and I_{targ} (see Figure 6.6(a) and (b)). We can write: $I_{in} = \rho E_{in}$ and $I_{targ} = \rho E_{targ}$. The above leads to the well known property governing all diffuse materials, which we call ratio I_r:

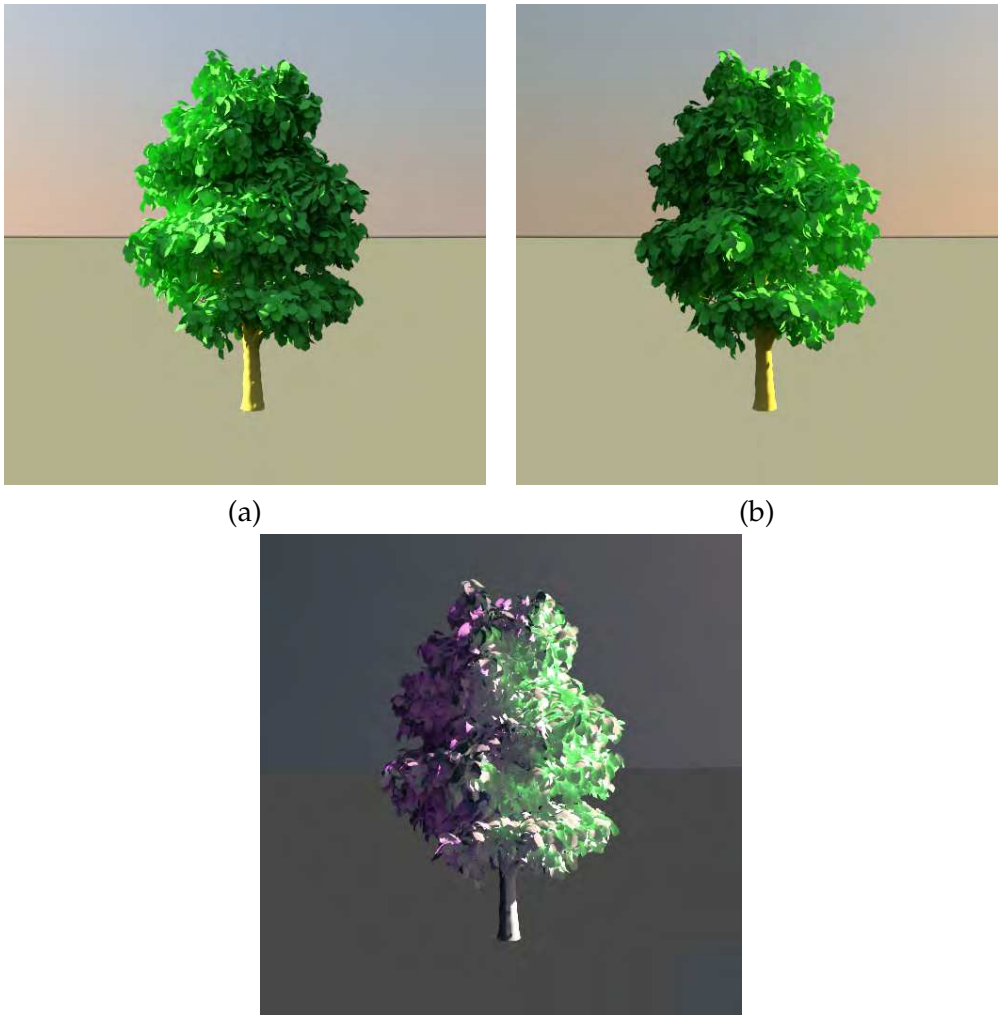
$$I_r = \frac{\rho E_{targ}}{\rho E_{in}} = \frac{E_{targ}}{E_{in}} \tag{6.3}$$

An example image I_r is given in Figure 6.6(c). Clearly, if we only have I_{in} as input, we only need the ratio I_r to compute I_{targ}, since

$$I_{targ} = I_r \cdot I_{in} \tag{6.4}$$

There are two interesting observations to be made from Eqs. 6.3 and 6.4. First, we can compute relighting *without estimating reflectance*, since only irradiance values are required to compute I_r. Second, only *relative* values are required, since only the ratio of the input and target irradiance values is required, not their absolute values. As a consequence, all we need is to compute appropriate approximations for E_{in} and E_{targ} so that their *ratio* is accurate.

The analysis presented in the previous section indicates that a good way to compute such approximations $\tilde{E}_{in}$ and $\tilde{E}_{targ}$ would be with a participating media lighting model. We describe our volumetric lighting solution in the following section.



(a) (b)

(c)

Figure 6.6: (a) The input image I_{in}. (b) The target lighting condition I_{targ}. (c) The ratio image I_r of (b)/(a).

6.2.1 Participating Media Lighting Model

Our goal is to compute values $\tilde{E}_{in}$ and $\tilde{E}_{targ}$. This will give us an approximation of E_{in} and E_{targ} and ultimately a good approximation of I_r. Once we have I_r we can relight the canopy using Eq. (6.4).

We start by constructing an approximate voxel representation of the tree; we use the approach of [Reche et al., 2004] to achieve this. We briefly recall the process here: a set of photographs (around 12-15) is taken around the tree, the cameras are calibrated, and a set of mattes created to separate the canopy from the background.

The quality of the mattes plays an important role since our method for relighting heavily depends on the quality of the reconstructed volume. When possible, blue screens

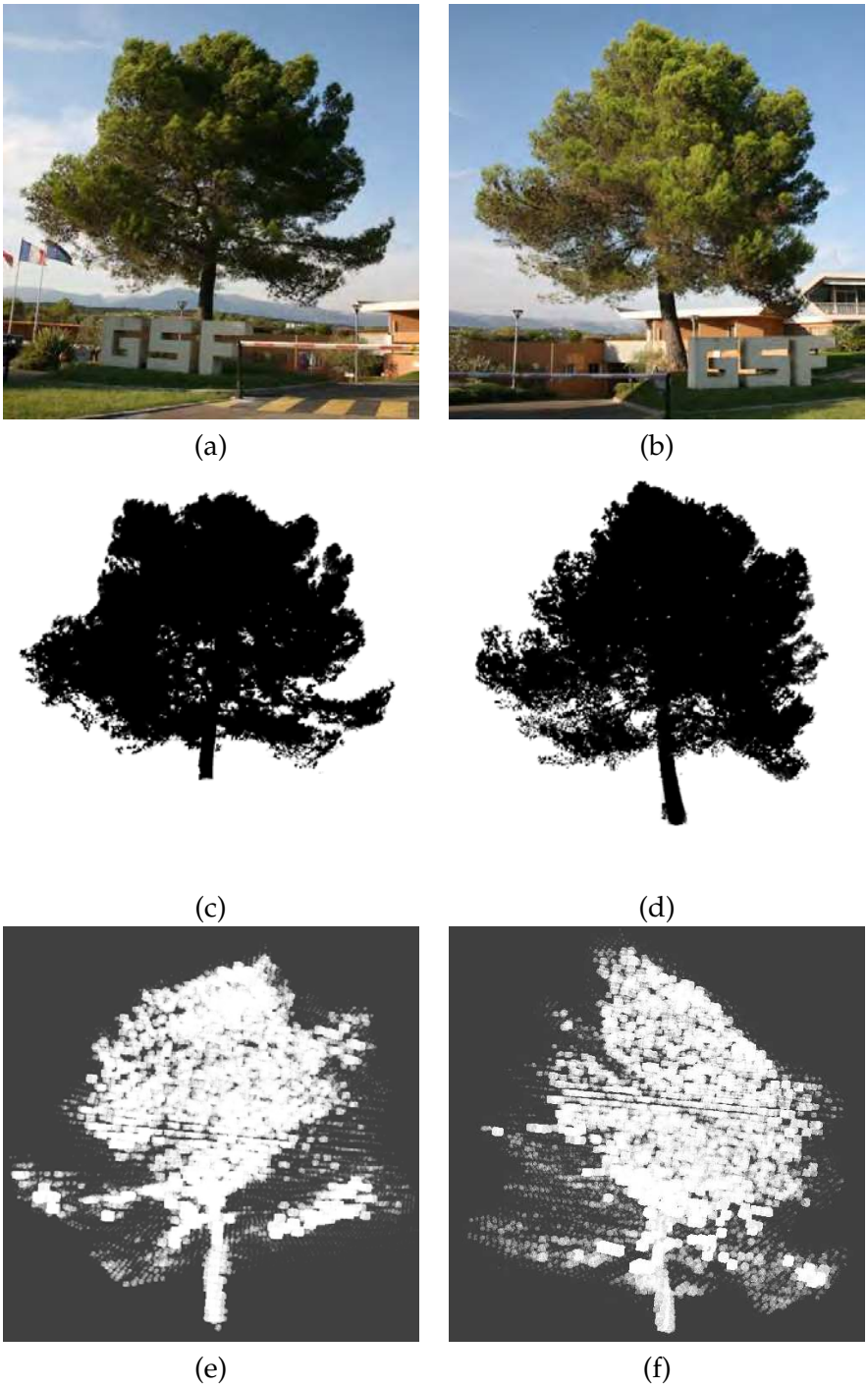


Figure 6.7: Two input photos of the same tree ((a) and (b)), the corresponding mattes ((c) and (d)) and two views of the resulting volume ((e) and (f)).

were utilized in order to isolate the subject tree properly from other vegetation behind it (see Figure 6.8). This process improves the quality of the resulting mattes.

Figure 6.8: When conditions allowed, a blue screen was placed behind the tree (*left*) to aid matte extraction (*right*).

Once the input images with calibrated cameras and the mattes are in place, an optimization process is run to estimate opacities in each voxel and to "carve out" the shape of the tree. We show an example of tree photos and the resulting reconstructed volume in Figure 6.7.

As mentioned in the introduction, the basic intuition for the model we adopted is that the leaves of the tree canopy can be seen as a participating media volume in terms of lighting; such a model thus appears suitable for the computations of $\tilde{E}_{in}$ and $\tilde{E}_{targ}$. We thus adopt a standard participating media rendering approach, based on Beer's law [Cerezo et al., 2005, Perez-Cazorla et al., 1997, Kajiya and Von Herzen, 1984]. Radiance at point x is given as follows:

$$L(x) = L(x_0)e^{-\int_{x_0}^{x} k_t(u)du} \tag{6.5}$$

where k_t is the extinction coefficient of the medium and light travels from x_0 to x. We write:

$$\tau(x_0, x) = e^{-\int_{x_0}^{x} k_t(u)du} \tag{6.6}$$

$\tau(x_0, x)$ is the *transmittance* from x_0 to x. We will also write $\tau(x_0, \omega)$ which denotes the transmittance along a direction ω from the entry point of the volume to the point x_0.

In our case, x will typically be the eye point, or equivalently the pixel, and x_0 will be the furthermost point of the volume along a ray emanating from the eye. We illustrate these quantities, and those of Eq. (6.7) and Eq. (6.8), in Figure 6.9.

Given these assumptions, and expanding $L(x_0)$ in Eq. (6.5), radiance at L is given as [Cerezo et al., 2005]:

$$L(x) = \int_{x_0}^{x} \tau(u, x)k_t(u)L_{ss}(u)du \tag{6.7}$$

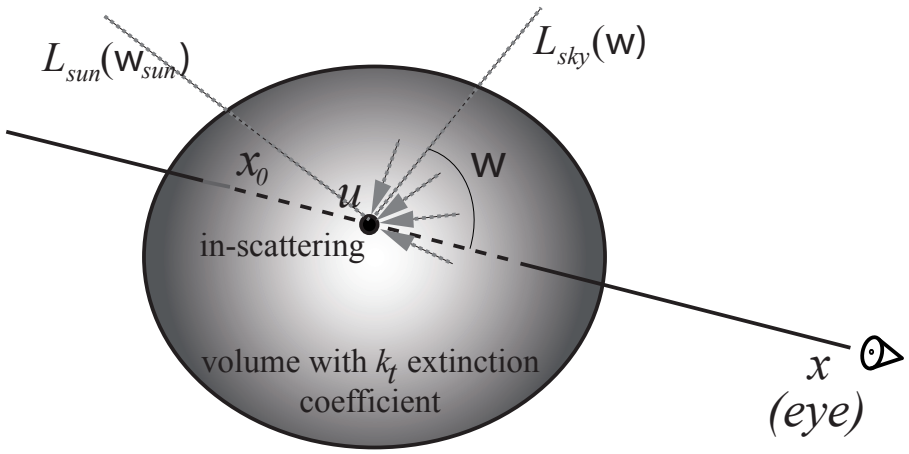


Figure 6.9: The geometry of our single scattering participating media model, illustrating the quantities of Eq. (6.5). Note that ω_{sun} is a constant in this equation.

where L_{ss} is the single-scattering radiance arriving at u. In our case we write:

$$L_{ss}(u) = L_{sun}\tau(u,\omega_{sun}) + \int_{\Omega} L_{sky}(\omega)\tau(u,\omega)d\omega \tag{6.8}$$

where Ω is the positive hemisphere of directions ω and $\tau(u,\omega_{sun})$ is the transmittance through the volume along the sun direction ω_{sun} to the point u.

The integrals of Eqs. 6.6 and 6.7 can be approximated with a standard method, similar to [Kajiya and Von Herzen, 1984], using ray traversal through the discrete voxels of the grid and numerical integration.

The participating media model we use computes radiance at a given pixel. However, since our phase function is equal to one in all frequencies, the final quantity we compute is actually irradiance. We thus write $\tilde{E}(x)$ for the value computed by Eq. (6.7).

To compute the values $\tilde{E}_{in}$ and $\tilde{E}_{targ}$, we use the discrete volume, a k_t coefficient for each voxel (both computed using [Reche et al., 2004]), and values for L_{sun} and $L_{sky}(\omega)$ using the Preetham model [Preetham et al., 1999]. At each pixel we evaluate Eq. 6.7, with appropriate values for the sun and sky illumination depending on the time of day of the input and target images.

6.2.2 Sunlight

The contribution of the sun to radiance at the eye point x is given as follows:

$$L_{vol}^{sun}(x) = \int_{x_0}^{x} \tau(u,x)k_t(u)L_{sun}\tau(u,\omega_{sun})du \tag{6.9}$$

L_{sun} is attenuated according to its respective zenith angle position, as described in [Lalonde et al., 2009]:

$$L_{sun} = \alpha e^{(-\beta m(\omega_{sun}))}N_{max} \tag{6.10}$$

where $m(\omega_{sun})$ is given as in [Lalonde et al., 2009], and is the relative optical path length through Earth's atmosphere [Kasten and Young, 1989], N_{max} is the maximum sun brightness value, α is a scale factor and β is a scattering coefficient. We used $N = 1 \times 10^6$ and α, β as described in [Lalonde et al., 2009]. The final contribution of sunlight is then:

$$L_{vol}^{sun}(x) = \int_{x_0}^{x} \tau(u,x) k_t(u) \alpha \, e^{(-\beta m(\omega_{sun}))} N_{max} \tau(u, \omega_{sun}) du \tag{6.11}$$

6.3 Efficient Relighting Algorithm

To approximate Eqs. 6.7 and 6.8 at each pixel we need to compute sun and skylight along the corresponding viewing ray. We separate the contributions of the sun and sky in Eq. (6.7) for clarity of presentation, and without loss of generality we write:

$$L(x) = L_{vol}^{sun}(x) + L_{vol}^{sky}(x) \tag{6.12}$$

The contribution $L_{vol}^{sun}(x)$ of sunlight is computed by directly estimating the value of $\tau(u, \omega_{sun})$ at a sample point u at the center of each voxel along the viewing ray, providing a discrete approximation of Eq. (6.9).

Evaluating sky illumination in Eq. (6.8) is the most expensive part of our computation. Using the naive approach discussed previously, to relight each pixel we need to sample in the order of 1024 directions in the sky, stepping through the voxelization for each sample direction. We do this for *each* voxel visited along a viewing ray. Clearly, this results in prohibitively high relighting costs. We will next show how to use spherical harmonics to precompute the accumulated effect of transmittance through the volume, both in each sky sample direction and along the viewing ray for a given pixel.

Radiance L_{vol}^{sky} due to the sky in our volumetric model is given as follows:

$$L_{vol}^{sky}(x) = \int_{x_0}^{x} \tau(u,x) k_t(u) \int_{\Omega} L_{sky}(\omega) \tau(u,\omega) d\omega du \tag{6.13}$$

where the quantities τ, ω, Ω and u are the same as in Eq. (6.6)-(6.8).

Since $\tau(u,x)$ does not depend on ω we can write:

$$L_{vol}^{sky}(x) = \int_{x_0}^{x} \int_{\Omega} \tau(u,x) k_t(u) L_{sky}(\omega) \tau(u,\omega) d\omega du \tag{6.14}$$

We can then invert the order of integration and factor out L_{sky}:

$$L_{vol}^{sky}(x) = \int_{\Omega} L_{sky}(\omega) \left(\int_{x_0}^{x} \tau(u,\omega) \tau(u,x) k_t(u) du \right) d\omega \tag{6.15}$$

We can write the inner integral as $g(\omega)$:

$$g(\omega, x) = \int_{x_0}^{x} \tau(u, \omega)\tau(u, x)k_t(u)du \tag{6.16}$$

One way to reason about the above equation is that, for a given point u along a viewing ray (in the direction $\overrightarrow{xu}$), this expression simultaneously captures the contribution in skylight sample direction ω at the point u, and the attenuation from point u to the eye-point x (see Figure 6.9).

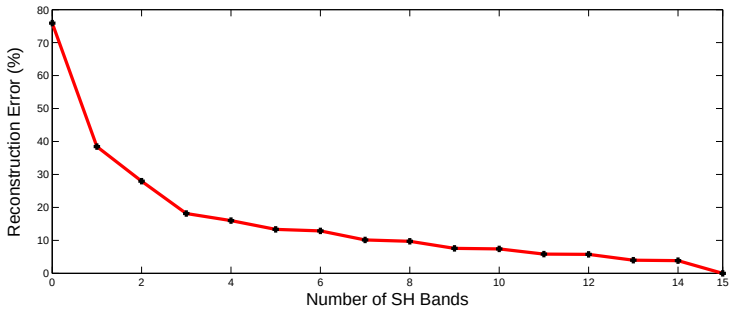


Figure 6.10: Graph plotting the reconstruction error (Y) for a given number of bands.

We precompute a discretized version of $g(\omega, x)$ and project it onto a spherical harmonic basis $\tilde{g}(\omega, x)$ using 6 bands.

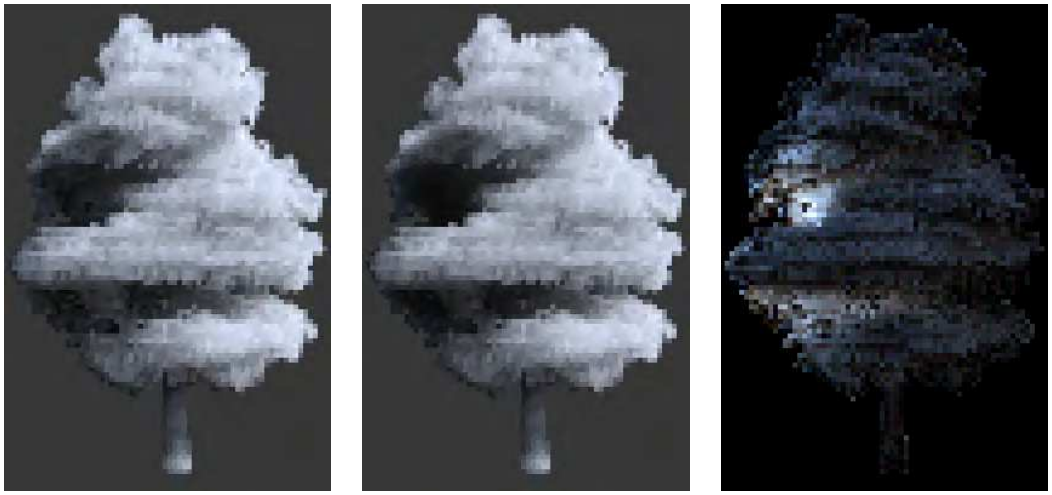
Spherical Harmonics were first introduced by Sloan et al. [Sloan et al., 2002] in the context of pre-computed radiance transfer. More recently, Jansen et al. [Jansen and Bavoil, 2010] introduced Fourier opacity mapping, in which absorption is projected along one ray onto a 1D Fourier basis to speed up the integral computation. In our case however, we project the directional (2D) distribution of transmissivities onto a sphere to allow for faster integration. This precomputation is done only once for each reconstructed tree, which allows faster relighting.

The reconstruction error using 6 bands is 12.86% in L_2 norm. See Figure 6.10 for a graph which plots the errors in reconstruction using different number of bands. At runtime, we can then simply compute a dot product of the skylight projected on-the-fly onto spherical harmonics (we could alternatively use an analytical formulation [Habel et al., 2008]), and the precomputed combined transmittance:

$$L_{vol}^{sky}(x) = \int_{x_0}^{x} L_{sky}(\omega)\tilde{g}(\omega, x) \tag{6.17}$$

This allows us to compute $\tilde{E}_{in}$ and $\tilde{E}_{targ}$ very efficiently (i.e., in the order of 3 seconds each for a 512x512 image[1]). Precomputation for a 128^3 volume required 20 minutes. However, computing the sky contribution on a coarsened 64^3 volume does not result in significant degradation, but reduces the precomputation time to 5 minutes.

[1]Timings on 4-core 2.3Ghz Xeon; only relighting is multi-threaded.



a b c

Figure 6.11: (a) Image computed by explicit sampling of the sun dome at each sample point. (b) The result of our spherical harmonics approximation. (c) Difference image (x10).

To validate the quality of this approximation, we compare the result of Eq. (6.17) to the ground truth reference in which we approximate the integral of Eq. (6.13). Ground truth is computed by explicitly sampling 1024 sky directions at each sample point of the line integral along the viewing ray. In Figure 6.11, we see that the difference is very low (note that the difference image is multiplied by x10).

6.4 Validation Results on Synthetic Trees

We first apply our model on several synthetic trees; the advantage of these tests is that it is much easier to generate reference solutions and obtain a variety of data for analysis, in addition, we can vary the location and dates of the images freely. Note that we only use the *images* generated by PBRT as input. The only difference with the real photographs is that no calibration is required for the camera, since the exact cameras are provided as input. All other steps (volumetric construction etc.) are identical to the process for real photographs.

We also investigate the effect of varying reflectance parameters (specularity, translucency), of the leaves, and the resulting effect on the performance of our algorithm.

6.4.1 Synthetic Validation Results

We show two examples taken from the $Xfrog$ (http://www.xfrog.com) European tree database. Two more examples are provided in the online Appendix located at *http://www-sop.inria.fr/members/Marcio.Cabral/thesis/*. The date used is July 22nd, and the

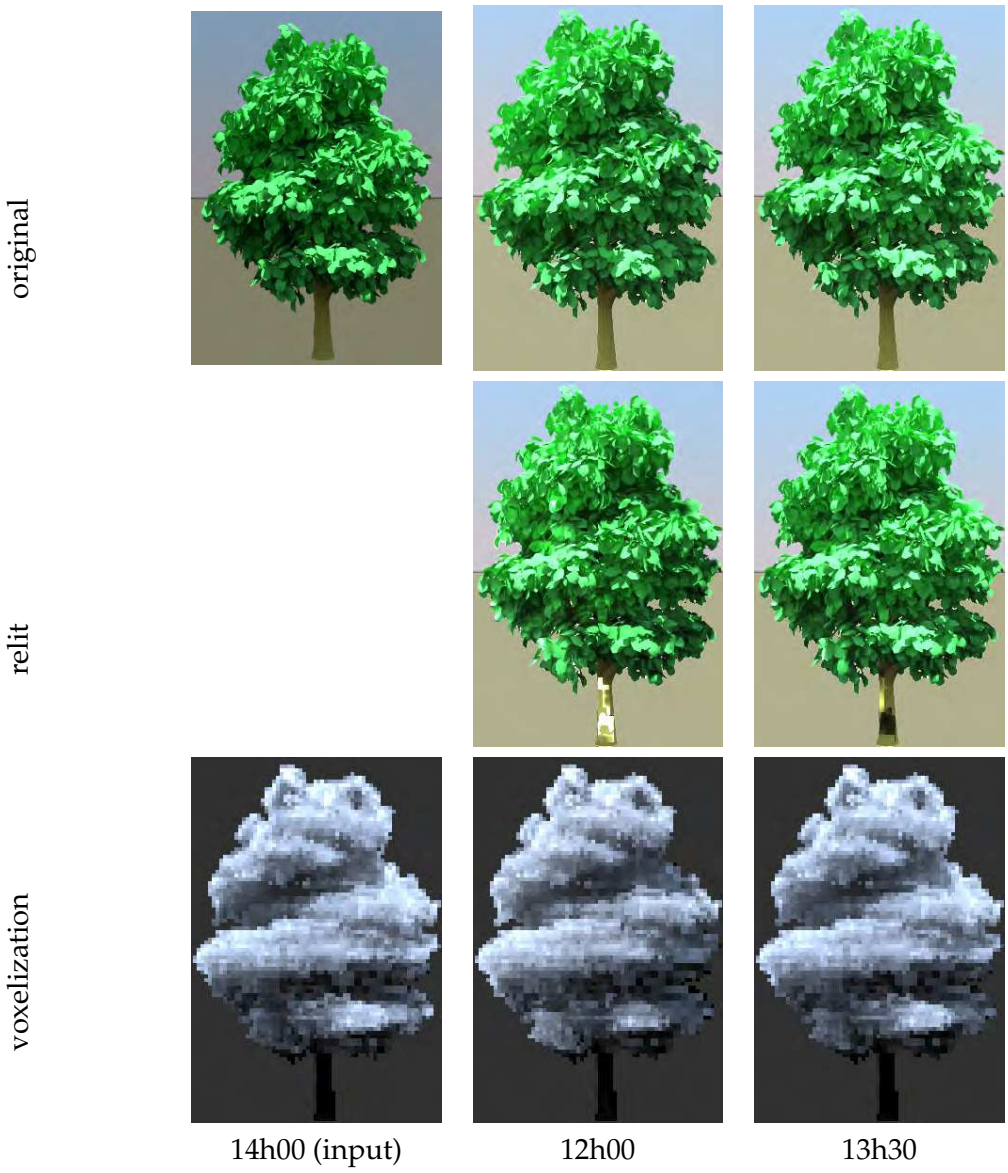


Figure 6.12: **Horse Chestnut** Top row: input image and 2 target ground truth images with corresponding times of day. Middle row: 2 resulting relit images using our approach. Bottom row: $\tilde{E}_{in}$ and the two $\tilde{E}_{targ}$ images. Two additional times can be found in the following Figure 6.13.

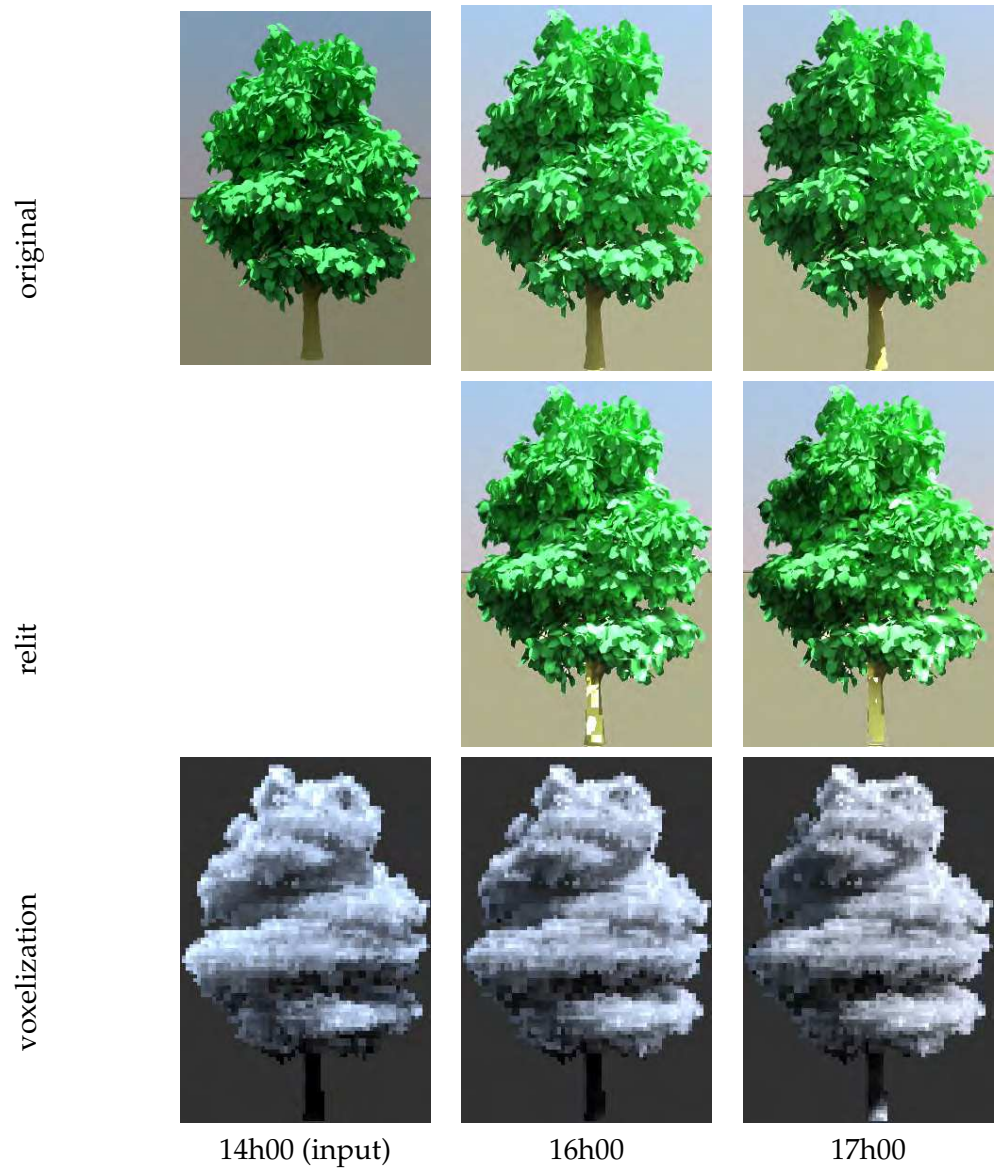


Figure 6.13: **Horse Chestnut** Top row: 2 target ground truth images with corresponding times of day. Middle row: 2 resulting relit images using our approach. Bottom row: $\tilde{E}_{in}$ and the two $\tilde{E}_{targ}$ images. Two earlier target times can be found in the previous Figure 6.12

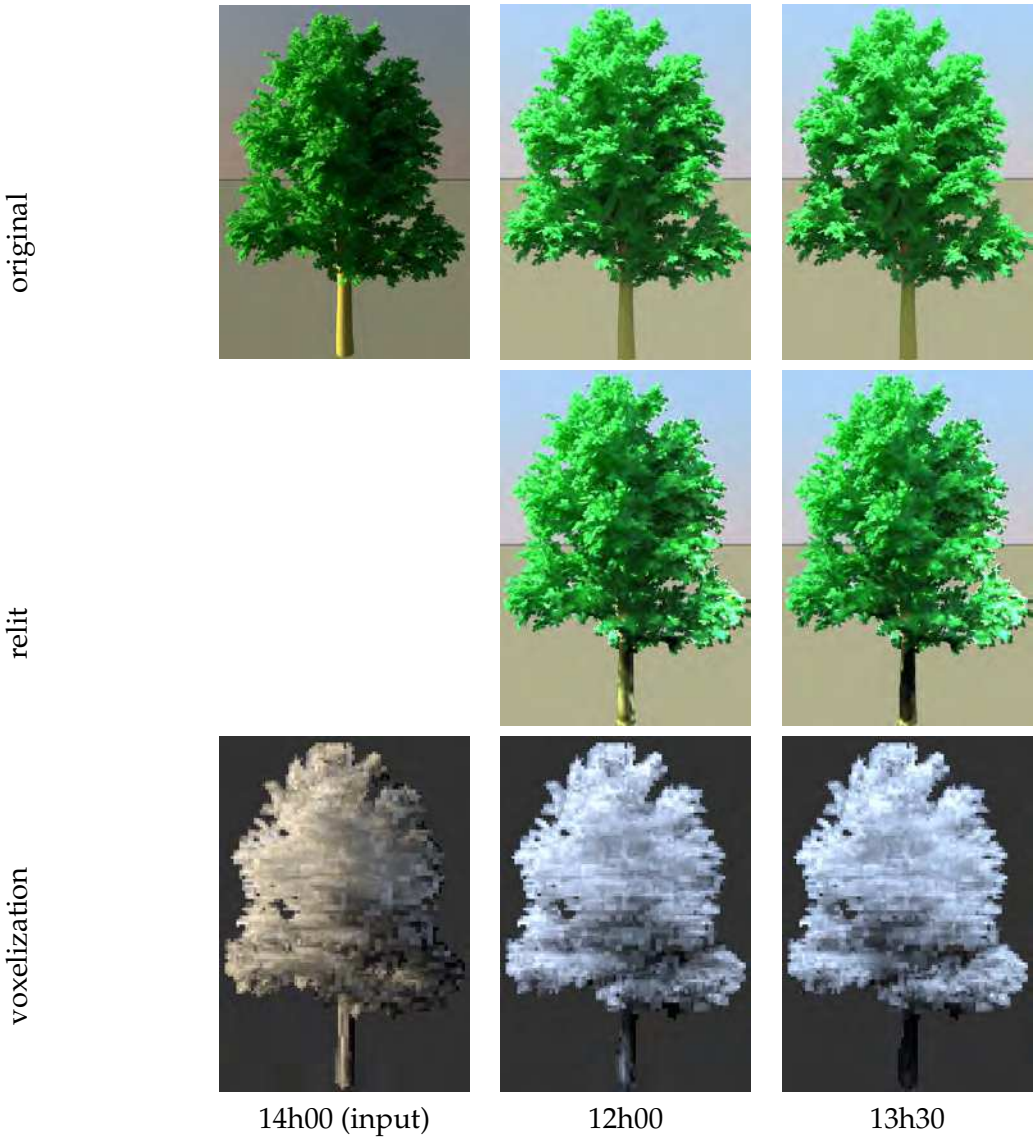


Figure 6.14: **European Beech** Top row: input image and 2 target ground truth images with corresponding times of day. Middle row: 2 resulting relit images using our approach. Bottom row: $\tilde{E}_{in}$ and the two $\tilde{E}_{targ}$ images. Two additional times can be found in the following Figure 6.14.

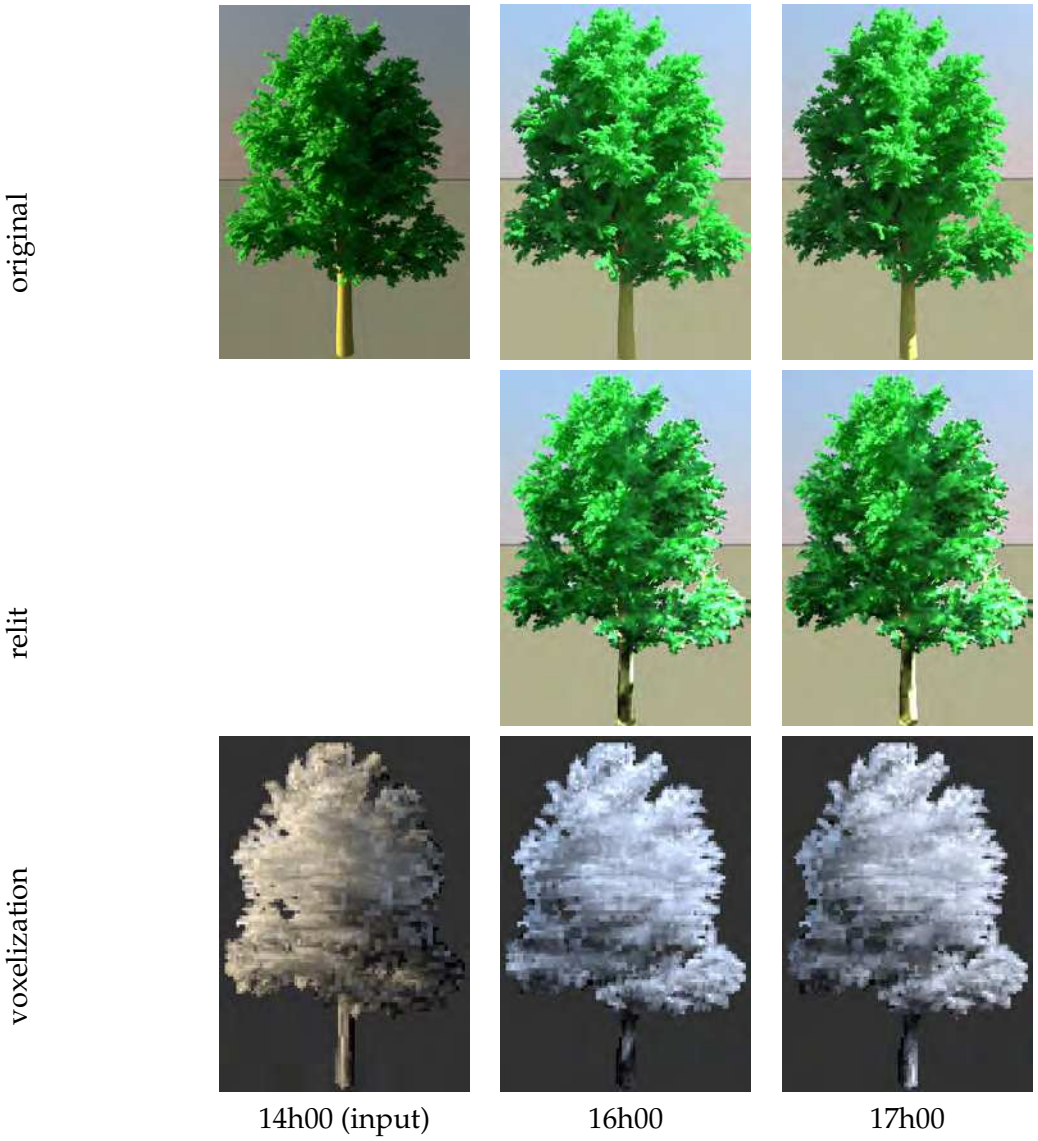


Figure 6.15: **European Beech** Top row: 2 target ground truth images with corresponding times of day. Middle row: 2 resulting relit images using our approach. Bottom row: $\tilde{E}_{in}$ and the two $\tilde{E}_{targ}$ images. Two earlier target times can be found in the previous Figure 6.14

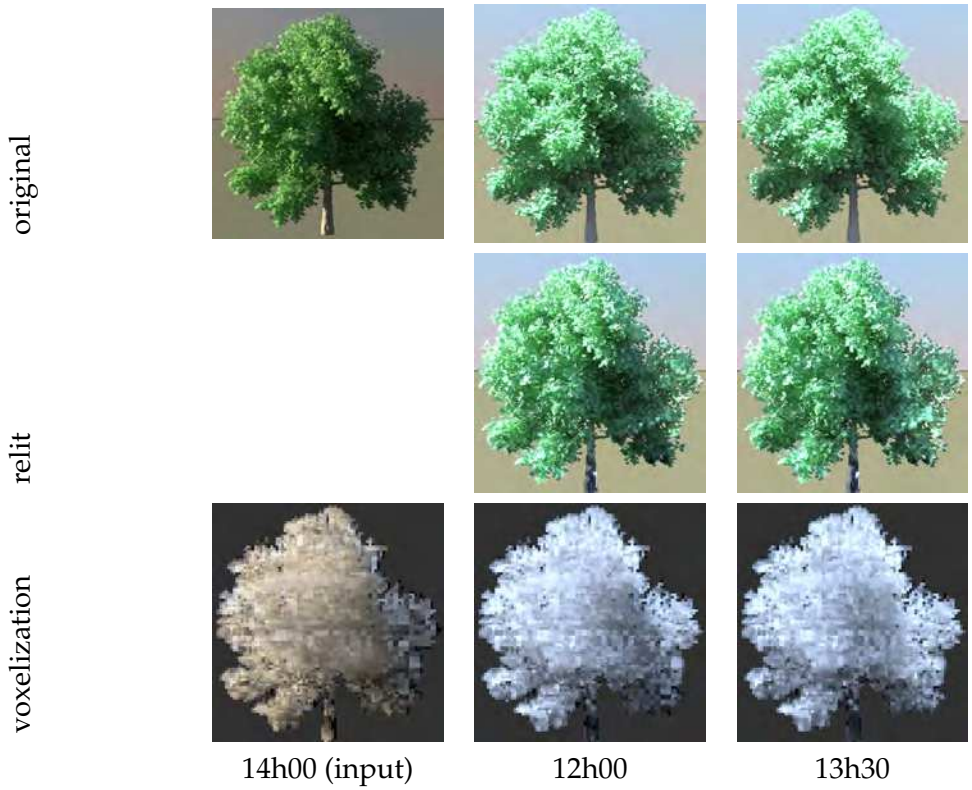


Figure 6.16: **London Planetree** Top row: input image and 2 target ground truth images with corresponding times of day. Middle row: 2 resulting relit images using our approach. Bottom row: $\tilde{E}_{in}$ and the two $\tilde{E}_{targ}$ images. Two additional times can be found in the following Figure 6.17.

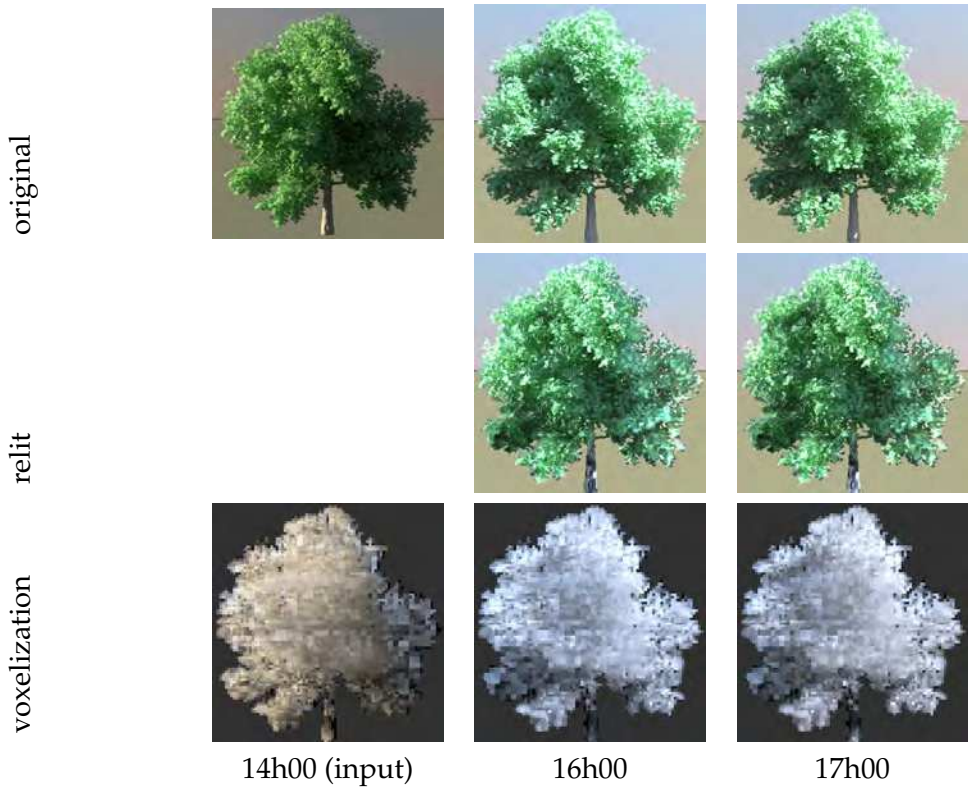


Figure 6.17: **London Planetree** Top row: 2 target ground truth images with corresponding times of day. Middle row: 2 resulting relit images using our approach. Bottom row: $\tilde{E}_{in}$ and the two $\tilde{E}_{targ}$ images. Two earlier target times can be found in the previous Figure 6.16

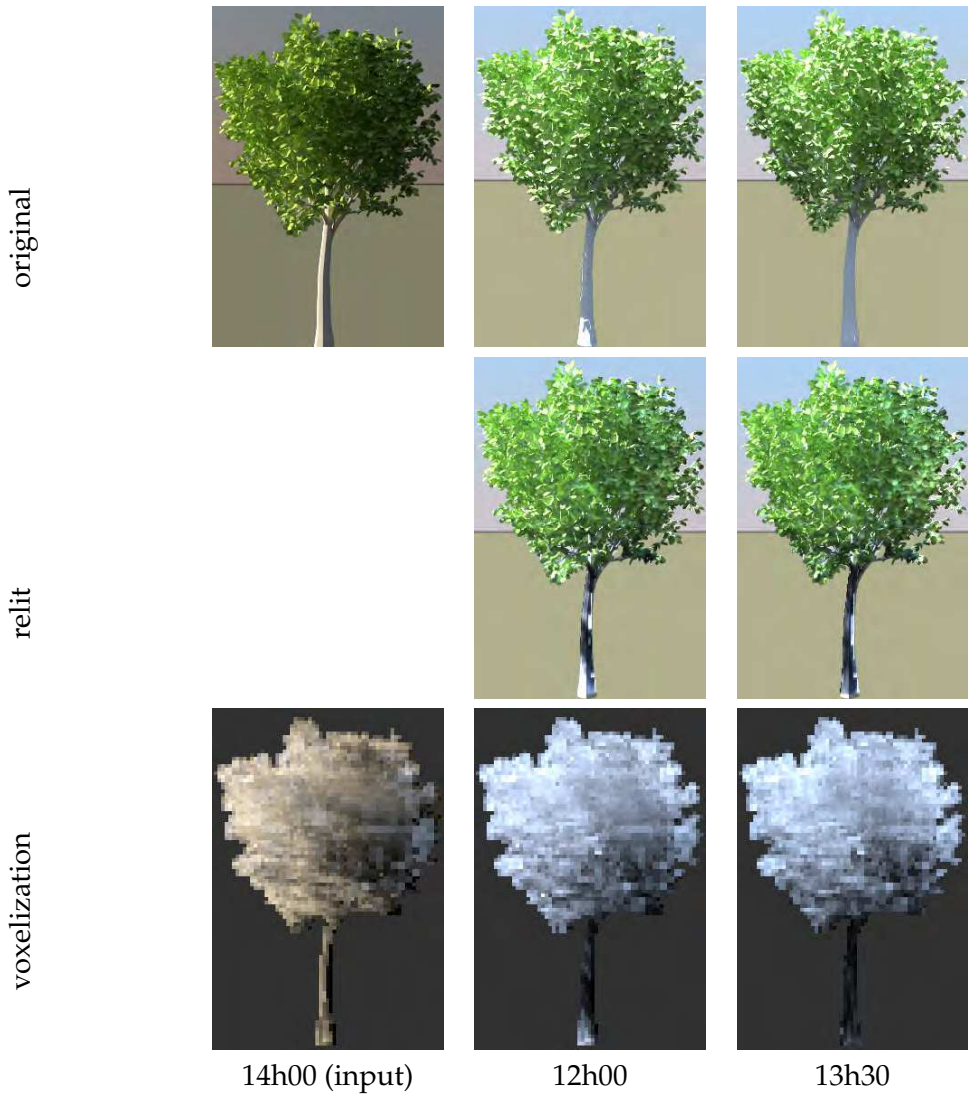


Figure 6.18: **European Mountain Ash** Top row: input image and 2 target ground truth images with corresponding times of day. Middle row: 2 resulting relit images using our approach. Bottom row: $\tilde{E}_{in}$ and the two $\tilde{E}_{targ}$ images. Two additional times can be found in the following Figure 6.19.

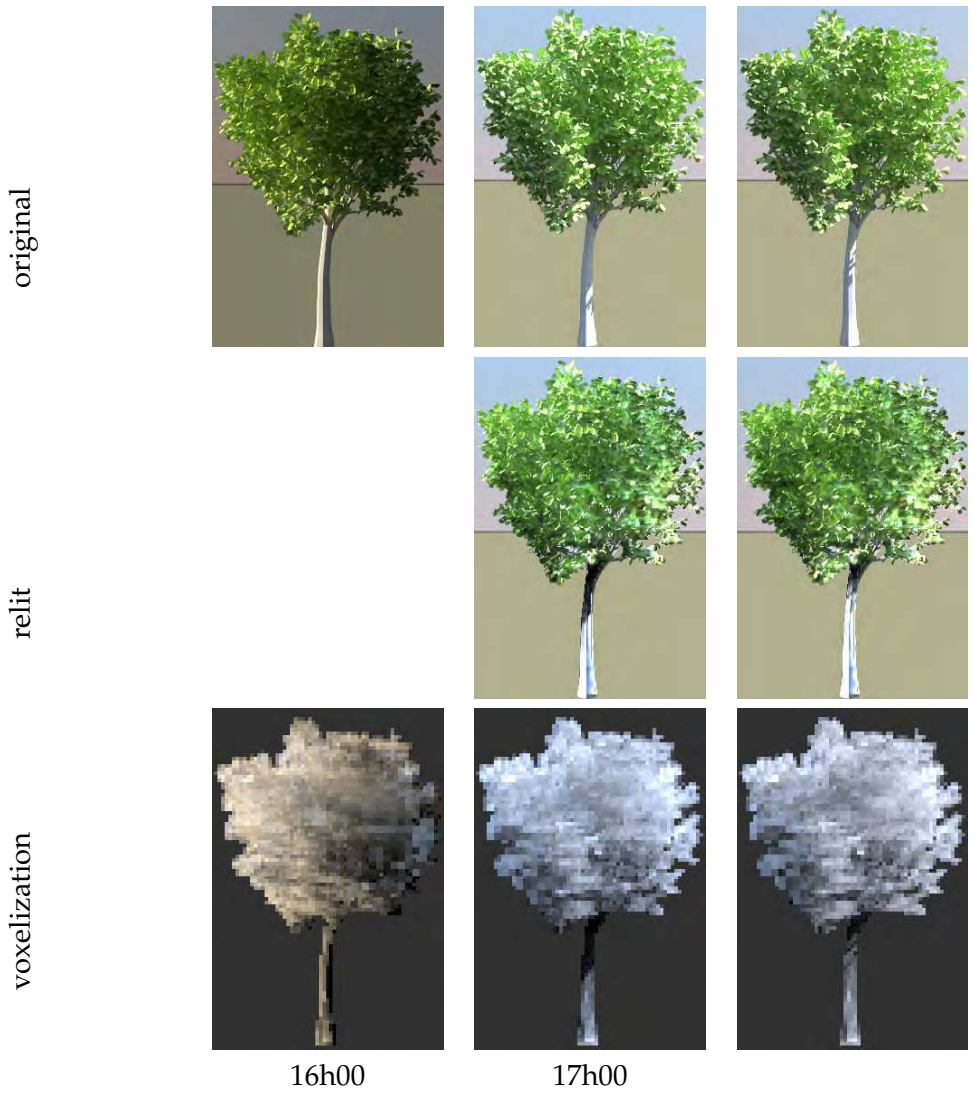


Figure 6.19: **European Mountain Ash** Top row: 2 target ground truth images with corresponding times of day. Middle row: 2 resulting relit images using our approach. Bottom row: $\tilde{E}_{in}$ and the two $\tilde{E}_{targ}$ images. Two earlier target times can be found in the previous Figure 6.18

location is East Coast US.

For each tree on the top row, we show the input image and 4 target images at different times of day. In the middle row we show the relit images using our method and in the lower row we show the $\tilde{E}$ images (Figures 6.12, 6.13, 6.14,6.15, 6.16,6.17 and 6.18,6.19 are respectively called *Horse Chestnut*, *European Beech*, *London Planetree* and *European Mountain Ash*). Please also see the accompanying video (*http://www-sop.inria.fr/members/Marcio.Cabral/thesis/*), where the movement of the sun and the corresponding illumination is much easier to comprehend.

As we can see, our relighting approach captures the overall behavior of lighting well. Evidently, the limitation of the volumetric reconstruction compared to the actual detailed geometry results in minor differences in levels of illumination. In the online Appendix of this thesis and the additional video, located at *http://www-sop.inria.fr/members/Marcio.Cabral/thesis/*, we can also see that our approach works quite well even for cases of relatively sparse trees, and for trees of different canopy shapes. For the case of the European Mountain Ash, we see slight "banding" artifacts, which are more visible in the video (see Figure 6.20). We believe this is due to the close-to-spherical nature of this particular canopy, and the consequent inability of the volumetric reconstruction to capture fine geometric details. This renders the grid structure more visible in the final results.

Figure 6.20: The European mountain Ash. (Left) Ground Truth; (Right) Relit result. The almost spherical nature of the canopy results in lack of detail from the volumetric reconstruction. Slight banding artifacts can be seen (better seen in the video - *http://www-sop.inria.fr/members/Marcio.Cabral/thesis/*).

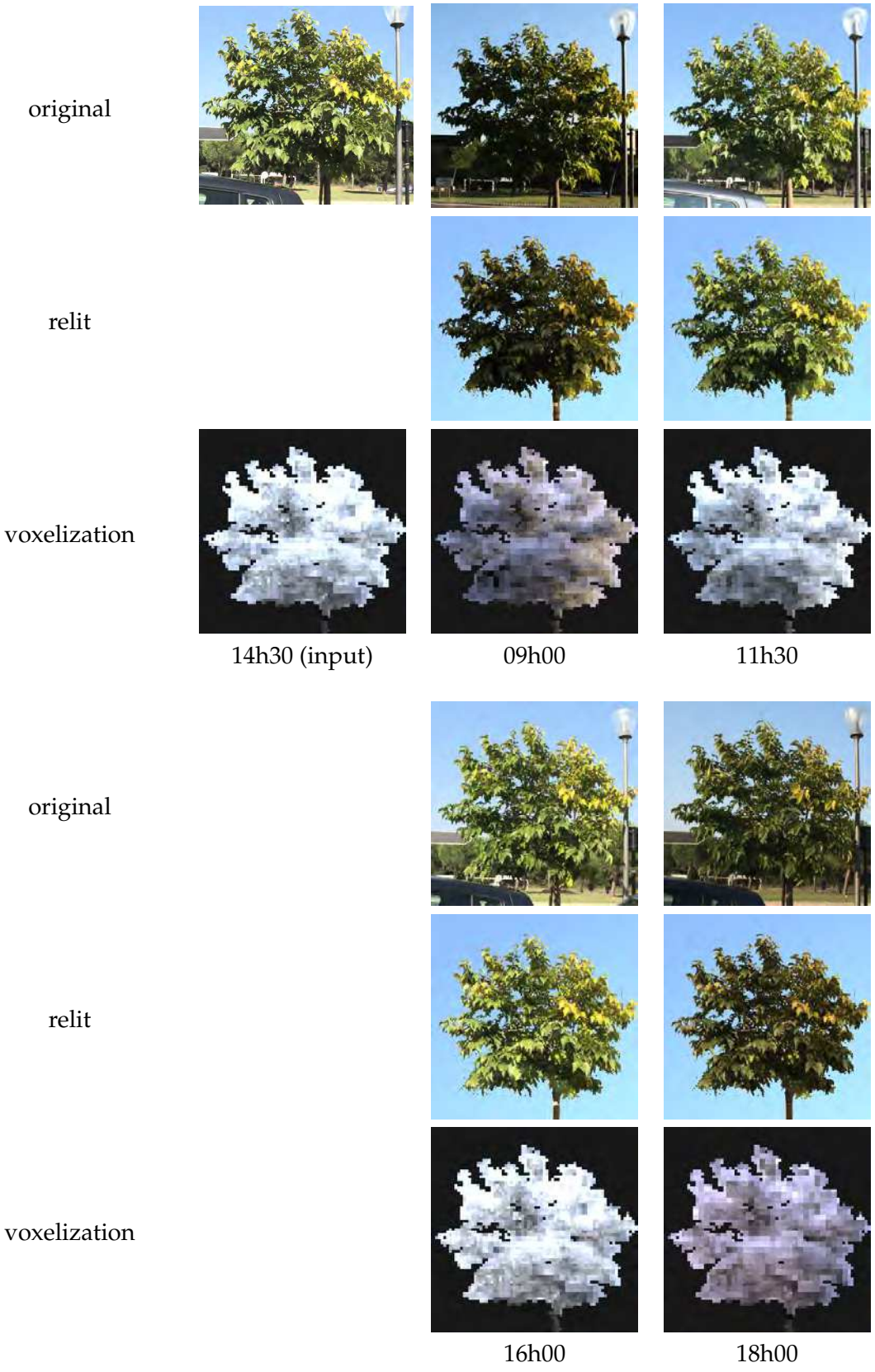


Figure 6.21: **Mulberry Tree** *First row:* input image and 2 target images with corresponding times of day. *Second row:* resulting relit images using our approach. *Third row:* $\tilde{E}_{in}$ and the four $\tilde{E}_{targ}$ images. Next three rows follow the same pattern with additional hours.


input at 15h30


ground truth at 11h00


ground truth at 12h30


relit at 11h00


relit at 12h30


$\tilde{E}_{in}$ at 15h30


$\tilde{E}_{targ}$ at 11h00


$\tilde{E}_{targ}$ at 12h30


ground truth at 16h30


ground truth at 18h30


relit at 16h30


relit at 18h30


$\tilde{E}_{targ}$ at 11h00


$\tilde{E}_{targ}$ at 12h30

Figure 6.22: **Oak Tree** *First row:* input image and 2 target images with corresponding times of day. *Second row:* resulting relit images using our approach. *Third row:* $\tilde{E}_{in}$ and the four $\tilde{E}_{targ}$ images. Next three rows follow the same pattern with additional hours.

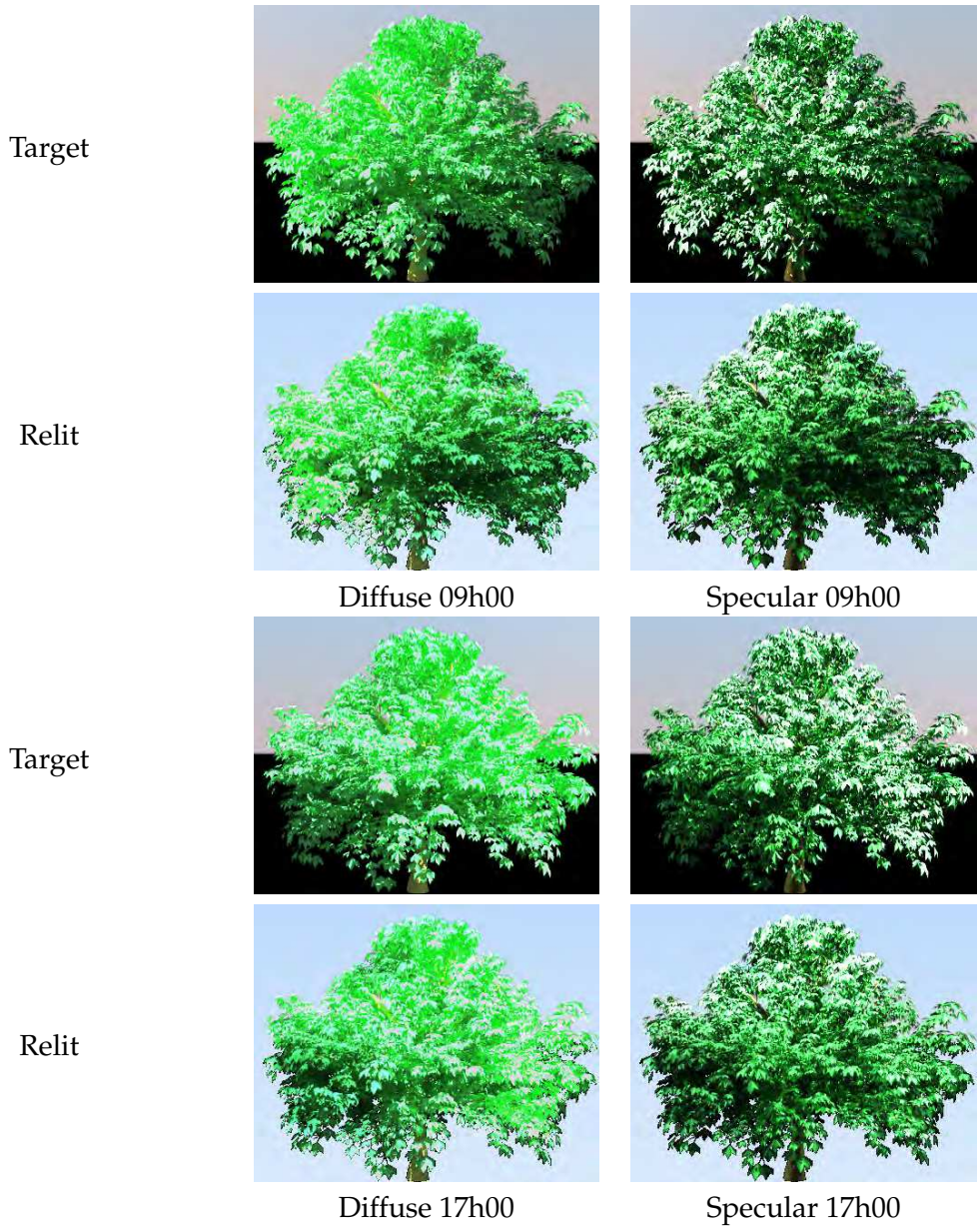


Figure 6.23: **Comparison with Specular/Non-Specular leaves** 1^{st} and 3^{rd} rows: target images with corresponding times of day for the same tree rendered with diffuse (*left image*) and specular (*right image*) leaves. 2^{nd} and 4^{th} rows: the four corresponding relit images.

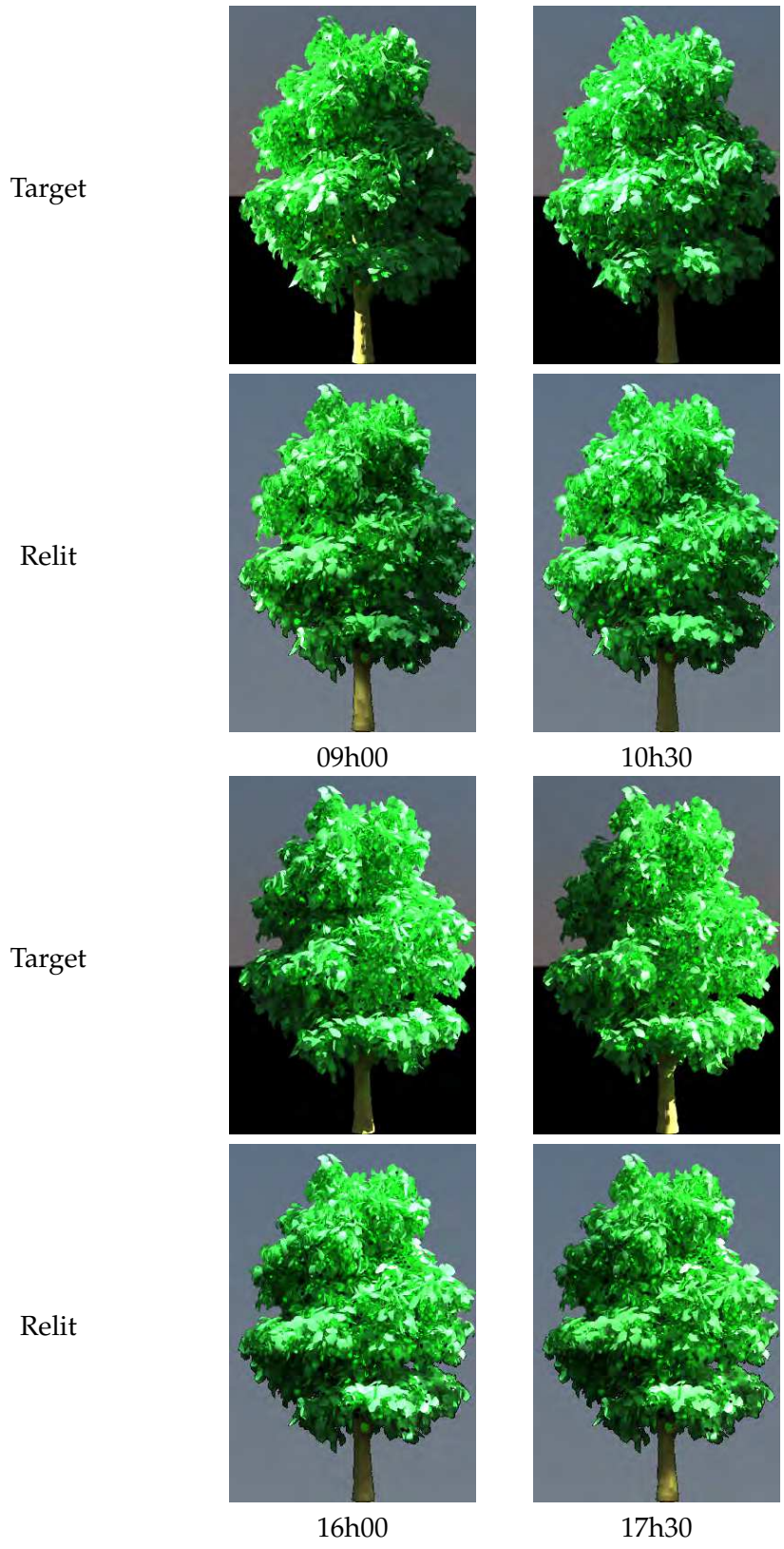


Figure 6.24: A tree canopy with a diffuse/glossy transmissive material for leaves; note that the result of our algorithm is significantly better than the specular only case.

6.4.2 Varying Reflectance Parameters

To study the effect of high specularity on the performance of our algorithm we rendered a synthetic tree in PBRT with a leaf material that contains both diffuse and glossy specular reflections. The material uses Blinn BRDF with a microfacet model for surface roughness (see [Pharr and Humphreys, 2004] for details). Glossy reflectivity was set up to be 5 $\times$ stronger than the diffuse reflectivity to exaggerate the results. Surface roughness was set up to 0.1 which indicates smaller but highly specular highlights (see Figure 6.23). From our test, we clearly see that specular highlights are not well captured by our relighting method since we do not have geometry (and normal) information.

Despite the above test, we believe that our model works well in practice for photographs because real tree leaves are in reality translucent. To investigate this hypothesis we created a synthetic tree with very translucent leaves. We rendered synthetic images of a tree using a translucent material with glossy/diffuse transmissivity for leaves - values for glossy transmissivity are 0.15 and diffuse/glossy reflection are 0.85 – see PBRT [Pharr and Humphreys, 2004] for more details (see Figure 6.24). As we can see, the results are significantly improved compared to the specular only test in Figure 6.23.

6.5 Results on Photographs

We next present our results on real photographs. We first present issues related to the procedure and implementation, then present and discuss our results.

6.5.1 Procedure and Implementation

As mentioned previously, we use the method of [Reche et al., 2004] to construct the volume. We currently use [Levin et al., 2008] for the mattes, and ImageModeler (http://usa.autodesk.com) for calibration. Automatic camera calibration using e.g., [Snavely et al., 2008] can also be used, simply requiring a larger number of photographs. Mattes would still only be required on 10-12 photos however.

We applied two modifications to the initialization described in [Reche et al., 2004]. Due to inaccurate camera calibration and the fact that we were unable to use blue screens everywhere, we observed that the algorithm culls voxels too aggressively, resulting in too sparse volumes and relighting artifacts. The first modification involves keeping voxels even if they are not present in 2-3 mattes/photos. The second modification involves artificially "densifying" the voxel reconstruction in very sparse areas. We first find voxels with less than 8 non-empty neighbors. For each such voxel, we collect its 6 axial neighbors, and we give those that are empty an extinction coefficient value corresponding to the average of the coefficients of the non-empty neighbors. The denser voxel grid significantly improves the results of relighting. Better camera calibration and matting algorithms would render this step unnecessary.

We tested a Mulberry, an Oak and a Pine tree (shown respectively in Figs. 6.21, 6.22, 6.28.) For the volume reconstruction, we used 11, 12 and 11 images respectively for each tree. The Mulberry tree was the only one where a blue screen was

used to aid matte extraction. The Oak and Pine tree locations and sizes prevented us from using a blue screen. The values of k_t computed using [Reche et al., 2004] can be modulated by a global scaling factor s. For the three examples used, we chose s so that the E_{in} image best represents the input lighting (see first column of Figures 6.28-6.22). The values for s we used were 0.95, 0.90 and 0.85 for each of the three trees respectively, and account for the density of leaves in each tree. In some cases the resulting masks have small imperfections, and additional manual editing is required. In all the cases shown here, manual editing of masks required less than 5 minutes, for each dataset (i.e., all 11-12 images). We used a Canon EOS 5D camera, and performed all processing on linearized, 12-bit ".RAW" images. HDR images of trees (composed using different exposures) are hard to capture because of the inherent motion of leaves due to wind. We found that the 12-bit images contain sufficient dynamic range for our method. Care has to be taken to ensure relatively high-quality camera calibration, otherwise the volume reconstruction is unsatisfactory and will not give good relighting results.

Photos are always captured at a single time of day. The camera is positioned around the tree, with the tree canopy at the focal point. The viewpoints around the tree ($10 - 12$) were chosen in a way to minimize occlusion and interference of nearby objects. When possible, blue screens were utilized to provide a better matte from vegetation behind (see Figure 6.8). For each shot a compass was used to determine the viewpoint direction, and a standard GPS provided coordinates to determine the spatial location of the tree. These data were then used to compute the sky and sun models.

For the Preetham sky model, we use a turbidity parameter of 4.2 which appears to work well in all cases. All photos were taken on September 23rd and 24th.

Since our method is based on the Preetham model, sky luminance values early in the morning or late in the evening can be inaccurate, as mentioned in [Zotti et al., 2007]. We alleviate this problem by making the sun brightness decay towards sunset and sunrise more strongly. To do this, in Eq. (6.11), we change the relative optical path $m(\omega_{sun})$ to $m(\omega_{sun})^k$, where $k = 2.5$ for the red channel and $k = 2.0$ for the green and blue channels. This heuristic yields good results in all experiments, significantly improving the quality of the results. An example is shown in Figure 6.25.

The quality of our results is degraded if the input data used lies within this period of the day, as can be seen in Figure 6.26 - *middle column*. Our experiments indicate that to obtain optimal results with our method, it is best to take input images between 12am and 3pm.

6.5.2 Results

We took sparse "time-lapse" sequences of the three trees to provide ground truth references, from 9am to 6pm over one hour intervals. In Figures 6.21, 6.22 and 6.28 we show the results in the top row, and the ground truth photographs in the second row. We show four times here (indicated in the figure); the entire sequences are provided in the additional material and in the accompanying video (*http://www-sop.inria.fr/members/Marcio.Cabral/thesis/*). As in the case of the synthetic validation,

(a) (b) (c)

Figure 6.25: (a) Target image at 17:30. (b) the relit image - the color shift is due to the inaccuracies of luminance values in the Preetham sky model for the early evening. (c) A simple correction factor brings the result much closer to the target.

watching the video provides a better sense of the moving sun.

As we can see the quality of the relighting results is satisfactory. Our diffuse reflectance assumption, and the consequent volumetric approximation appears to work well on the examples tested, despite its apparent simplicity. From the results we see that the quality for the real photographs is on a par with that of the synthetic trees, despite the inaccuracies in camera calibration and the non-diffuse nature of the real tree leaves. In addition, even for the case of a relatively sparse tree (the Mulberry example, but also the Oak to a lesser extent), the results are of high quality.

The quality of the volumetric reconstruction does affect the results. A blue screen was used on the Mulberry tree and we can see that the results are slightly better. Adverse capture conditions such as the impossibility to use a blue screen to aid matte extraction due to size and the difficulty of calibrating cameras using scene features (see Fig 6.27 for the capture conditions of the Pine tree) produces somewhat lower quality relighting results as can be seen in Fig 6.28.

We believe that improvements in vision and image processing algorithms, which are beyond the scope of this thesis, will allow the creation of better mattes and higher quality camera calibration, thus improving the results of our approach, and removing the need for the heuristics for the volumetric reconstruction.

Qualitative comparisons between the real trees and the synthetic results show that our approach works slightly better on the real trees photographs. We believe that this is due to the quality of the geometric model used for the synthetic renderings. Although each synthetic tree is on the order of $300K$ triangles, individual leaves were flat, containing only a few faces. As a result a lot of the geometric subtlety is missing, which allows the volumetric reconstruction method to produce good results.

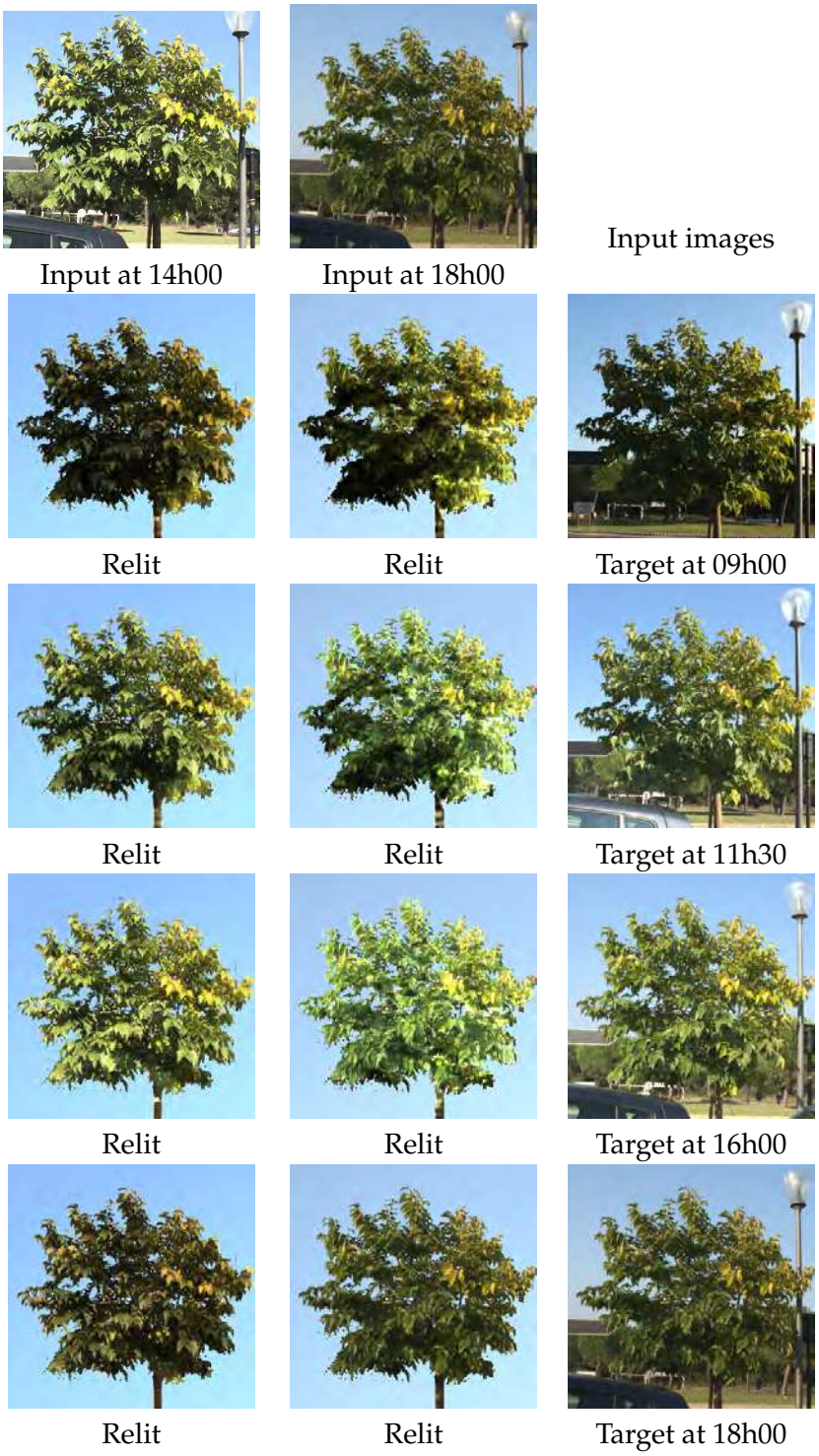


Figure 6.26: **Mulberry tree** In contrast to Figure 6.21, in this example we also show relit results when the input data photographs were taken at different times. 1^{st} *row:* input images taken at different times of the day: 14h00 in 1^{st} column and 18h00 in 2^{nd} column. Following rows show in the 1^{st} column: relighting results using the input image taken at 14h00; in the 2^{nd} column relighting results using the input image taken at 18h00; in the 3^{rd} column shows ground truth photographs for comparison.

Figure 6.27: Capture conditions for the Pine Tree (Figure 6.28).

6.5.3 Limitations

Our method is intended for distant cameras only, such that the entire canopy is present in the image. We thus propose a solution to the specific case when no 3D information can be reliably captured in a pixel accurate way, and robustly enough to handle small variations in the geometry (e.g., presence of slight wind). If the camera gets close enough to the tree to be able to perceive large regions of individual leaves and branches, our volumetric assumption would not hold.

It would be possible to use a high-dynamic range (HDR) light-probe to capture the *input* sky. However, the advantage of using the Preetham model is that it is available for both the input and the target times. Such an additional capture overhead is thus unnecessary for our approach, and would defeat the purpose of our method which is to require no lighting information at the target time. Additionally, using different models for the input and target skies will inevitably lead to inconsistencies and probably give worse results.

We assume clear skies in our method since these are well simulated by the Preetham model [Preetham et al., 1999]. To our knowledge, overcast skies cannot be accurately and consistently represented to allow our ratio-based approach to work.

Due to the small number of SH bands used for reconstruction of natural skies created by the Preetham model, our method is expected to produce lower quality results for higher frequency lighting such as those encountered in some high frequency captured HDR environment maps. Additionally, our method requires consistent *input* and *target* lighting models to work. Since our method focuses on minimizing the capture procedure to a *single* time of day, there is no captured HDR available for the target time.

Although our method can relight a canopy for different lighting conditions, such as winter or summer sun positions, our method cannot simulate changes in the leaf structure of the tree canopy that occurs between two extremely different seasons - this would require explicit geometric reconstruction of leaves and branches, and appropriate modification.

Finally, our method depends heavily on the quality of the volumetric reconstruction. Irregularly shaped trees with complex isolated branch structures pose a difficult problem for reconstruction, and thus our relighting results are of somewhat lower quality. This can be seen in Figures 6.14,6.15 at 14h30 and 17h00.

6.6 Discussions and Conclusions

Other approaches could be envisaged to solve the tree canopy relighting problem. For example, texture synthesis could be used rather than a ratio, in which we would search for similar luminance pixels in the input. Our experiments with such an approach showed that the visual quality was not as good as that presented here, since the synthesis stage alters the image and thus degrades the overall quality of the result. A hybrid volumetric/geometric approach can also be considered, in which 3D leaf positions of pixels are estimated to perform relighting. We also experimented with this idea, and despite promising initial results, extensive tests showed that the pixels positions could not be reliably estimated in the general case.

Concerning the application in Figures 6.2(a)-(b), it could be argued that simple histogram transfer would suffice. As we can see in Figure 6.29(a), colors are correctly reproduced, but lighting is incorrect: the right side of the tree should be in shadow. This is correctly reproduced by our approach (Figure 6.2(b)).

While beyond the scope of this paper, it would of course be highly desirable to be able to relight the entire environment including the tree canopies. We consider this to be important future work, starting for example with shadow removal (including the cast shadows from the tree canopies and trunks) and then treating general relighting. We also expect our method to have somewhat reduced performance on very sparse trees, or trees where leaves have a very strong preferred orientation.

The volumetric model should be applicable to all photographs of materials which are either truly volumetric (e.g., clouds or smoke), or have behavior which is similar to a volume (e.g., large collections of small objects or grass). While the model presented here is not directly transposable on all geometries and all scales, we do believe that some of the ideas presented here could well generalize to non-volumetric materials. In addition, we believe that our approach could fit well with more traditional approaches such as [Debevec, 2002], resulting in a general and complete relighting method.

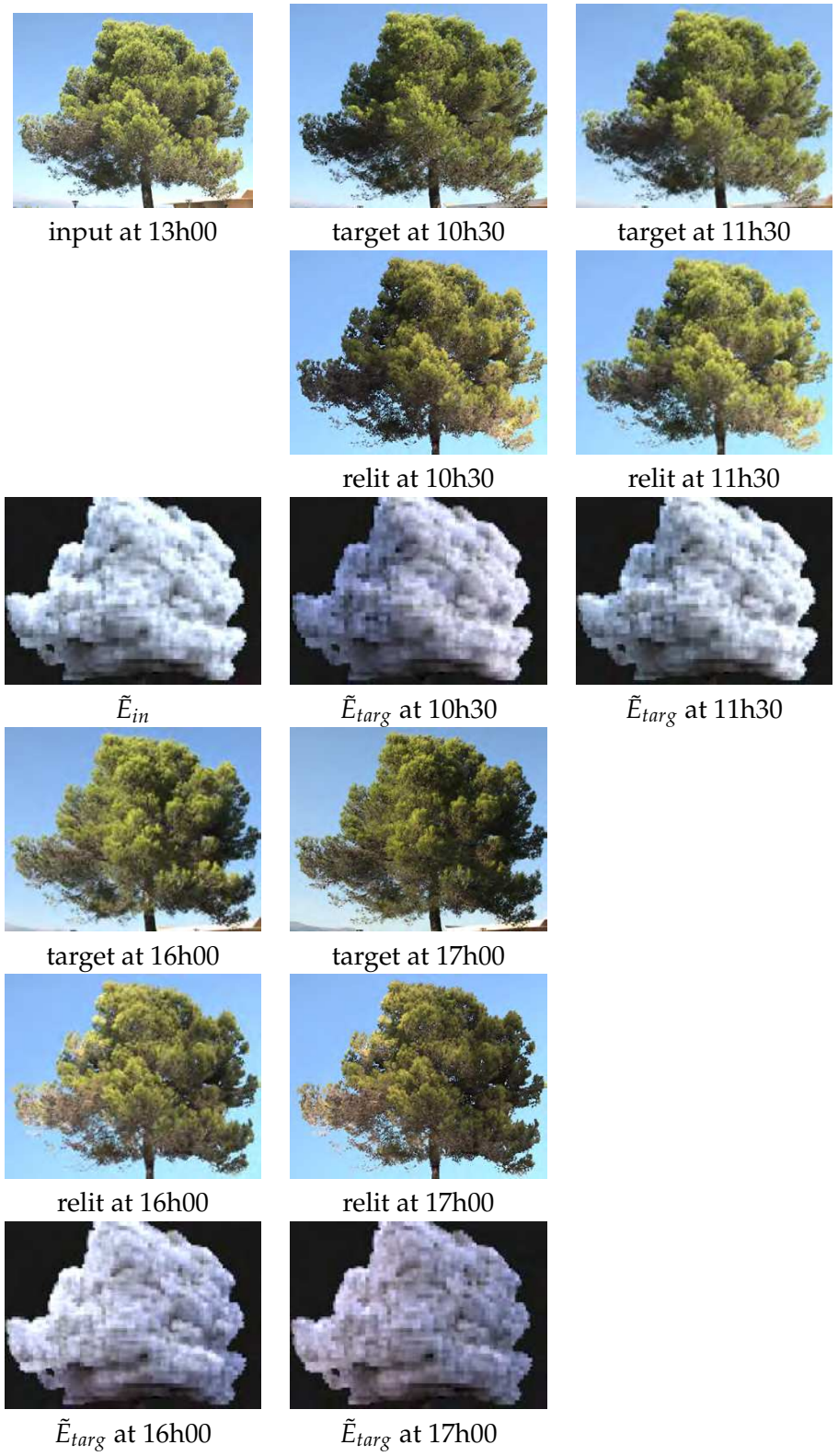


Figure 6.28: **Pine Tree** *First row:* input image and two target (ground-truth) images with corresponding times of day. *Second row:* resulting relit images using our approach. *Third row:* $\tilde{E}_{in}$ and the four $\tilde{E}_{targ}$ images. Next three rows follow the same pattern with additional hours.

(a) (b)

Figure 6.29: (a) The input image has been pasted to the background after an histogram transfer. Although the colors are correctly reproduced, the overall lighting direction, and corresponding shadows, are not well captured compared to our method (b).

CHAPTER 7

Conclusions and Future Work

This thesis provides novel contributions towards to the goal of creating new digital content from existing data. Our contributions span three areas which are important in achieving this goal: interactive geometry editing, image-based relighting and Human Computer Interaction. In each case, we have provided solutions which will contribute to the development of new tools which will hopefully make such content creation easier in the future.

7.1 Interactive Geometry Editing

Our mesh editing solution provides a new approach for creating new content interactively. Structured man-made objects and architectural 3D models can be reshaped by simply dragging and dropping vertices. Our approach also handles underlying texture, reshaping it appropriately as well. Each of these reshaped models can then be connected, allowing the creation of new models. This approach is rapid and it allows for fast construction of complex models. These tasks, if done manually using common 3D editing tools, would require a lot of effort, even from experienced users. On the other hand, our algorithm enables inexperienced users to perform such operations. Our approach automatically extracts a series of linear equations that characterizes the overall shape of the 3D model. During run-time, as the user moves vertices to achieve the desired reshape, a least-squares solver solves these equations, preserving the original aspect of the 3D model while also satisfying the user's motion. The reshaped mesh triggers deformation of the texture faces, in a similar fashion, also using a least-squares solver.

We demonstrate our approach through several scenarios, using in-house models but also models created specifically for games, which are widely available online. This is a major advantage, that allows users to expand their ability of creating new content from previously created models. Another key feature is the reshaping of textures and geometry concurrently which had not been proposed prior to this work. Our directional autosimilarity measure is able to identify areas that can be stretched, such as stochastic regions, and areas with structured details, which should be rigid for deformation. Other examples we applied our algorithm include road and bridge construction based on a few basic pieces.

7.1.1 Future Work

We would like to extend our approach to general geometries as opposed to architectural and structured man-made objects. Recent work that address the reshaping of general

geometries such as iWires [Gal et al., 2009] proposes an interesting direction.

Currently, the coupled editing of geometry and texture is unidirectional: geometry reshape triggers texture deformation. It is straightforward to explore this event in the other direction, by introducing geometry constraints based on the underlying rigidity of the texture.

7.2 Human Computer Interaction

The first results for interactive geometry editing showed us that although our tool performed well, users struggled with its 2D interface. Therefore, we investigated further how to address the problem of transposing our system to an immersive setting. We achieved this by creating a novel 3D interface and allowing users to interact with the system in a fully immersive VR environment utilizing simple command gestures.

The immersive system developed allowed simple conceptual design of architectural models with the possibility of exploring different lighting configurations by changing the day and time of the year. A simplified global illumination algorithm simulated lighting interaction within a concept design of a house allowing users to simulate how sun light would contribute to the overall illumination of the scene. An informal user study showed that novice users to an immersive system found the experience and interactivity rewarding and interesting.

Our system is fully immersive and it allows simple conceptual design of textured geometric models. It is the first to also include basic lighting design. Three different modes allow users to interact at different scales. One of them, called Mixed mode is a variation of an interaction mode based on the WIM [Stoakley et al., 1995] paradigm and it provides both local and global views of the 3D world being manipulated. Users who experimented with the system equally used all three modes when performing more open tasks. Overall, users enjoyed using the system. Moreover, users with no experience in VR immersive systems were able to use our system after a short 10 minute training session.

7.2.1 Future Work

The ultimate goal for such immersive editing of 3D geometry and textures is to completely eliminate the need for 3D input devices, such as the joystick used. We hope to use novel capture devices in the future, such as finger and hand tracking systems. We began experimenting with these devices at a final stage of development of these thesis but found them to be fragile, making them uncomfortable for regular usage.

Other avenues include extending the global illumination algorithm to more complex models, investigating how different levels of realism affects user's perception of the system.

7.3 Image-based Relighting

We have proposed a method for tree canopy relighting. Using only photographs taken at a single time of the day, our algorithm can relight the tree canopy to any other time, or day. Our method faces the problem of relighting by interpreting the tree canopy as a volumetric entity. As such, a single scattering volume rendering technique was used to estimate lighting parameters for the tree canopy. With these parameters estimated per pixel for the input image, our method estimated the same parameters for the target lighting condition and performed relighting by using a simple image ratio method. Our algorithm worked well for a variety of trees. We validated our method using synthetic renderings of trees as well as real times, with time-lapse sequences taken throughout the course of a day.

7.3.1 Future Work

An important area of future work is the development of solutions for relighting for the whole environment and not only tree canopies. As a first step, shadows need to be removed appropriately from the surroundings. The more challenging problem is to solve relighting for the whole environment which includes different types of objects with different characteristics. As such, a hybrid solution can be proposed where several approaches are used to relight different areas of the scene.

7.4 Concluding Remarks

We have seen lately a breakthrough in terms of quantity and quality of content created by users everywhere, including novice users to computer systems. In particular, the solutions proposed in this thesis provide users with intuitive ways to create textured geometry from existing pieces, including in an immersive setting, and to manipulate lighting in photographs of trees. We hope that this thesis is a step forward towards providing users with better tools for creating content that would otherwise be hard to create manually. Ultimately, our goal is to empower users with abilities allowing them to easily create and manipulate 3D content, without the need for talent and lengthy training. When this goal is achieved, their creative imagination will no longer be limited by technology.

APPENDIX A

Traduction en Français

"When a shape creates beauty
its own beauty justifies it."
Oscar Niemeyer

Contents

A.1 Introduction

Sculpture, peinture, modelisation ainsi que d'autres formes d'art ont existé depuis aussi longtemps que l'humanité existe. Des artistes, dans leur domaine particulier, se sont exprimés à travers l'utilisation de ces différentes techniques, qui requièrent de la dextérité ainsi qu'un sens aigu à tirer le meilleur parti de chaque outil. L'avènement récent des systèmes informatiques a introduit de nouveaux outils, qui permettent aux artistes de s'exprimer en utilisant d'autres médias. Les images générées par ordinateur ont particulièrement gagné en attention, étant donnée les possibilités qu'elles offrent de créer des mondes virtuels très réalistes, ou encore d'augmenter la réalité avec des objets virtuels.

En informatique personnelle, l'infographie s'est dotée d'outils et de techniques aptes à toucher un plus large public, offrant aux Ônon-artistes" la possibilité de créer de une forme d'art leur étant propre. Cependant, la plupart des utilisateurs n'ont pas encore acquis suffisamment de dextérité et de capacités pour créer du contenu qui soit à la fois attrayant et utile pour eux.

Pour combler cette lacune, c'est à dire fournir des outils qui peuvent être utilisés par des utilisateurs non-experts afin de les aider à la création de contenu pour les mondes virtuels, nous allons présenter de nouvelles solutions en explorant à la fois la création de contenu assistée et des techniques d'interaction qui tentent de diminuer la barrière entre les utilisateurs et les dispositifs d'interaction actuels basé sur des métaphores 2D.

Plus précisément nous ciblons deux fronts différents: édition de maillage texturés architecturaux, et ré-illumination de photographies. Pour le premier, nous élaborerons d'abords un cadre de travail général permettant une édition simplifiée, puis nous proposerons une solution dans un environnement immersif. Pour le second, nous nous concentrerons sur le cas de la cime des arbres, qui, comme nous le verrons, présentent plusieurs propriétés intéressantes. Nous commençons cette introduction en présentant les principales motivations de notre travail, suivies par la présentation de nos objectifs centraux. Nous terminerons ensuite ce chapitre par un bref aperçu des techniques qui sont explorées et développées dans cette thèse.

A.2 Motivation

Nous voulons fournir aux utilisateurs des techniques et des outils qui maximisent leurs capacités à créer un contenu qui, autrement, exigerait des compétences artistiques. Notre thèse se focalise dans deux directions possibles: assemblage complexe de contenu existant pour créer de nouveaux modèles à partir de pièces de base, et, paramétrage et extraction des données d'éclairage contenues par des photos pour permettre leur édition.

A.2.1 Modélisation interactive facile

Il est difficile pour les utilisateurs non-experts de traduire leurs idées créatives afin de créer des objets physiques, ou au sein d'un programme de création de contenu numérique, sans les compétences requises pour les mettre en pratique. Même si l'on considère, par exemple, la tâche des changements à petite échelle sur un modèle 3D pré existant, cela requiert un effort considérable de la part d'un artiste 3D expert pour réaliser de telles modifications dans les logiciels de modelisation commerciaux existants. D'autre part, un petit nombre de personnes qualifiées, à savoir les artistes, mettent quotidiennement à disposition une grande quantité de nouveaux contenus, allant de la photographie au modèle 3D, qu'ils créent et partagent souvent gratuitement. Dans un monde idéal, les utilisateurs seraient en mesure de modifier de façon interactive ce contenu, dans ses divers formats et types de données, de la manière qui leur plaît. Les utilisateurs pourraient réaliser ce qui leur serait autrement impossible étant données leurs compétences limitées. Du contenu entièrement nouveau pourrait être créé en prenant ce qui a

déjà été développé. Dans un tel scénario idéal, il serait possible de créer du contenu entièrement nouveau en prenant et en modifiant et en assemblant des morceaux différents.

Une autre problématique importante est que la taille des environnements virtuels augmente à un rythme sans précédent, tant en termes d'étendue spatiale que de quantité de détails. En conséquence, la plupart des difficultés dans la production d'applications interactives se trouvent désormais dans le traitement des contenus graphiques. Cela est particulièrement vrai avec les environnements en ligne proposant des espaces de plusieurs kilomètres carrés.

Dans le même temps, de nombreux jeux et applications en ligne permettent aux utilisateurs de produire du contenu supplémentaire (appelé contenu créé par l'utilisateur). Cela a favorisé la mise en place de grandes communautés de ce qu'on appelle modders, et des milliers de niveaux de jeu supplémentaires, ainsi que des objets et des personnages peuvent être trouvés sur Internet pour les applications les plus populaires. Par exemple, les joueurs aiment souvent créer des niveaux de jeu par leurs propres moyens. Quand les possibilités offertes par un jeu semblent épuisé aux yeux de l'utilisateur, de nouveaux niveaux, créés par l'utilisateur lui-même, peuvent ouvrir de nouvelles possibilités de jeu. D'autres possibilités incluent la conception architecturale, où les utilisateurs veulent visualiser le résultat de leurs idées de rénovation. Par exemple, augmenter la taille de la salle de séjour tout en conservant la surface de la cuisine.

Ces deux tendances rendent les approches de modélisation à base d'exemples très intéressantes: L'idée est de permettre aux utilisateurs de créer des environnements complexes à partir d'un ensemble d'éléments fondamentaux. Ces approches ont été appliquées avec succès à la création d'avatar et de personnages [EA, 2008], mais aussi d'objets [Funkhouser et al., 2004, Kraevoy et al., 2007]. Toutefois, elles n'ont jamais été appliquées à grande échelle à des environnements architecturaux, en dépit de son attrait : les artistes informatiques économisent beaucoup de temps en produisant de grandes quantités de contenu à partir d'existants élémentaires, et les utilisateurs non-experts peuvent facilement utiliser un tel système, tant qu'ils des exemples leurs sont fournis - exemples qui abondent en ligne. Malheureusement, les approches pour l'édition de maillages qui fonctionnent bien sur les découpages fins d'objets non texturés, ne parviennent pas à se généraliser aux bâtiments et aux environnements très structurés [Sorkine et al., 2004]. La raison principale est que les propriétés souhaitables pour un objet classique ne sont pas celles qui doivent être conservés dans un modèle architectural.

A.2.2 Édition de photos et textures

On peut dire Il est vrai que les utilisateurs occasionnels utilisent très rarement un outil pour la modélisation 3D - mais ils sont souvent amenés à prendre des photos. Une autre activité commune que les utilisateurs font de l'outil informatique est l'édition photographique. Souvent, les utilisateurs veulent apporter des modifications à une photographie existante. Les outils actuellement disponibles pour l'édition de photos permettent une multitude d'opérations. Toutefois, changer l'éclairage dans une photographie est une tâche difficile. Les utilisateurs peuvent reproduire cet effet en modifiant la lu-

minosité de la photographie. Mais des effets subtils tels que l'éclairage indirect et les ombres autoportées sont difficiles à modifier ou supprimer. Dans les scènes en plein air par exemple, des arguments convaincants seraient de permettre aux utilisateurs de modifier l'heure de la journée, ou encore le jour de l'année à laquelle la photo a été prise. Si un tel outil existait, une photographie prise un matin d'été pourrait être transformée automatiquement en une photographie prise lors d'une soirée au début de l'hiver. Dans ce contexte, une autre opération d'édition commune telle que le montage photo, bénéficieraient de l'insertion de nouveaux éléments dans cette scène avec leur éclairage assorti. D'autres opérations d'édition de photo comme le redimensionnement se sont considérablement améliorés au fil des ans [Avidan and Shamir, 2007]. Des méthodes centrées sur le contenu aident les utilisateurs à appliquer des opérations intelligentes de redimensionnement sur les photographies, préservant les zones d'intérêt et supprimant des zones qui ne manqueront pas dans tous les cas.

A.2.3 Interfaces Homme-Machine

Au cours du développement de cette thèse, nous avons réalisé que, malgré le renforcement de fonctionnalités techniques, des outils bénéficieraient d'une amélioration de leur paradigme d'interface. Les outils 3D sont principalement utilisés par des métaphores d'interface 2D tels que la souris et le clavier. Les utilisateurs expérimentés et les experts, sur le long terme, bénéficient de plus en plus de ce paradigme à mesure qu'ils approfondissent leur maîtrise des outils. Les nouveaux arrivants dans le monde de la 3D ont cependant besoin d'apprendre l'interface 2D afin d'interagir et éventuellement de modifier l'environnement en 3D. Pour les changements à petite échelle et le prototypage rapide, une interface 3D est mieux adaptée. Nous nous inspirons d'une vidéo concept créé par Bruce Branit appelée World Builder [1] - voir les Figures A.1 (a) et (b).

(a) (b)

Figure A.1: *World Builder par Bruce Branit:* World Builder par Bruce Branit: (a) présente un paradigme d'interface pour redimensionner un objet en 3D, qui est simplement de pincer deux coins d'une boite et de les écarter, pour obtenir (b)

[1] http://www.branitvfx.com/worldbuilder/

Dans ce contexte, les dispositifs qui permettent d'accomplir ces tâches jouent un rôle important. Des dispositifs appropriés permettraient aux utilisateurs de mieux communiquer avec l'ordinateur, tout en s'exprimant avec précision. A long terme, comme illustré par le concept World Builder et remarqué par [Bowman et al., 2008], les interfaces gestuelles permettent aux utilisateurs de communiquer en toute transparence avec l'ordinateur, leur offrant une véritable immersion.

A.3 Contexte et travaux antérieurs

Plusieurs méthodes ont été proposées pour relever les défis décrits ci-dessus. La modélisation à partir de photographies [Shlyakhter et al., 2001], ou à partir d'exemples [Funkhouser et al., 2004] a cherché à faciliter la création de contenu.

L'édition de modèles est également un défi intéressant, où à la fois les informations géométriques et les informations sémantiques sous-jacentes doivent être prises en compte afin que les modifications soient raisonnables. L'agencement de meubles de bureau a été traité avec un algorithme de placement automatique proposé par [Xu et al., 2002]. Bien qu'il ne soit pas interactif, il permet aux utilisateurs de placer des objets dans la scène de manière à mettre en place des contraintes. [Gleicher, 1994] a proposé une méthode pour intégrer les contraintes et la manipulation directe pour l'édition interactive. Si de l'information sémantique est disponible, une modélisation automatique plus complexe et des opérations d'édition supplémentaires peuvent être réalisées [Lipp et al., 2008]. Bien que ce ne soit pas applicable pour la plupart des maillages d'architecture, les maillages fortement *tesselated* peuvent utiliser d'autres approches d'édition, telles que les techniques basées sur le Laplacien [Sorkine et al., 2004] ou Poisson [Yu et al., 2004]. En raison de leur complexité, le maillage est édité localement.

Cependant, ces techniques cherchent à éditer des maillages 3D en utilisant une métaphore du bureau 2D comme interface. D'autres approches tentent de traiter le problème sous un angle différent, en s'intéressant non seulement à la technique utilisée, mais également la façon dont les utilisateurs peuvent agir efficacement sur l'édition de ces maillages. Teddy [Igarashi et al., 2007] est une métaphore d'interface pour créer des formes 3D à partir d'esquisses 2D. Comme indiqué dans ce travail, au bout de quelques minutes d'apprentissage de l'interface, les utilisateurs sont en mesure de créer des modèles 3D intéressants. D'autres techniques tirent profit des grands écrans et de leur caractère immersif pour créer de nouvelles métaphores d'interface qui permettent de contrôler le niveau de précision lors de l'interaction [Peck et al., 2009], ou pour atteindre des objets éloignés [Pierce et al., 1999, Bowman and Hodges, 1997]. Les récents travaux de [Mine et al., 1997a] explorent le concept de proprioception pour la création d'une métaphore d'interaction 3D, en positionnant les objets à proximité des mains et du corps des utilisateurs.

En dépit de récents travaux, [Bowman and Fröhlich, 2005] affirment qu'il n'y a pas eu d'augmentation de la qualité et du nombre de solutions utilisables pour des applications de Réalité Virtuelle dans des environnements immersifs, principalement en raison

du manque de véritable recherche en interfaces utilisateur 3D. Nous avons donc mis l'accent sur l'extension de notre approche 2D vers une métaphore 3D qui propose une nouvelle interface pour la conception de base et l'éclairage d'architectures, en mêlant simultanément plusieurs échelles d'interaction (voir le Chapitre 4).

Modifier l'éclairage de photographies est un objectif de longue date en infographie. Plusieurs méthodes ont été proposées, mais elles nécessitent une procédure de capture difficile pour effectuer un ré-éclairage automatique [Yu et al., 1999] ou interactif [Loscos et al., 2000]. L'insertion de nouveaux objets dans une photographie avec un éclairage cohérent a également été résolu à l'aide de procédures de capture complexes [Debevec, 1998] qui exigent un certain effort, et qui ne sont pas destiné à des utilisateurs occasionnels. Les techniques récentes abordent ce problème en tirant parti de la grande quantité de données disponibles en ligne pour permettre le ré-éclairage par transfert d'information sur les sources lumineuses entre les photographies [Lalonde et al., 2009]. Ces approches guidées par les données fonctionnent bien pour la plupart des cas, mais nécessitent une quantité considérable de pré-traitement.

Nous pensons que, malgré les travaux antérieurs dans ces domaines, il est nécessaire de poursuivre le développement d'outils permettant de faciliter la création de contenu transparente pour l'utilisateur.

A.4 Objectifs

Notre but est d'augmenter les possibilités de l'utilisateur moyen, en lui permettant d'effectuer des tâches qui seraient autrement trop difficile à effectuer manuellement. Dans ce contexte, notre objectif est de créer des outils qui aident les utilisateurs à créer de nouveaux contenus à partir des données existantes. Nous ciblons deux domaines différents: (1) la création de nouveaux modèles 3D texturés à partir d'existants, et (2) la modification de l'éclairage sur des photographies existantes.

Plus spécifiquement pour (1), nous nous concentrons sur les modèles 3D architecturaux car ils sont largement disponibles en ligne, et sont couramment utilisés dans des applications comme les jeux et le design. Pour (2) nous avons choisi de modifier l'éclairage de la cime des arbres pour des environnements extérieurs, car ils sont fréquemment rencontrés dans la plupart des scènes en extérieur.

A.4.1 Modification de géométrie interactive

Pour résoudre le problème de la modélisation interactive d'environnements architecturaux, nous nous inspirons des approches existantes pour l'édition de maillage. Notre principale contribution, comme nous le verrons en détail au Chapitre 3, est le développement d'une technique interactive qui extrait automatiquement un ensemble d'équations linéaires à partir de l'entrée qui exprime leur forme générale. Au moment de l'exécution, l'interaction avec l'utilisateur déclenche la résolution du système au sens des moindres carrés. L'utilisateur a la possibilité de déplacer les sommets du maillage, tandis que le

système sous-jacent prend en charge les détails en préservant les caractéristiques originales du maillage.

Les textures sont un moyen essentiel de représenter les détails dans les maillages architecturaux. Elles contiennent des informations géométriques qui ajoutent de la profondeur et de la décoration dans une variété d'environnements. De la même manière que la géométrie est remodelée, nous avons adapté notre méthode à un opérateur similaire pour modifier les textures. Notre approche identifie les régions de la texture qui peuvent être étirées, afin de concentrer les déformations dans ces zones. Les détails sont ensuite réintroduits, ce qui permet un résultat global de haute qualité.

Une modification de la géométrie déclenche la modification de la texture sous-jacente, lui permettant de s'adapter à la nouvelle taille du maillage. Notre système est interactif et toutes ces opérations sont effectuées au moment de l'exécution. En raison de sa nature interactive, les utilisateurs peuvent tester plusieurs possibilités d'édition du maillage, et revenir en arrière, avant de décider des réglages finaux.

Les Figures A.2 (a) et (b) montrent les résultats de notre outil d'édition de maillage interactif. La Figure A.2 (a) montre un ensemble initial de maillages 3D provenant de plusieurs niveaux de jeux vidéos. Dans la Figure A.2 (b), ces pièces ont été remodelées et connectées afin de créer un nouveau niveau. Notez comme les textures ont été modifiées de manière significative entre le modèle d'entrée et la géométrie finale remodelée.

(a)

(b)

Figure A.2: (a) maillages 3D originaux provenant de niveaux de jeux vidéo; (b) notre approche pour l'édition interactive de maillage permet de remodeler individuellement et connecter les pièces, afin de créer de nouveaux environnements avec des textures adaptées.

A.4.2 Modification en 3D

En testant notre outil pour l'édition de maillage, nous avons réalisé que l'une des difficultés a été la configuration du clavier et de la souris 2D pour traduire des commandes 3D. L'ambiguïté inhérente à cette métaphore ne convient pas pour l'interaction dans une application 3D comme l'édition d'architecture. Visualiser et interpréter la véritable profondeur des modèles réduit l'ambiguïté lors de l'interaction. En outre, il est sans doute préférable d'utiliser des commandes 3D dans l'espace pour ce type de modifications géométriques.

Dans cet esprit, nous avons encore étendu notre approche et avons conçu une interface 3D afin de réaliser l'édition de géométrie dans un environnement immersif de Réalité Virtuelle, et en particulier le iSpace 4-faces de BARCO à l'INRIA Sophia-Antipolis. Dans cet environnement, l'utilisateur est entouré par des écrans de projection représentant le monde virtuel. Grâce à un tracker optique, les mouvements de la main et de la tête de l'utilisateur peuvent être suivis en temps réel.

Nous utilisons cette technologie et permettons d'effectuer les opérations d'édition de maillage d'architecture en 3D. Les utilisateurs peuvent saisir et déplacer les sommets du maillage, au moyen d'un *flystick*, comme s'ils étaient réellement debout devant le modèle 3D. Dans ce système, l'utilisateur dispose de trois options différentes pour l'interaction: une version à petite échelle du modèle est présentée devant lui; à l'échelle 1:1 du modèle; et, un mode mixte, où le modèle est présenté à la fois à l'échelle 1:1 et à petite échelle. Nous évaluons ensuite les actions des utilisateurs et leur performance à travers ces trois paradigmes différents.

De plus, nous permettons aux utilisateurs d'explorer la conception d'éclairage. Nous utilisons une version simplifiée d'un algorithme d'illumination globale pour évaluer l'éclairage indirect du monde architectural en cours d'édition. Les utilisateurs peuvent changer le jour de l'année et le moment de la journée afin de simuler comment la lumière du soleil illuminera ces modèles.

La Figure A.3 montre un utilisateur en train de changer la position d'une fenêtre dans un environnement en mode mixte, où le modèle est présent à la fois en version à l'échelle 1:1 et à petite échelle.

A.4.3 Ré-éclairage de photographies d'arbres

Pour atteindre cet objectif, nous combinons des algorithmes avec de nouvelles techniques. Tout d'abord, dans le chapitre 6 nous développons une nouvelle approche qui s'appuie sur la modélisation volumétrique des arbres pour permettre le ré-éclairage de photographies de la cime des arbres. En utilisant un petit ensemble de photographies prises à partir d'un arbre à un **unique** moment de la journée, nous sommes en mesure d'effectuer le ré-éclairage, à savoir, changer l'heure de la journée et / ou le jour de l'année à laquelle la photo a été prise. Nous construisons un proxy volumétrique approximatif du feuillage des arbres et estimons les paramètres d'éclairage au moment où la photo a été prise en entrée. En utilisant une seule méthode de diffusion de rendu volumétrique avec des ratios d'images, nous sommes en mesure d'obtenir un effet convaincant pour

Figure A.3: L'utilisateur modifie la position d'une fenêtre en mode mixte (détails au Chapitre 4).

le ré-éclairage du feuillage qui est qualitativement comparable à une photographie de l'éclairage réel d'un arbre prise au moment cible. Pour un exemple de notre technique de ré-éclairage, voir les figures A.4 (a), (b) et (c). La figure 1.4 (a) montre la photo d'entrée, prise à midi. La figure A.4 (b) montre notre résultat de ré-éclairage pour changer l'heure de la journée à 18h00. La figure A.4 (c) montre une photographie réelle de l'arbre prise au moment cible (18h00).

(a) (b) (c)

Figure A.4: (a) photographie d'entrée prise à midi; (b) notre résultat de ré-éclairage pour changer l'heure de la journée à 18h00; (c) photographie réelle prise à 18h00 pour comparaison.

A.5 Contributions

Notre objectif est de contribuer à l'élaboration d'outils qui facilitent la création de contenu. En tant que tel, nous considérons que les apports de cette thèse sont:

- Outil interactif d'édition du maillage destiné aux modèles d'architecture: Nous développons un algorithme qui permet l'édition interactive de maillages 3D architecturaux, en tenant compte à la fois de la géométrie et des textures. Notre formulation extrait un ensemble d'équations linéaires pour la géométrie et les textures, permettant aux utilisateurs de modifier le maillage en temps réel tandis que le système sous-jacent assure la cohérence des maillages d'entrée en résolvant le système d'équations selon la méthode des moindres carrés. À notre connaissance, c'est la première fois qu'un algorithme propose de modifier la géométrie et les textures simultanément et de manière interactive.

- Interface 3D pour la conception interactive simple de modèles architecturaux et d'éclairage: Nous avons développé une interface utilisateur immersive multi-mode pour permettre l'édition interactive de maillages texturés. Nous permettons aux utilisateurs de faire de la conception architecturale simple à l'aide de gestes 3D au moyen d'un joystick dans un environnement de projection immersif. À notre connaissance, c'est la première approche qui permet une édition architecturale simple avec une conception de l'éclairage à exécuter dans un environnement immersif de réalité virtuelle, et qui est également accompagnée d'une étude comparative pour les différentes configurations de l'interface.

- Ré-éclairage de photographies de feuillages d'arbres utilisant en entrée des photographies prises à un seul moment de la journée: Nous développons un algorithme qui prend en entrée un petit ensemble de photographies d'un arbre à un **unique** moment de la journée et effectue le ré-éclairage de l'entrée en changeant l'heure et le jour de l'année de la photo, en changeant en réalité l'éclairage du feuillage de l'arbre. Notre méthode utilise une approche volumétrique pour effectuer le ré-éclairage. À notre connaissance, c'est la première fois qu'un algorithme est capable de ré-éclairer la cime des arbres en utilisant seulement une condition d'éclairage unique en entrée.

A.6 Organisation

Cette thèse est structurée en deux parties principales.

La partie I commence par le chapitre 2 qui donne un bref aperçu des travaux relatifs à notre outil d'édition de maillages et son extension à un environnement en 3D. Nous introduisons ensuite notre outil de remodelisation dans le chapitre 3 et nous montrons ses résultats. Dans le chapitre 4, nous expliquons comment nous étendons notre outil de remodelisation à un environnement 3D et comment nous concevons un ensemble de tests pour évaluer l'efficacité et la performance de notre solution.

La deuxième partie présente nos travaux sur le ré-éclairage de photographies. Dans le chapitre 5, nous passons en revue la littérature liée à la modélisation des arbres, au ré-éclairage à partir d'images et aux algorithmes volumétriques. Puis dans le chapitre 6, nous introduisons notre algorithme de ré-éclairage de la cime des arbres.

Le chapitre 7 présente nos conclusions et discussions finales, ce qui donne un aperçu des futurs travaux. Enfin, il y a une annexe en ligne, située à l'adresse *http://www-*

sop.inria.fr/members/Marcio.Cabral/thesis/ qui contient des résultats supplémentaires pour notre algorithme de ré-éclairage des arbres, contenant l'ensemble complet des sequences de photographies prises pour la comparaison avec la réalité ainsi que d'autres rendus réels pour nos arbres de synthèse. L'annexe contient aussi des vidéos montrant les résultats des chapitres 3, 4 et 6.

A.7 Conclusion

Cette thèse apporte de nouvelles contributions à la création de nouveaux contenus numériques à partir de données existantes. Nos contributions couvrent trois domaines qui sont importants dans la réalisation de cet objectif: l'édition interactive de géométrie, le ré-éclairage à partir d'images et les interactions homme-machine. Dans chaque cas, nous avons fourni des solutions qui contribueront à l'élaboration de nouveaux outils qui, nous l'espérons, faciliteront la création de tels contenus à l'avenir.

A.7.1 Edition interactive de Géométrie

Notre solution d'édition de maillages fournit une nouvelle approche pour la création interactive de nouveaux contenus. Les objets structurés créés par l'homme et les modèles 3D architecturaux peuvent être remodelés en deplaçant simplement les sommets. Notre approche gère également la texture sous-jacente, la transformant également de manière appropriée. Chacun de ces modèles retouchés peut alors être connecté, permettant la création de nouveaux modèles. Cette approche est rapide et elle permet la construction rapide de modèles complexes. Ces tâches, si elles sont effectuées manuellement à l'aide des outils courants d'édition 3D, exigeraient beaucoup d'efforts, même pour des utilisateurs expérimentés. D'autre part, notre algorithme permet aux utilisateurs inexpérimentés de réaliser de telles opérations. Notre approche extrait automatiquement une série d'équations linéaires qui caractérisent la forme globale du modèle 3D. Au cours de l'exécution, lorsque l'utilisateur déplace les sommets pour obtenir le remodelisation souhaité, un solveur des moindres carrés permet de résoudre ces équations, préservant l'aspect original du modèle 3D tout en répondant aux mouvements de l'utilisateur. Le maillage remodelé déclenche la déformation de la texture des côtés, d'une manière similaire, en utilisant également un solveur des moindres carrés.

Nous démontrons notre approche à travers plusieurs scénarios, en utilisant des modèles en intérieur, mais également des modèles créés spécialement pour les jeux, qui sont largement disponibles en ligne. Il s'agit d'un avantage majeur, qui permet aux utilisateurs d'étendre leur capacité de créer de nouveaux contenus a partir de modèles précédemment créés. Une autre caractéristique essentielle est le remodelisation des textures et de la géométrie en même temps, ce qui n'avait pas été proposé avant ce travail. Notre mesure d'autosimilarité directionnelle est capable d'identifier les zones qui peuvent être étirées, telles que les régions stochastiques, et les zones avec des détails structurés, qui doivent être rigides à la déformation. D'autres exemples auxquels nous avons appliqué notre algorithme comprennent la construction d'une route et d'un pont à partir

de quelques pièces de base.

A.7.1.1 Travaux Futurs

Nous aimerions étendre notre approche à des géométries générales par opposition aux modèles architecturaux et aux objets structurés créés par l'homme. Des travaux récents qui portent sur le remodelisation de géométries générales telles que iWires [Gal et al., 2009] proposent une direction intéressante.

Actuellement, l'édition couplée de la géométrie et de la texture est unidirectionnelle: le remodelisation de la géométrie déclenche la déformation de la texture. Il est facile d'explorer ce phénomène dans l'autre sens, en introduisant des contraintes géométriques basées sur la rigidité sous-jacente de la texture.

A.7.2 Interaction Homme-Machine

Les premiers résultats de l'édition interactive de géométrie nous ont montré que, bien que notre outil ait obtenu de bons résultats, les utilisateurs se débattaient avec son interface 2D. Par conséquent, nous avons approfondi la manière d'aborder le problème de la transposition de notre système dans un cadre immersif. Nous y sommes parvenu en créant une nouvelle interface 3D et en permettant aux utilisateurs d'interagir avec le système dans un environnement de réalité virtuelle totalement immersif en utilisant des gestes de commande simples.

Le système immersif développé a permis le design conceptuel simple de modèles architecturaux avec la possibilité d'explorer différentes configurations d'éclairage en changeant l'heure et la date de l'année. Un algorithme simplifié d'éclairage global simulait les interactions lumineuses dans le design conceptuel d'une maison permettant aux utilisateurs de simuler la façon dont la lumière du soleil contribuerait à l'illumination globale de la scène. Une étude utilisateur informelle a montré que les utilisateurs novices d'un système immersif ont trouvé l'expérience et l'interactivité enrichissantes et intéressantes.

Notre système est complètement immersif et permet un design conceptuel simple de modèles géométriques texturés. Il est le premier à également comprendre la conception d'un éclairage de base. Trois modes différents permettent aux utilisateurs d'interagir à différentes échelles. L'un d'eux, appelé mode mixte, est une variante d'un mode d'interaction basé sur le paradigme des WIM [Stoakley et al., 1995] et il fournit à la fois des vues locale et globale du monde 3D manipulé. Les utilisateurs qui ont expérimenté le système ont utilisé équitablement les trois modes pour les tâches plus libres. Globalement, les utilisateurs ont apprécié l'utilisation du système. De plus, les utilisateurs n'ayant aucune expérience des systèmes immersifs de réalité virtuelle ont été en mesure d'utiliser notre système après une courte session d'entraînement de 10 minutes.

A.7.2.1 Travaux Futurs

Le but final d'une telle édition immersive de géométrie 3D et de textures est d'éliminer complètement le besoin de dispositifs d'entrée 3D, tels que la manette utilisée. Nous espérons plus tard utiliser de nouveaux dispositifs de capture, tels que les systèmes de suivi des doigts et des mains. Nous avons commencé à expérimenter ces appareils à un stade final du développement de cette thèse, mais nous les avons trouvés fragiles, ce qui les rend inconfortable pour un usage régulier.

D'autres pistes incluent l'extension de l'algorithme d'illumination globale à des modèles plus complexes, analysant comment différents niveaux de réalisme affectent la perception du système par l'utilisateur.

A.7.3 Ré-éclairage à partir d'images

Nous avons proposé une méthode pour ré-éclairer le feuillage des arbres. En utilisant uniquement des photos prises à un seul moment de la journée, notre algorithme peut ré-éclairer la cime des arbres pour correspondre à n'importe quel autre moment, ou jour. Notre méthode traite le problème du ré-éclairage en interprétant le feuillage des arbres comme une entité volumétrique. En tant que tel, une technique de rendu de volume par diffusion simple a été utilisée pour estimer les paramètres d'éclairage du feuillage des arbres. Avec ces paramètres estimés pour chaque pixel pour l'image d'entrée, notre méthode estime les mêmes paramètres pour les conditions d'éclairage souhaitées et effectue le ré-éclairage en utilisant une simple méthode de ratios d'images. Notre algorithme a bien fonctionné pour plusieurs variétés d'arbres. Nous avons validé notre méthode en utilisant des rendus de synthèse d'arbres ainsi que les temps réels, avec des séquences de photographies prises tout au long de la journée.

A.7.3.1 Travaux Futurs

Une part importante des futurs travaux est le développement de solutions pour le ré-éclairage d'environnements entiers et non pas seulement de feuillages d'arbres. Dans un premier temps, les ombres doivent être éliminées de l'environnement de façon appropriée. Le problème le plus difficile est de résoudre le ré-éclairage de l'environnement entier qui comprend différents types d'objets avec des caractéristiques différentes. En tant que tel, une solution hybride peut être proposée où plusieurs approches sont utilisées pour ré-éclairer les différentes zones de la scène.

A.8 Conclusion

Nous avons vu récemment une percée en termes de quantité et de qualité du contenu créé par les utilisateurs de partout, y compris les utilisateurs novices de systèmes informatiques. En particulier, les solutions proposées dans cette thèse fournissent aux utilisateurs des moyens intuitifs pour créer une géométrie texturée à partir de pièces existantes, y compris dans un cadre d'immersion, et de manipuler l'éclairage dans les

photographies d'arbres. Nous espérons que cette thèse est un pas en avant pour fournir aux utilisateurs de meilleurs outils pour créer des contenus qui autrement seraient difficiles à créer manuellement. Au final, notre objectif est de donner aux utilisateurs les capacités leur permettant de facilement créer et manipuler du contenu 3D, sans avoir besoin de talent et de formation de longue durée. Lorsque cet objectif sera atteint, leur imagination créatrice ne sera plus limitée par la technologie.

Personal Publications

[Cabral et al., 2010] Cabral, M., Bonneel, N., Lefebvre, S., and Drettakis, G. (2010). Relighting photographs of tree canopies. *IEEE Transactions on Visualization and Computer Graphics*, 99(PrePrints).

[Cabral et al., 2009] Cabral, M., Lefebvre, S., Dachsbacher, C., and Drettakis, G. (2009). Structure preserving reshape for textured architectural scenes. *Computer Graphics Forum (Proc. of the Eurographics conference)*.

[Cabral et al., 2011] Cabral, M., Vangorp, P., Chaurasia, G., Chapoulie, E., Hachet, M., and Drettakis, G. (2011). A multimode immersive conceptual design system for architectural modeling and lighting. In *Proceedings of IEEE 3DUI (technote)*. IEEE.

Bibliography

[Anderson et al., 2003] Anderson, L., Esser, J., and Interrante, V. (2003). A virtual environment for conceptual design in architecture. In *Proc. of the workshop on Virtual environments 2003*, EGVE '03, pages 57–63.

[Autodesk Maya, 2010] Autodesk Maya (2010). Autodesk maya. http://www.autodesk.com/maya. Last accessed: 18 Nov. 2010.

[Avidan and Shamir, 2007] Avidan, S. and Shamir, A. (2007). Seam carving for content-aware image resizing. *ACM Trans. on Graphics (Proc. of SIGGRAPH 2007)*, 26(3).

[Balakrishnan and Kurtenbach, 1999] Balakrishnan, R. and Kurtenbach, G. (1999). Exploring bimanual camera control and object manipulation in 3d graphics interfaces. In *CHI '99: Proc. of the SIGCHI conference on Human factors in computing systems*, pages 56–62. ACM.

[Ball et al., 2007] Ball, R., North, C., and Bowman, D. A. (2007). Move to improve: promoting physical navigation to increase user performance with large displays. In *CHI '07: Proc. of the SIGCHI conference on Human factors in computing systems*, pages 191–200. ACM.

[Barnes et al., 2009] Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. B. (2009). Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28:24:1–24:11.

[Berndt et al., 2005] Berndt, R., Fellner, D. W., and Havemann, S. (2005). Generative 3d models: a key to more information within less bandwidth at higher quality. In *Web3D '05: Proceedings of the tenth international conference on 3D Web technology*, pages 111–121.

[Besl and McKay, 1992] Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14:239–256.

[Blender, 2010] Blender (2010). Blender. http://www.blender.org/. Last accessed: 18 Nov. 2010.

[Boivin and Gagalowicz, 2001] Boivin, S. and Gagalowicz, A. (2001). Image-based rendering of diffuse, specular and glossy surfaces from a single image. In *SIGGRAPH '01*, pages 107–116.

[Borning et al., 1997] Borning, A., Marriott, K., Stuckey, P., and Xiao, Y. (1997). Solving linear arithmetic constraints for user interface applications. In *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 87–96, New York, NY, USA. ACM.

[Botsch et al., 2005] Botsch, M., Bommes, D., and Kobbelt, L. (2005). Efficient linear system solvers for mesh processing. In *IMA Conference on the Mathematics of Surfaces*, pages 62–83.

[Botsch and Sorkine, 2008] Botsch, M. and Sorkine, O. (2008). On linear variational surface deformation methods. *IEEE Trans. on Vis. and Comp. Graphics*, 14(1):213–230.

[Boulanger et al., 2008] Boulanger, K., Bouatouch, K., and Pattanaik, S. N. (2008). Rendering trees with indirect lighting in real time. *Comput. Graph. Forum*, 27(4):1189–1198.

[Bowman et al., 2008] Bowman, D. A., Coquillart, S., Froehlich, B., Hirose, M., Kitamura, Y., Kiyokawa, K., and Stuerzlinger, W. (2008). 3d user interfaces: New directions and perspectives. *IEEE Comput. Graph. Appl.*, 28(6):20–36.

[Bowman and Fröhlich, 2005] Bowman, D. A. and Fröhlich, B. (2005). New directions in 3d user interfaces. In *VR '05: Proc. of the 2005 IEEE Conference 2005 on Virtual Reality*, page 312, Washington, DC, USA. IEEE Computer Society.

[Bowman et al., 2002a] Bowman, D. A., Gabbard, J. L., and Hix, D. (2002a). A survey of usability evaluation in virtual environments: Classification and comparison of methods. *Presence: Teleoperators and Virtual Environments*, 11(4):404–424.

[Bowman and Hodges, 1997] Bowman, D. A. and Hodges, L. F. (1997). An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *I3D '97: Proc. of the 1997 Symp. on I3D*, pages 35–ff. ACM.

[Bowman et al., 2001] Bowman, D. A., Wingrave, C. A., Campbell, J. M., and Ly, V. Q. (2001). Using pinch gloves for both natural and abstract interaction techniques in virtual environments. In *Proc. HCI International 2001*, pages 629–633.

[Bowman et al., 2002b] Bowman, D. A., Wingrave, C. A., Campbell, J. M., Ly, V. Q., and Rhoton, C. J. (2002b). Novel uses of pinch gloves for virtual environment interaction techniques. *Virtual Reality*, 6:122–129.

[Boykov and Jolly, 2001] Boykov, Y. Y. and Jolly, M. P. (2001). Interactive graph cuts for optimal boundary region segmentation of objects in n-d images. In *ICCV*, volume 1, pages 105–112 vol.1.

[Branit, 2009] Branit, B. (2009). World Builder. `http://www.branitvfx.com/worldbuilder/`. Short film. Last accessed: 18 Nov. 2010.

[Buchmann et al., 2004] Buchmann, V., Violich, S., Billinghurst, M., and Cockburn, A. (2004). Fingartips: gesture based direct manipulation in augmented reality. In *GRAPHITE '04: Proc. of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 212–221. ACM.

[Bullinger et al., 2010] Bullinger, H.-J., Bauer, W., Wenzel, G., and Blach, R. (2010). Towards user centred design (ucd) in architecture based on immersive virtual environments. *Computers in Industry*, 61(4):372 – 379.

[Cerezo et al., 2005] Cerezo, E., Perez-Cazorla, F., Pueyo, X., Seron, F., and Sillion, F. (2005). A survey on participating media rendering techniques. *the Visual Computer*.

[Chen and Bowman, 2009] Chen, J. and Bowman, D. A. (2009). Domain-specific design of 3d interaction techniques: An approach for designing useful virtual environment applications.

[Chen et al., 2010] Chen, R., Freedman, D., Karni, Z., Gotsman, C., and Liu, L. (2010). Content-aware image resizing by quadratic programming. *Proc. CVPR Workshop on Non-Rigid Shape Analysis and Deformable Image Alignment (NORDIA)*, pages 1–8.

[Cutler et al., 1997] Cutler, L. D., Fröhlich, B., and Hanrahan, P. (1997). Two-handed direct manipulation on the responsive workbench. In *I3D '97: Proc. of the 1997 Symp. on I3D*, pages 107–114. ACM.

[Daniels et al., 1997] Daniels, K., Milenkovic, V., and Roth, D. (1997). Finding the largest area axis-parallel rectangle in a polygon. *Computational Geometry: Theory and Applications*, 7(1-2):125–148.

[de Castro and Fetcher, 1998] de Castro, F. and Fetcher, N. (1998). Three dimensional model of the interception of light by a canopy. *Agricultural and Forest Meteorology*, 90(3):215 – 233.

[Debevec, 1998] Debevec, P. (1998). Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH '98*, pages 189–198.

[Debevec, 2002] Debevec, P. (2002). Image-based lighting. *IEEE Computer Graphics and Applications*, 22(2):26–34.

[Debevec et al., 2000] Debevec, P., Hawkins, T., Tchou, C., Duiker, H.-P., Sarokin, W., and Sagar, M. (2000). Acquiring the reflectance field of a human face. In *SIGGRAPH '00*, pages 145–156.

[Deisinger et al., 2000] Deisinger, J., Blach, R., Wesche, G., Breining, R., and Simon, A. (2000). Towards immersive modeling - challenges and recommendations: A workshop analyzing the needs of designers. In *In Proc. of the 6th Eurographics Workshop on Virtual Environments*, pages 145–156.

[Drori et al., 2003] Drori, I., Cohen-Or, D., and Yeshurun, H. (2003). Fragment-based image completion. *ACM Trans. on Graphics (Proc. of SIGGRAPH 2003)*, 22(3):303–312.

[EA, 2008] EA (2008). Spore. http://www.spore.com.

[Efros and Leung, 1999] Efros, A. A. and Leung, T. K. (1999). Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision*, pages 1033–1038.

[Frees and Kessler, 2005] Frees, S. and Kessler, G. D. (2005). Precise and rapid interaction through scaled manipulation in immersive virtual environments. In *VR '05: Proc. of the 2005 IEEE Conference 2005 on Virtual Reality*, pages 99–106, Washington, DC, USA. IEEE Computer Society.

[Funkhouser et al., 2004] Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., and Dobkin, D. (2004). Modeling by example. *ACM Trans. on Graphics (Proc. of SIGGRAPH 2004)*.

[Gal et al., 2009] Gal, R., Sorkine, O., Mitra, N. J., and Cohen-Or, D. (2009). iwires: an analyze-and-edit approach to shape manipulation. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–10. ACM.

[Gelfand and Guibas, 2004] Gelfand, N. and Guibas, L. (2004). Shape segmentation using local slippage analysis. *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*.

[Gleicher, 1992] Gleicher, M. L. (1992). Integrating constraints and direct manipulation. In *Proc. of the ACM SIGGRAPH Symp. on Interactive 3D Graphics*, pages 171–174, New York, NY, USA. ACM.

[Gleicher, 1994] Gleicher, M. L. (1994). *A Differential Approach to Graphical Interaction*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA. Advisor – Andy Witkin.

[Google Sketchup, 2010] Google Sketchup (2010). Google sketchup. `http://sketchup.google.com/`. Last accessed: 18 Nov. 2010.

[Greenberg et al., 1986] Greenberg, D., Cohen, M., and Torrance, K. (1986). Radiosity: A method for computing global illumination. *The Visual Computer*.

[Grossman et al., 2004] Grossman, T., Wigdor, D., and Balakrishnan, R. (2004). Multi-finger gestural interaction with 3d volumetric displays. In *UIST '04: Proc. of the 17th annual ACM Symp. on User interface software and technology*, pages 61–70. ACM.

[Guiard, 1987] Guiard, Y. (1987). Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *Journal of Motor Behavior*, 19:486–517.

[Habel et al., 2008] Habel, R., Mustata, B., and Wimmer, M. (2008). Efficient spherical harmonics lighting with the preetham skylight model. In *Eurographics 2008 - Short Papers*.

[Haumont et al., 2003] Haumont, D., Debeir, O., and Sillion, F. (2003). Volumetric cell-and-portal generation. In *Computer Graphics Forum*, volume 3-22 of *EUROGRAPHICS Conference Proc.*

[Igarashi et al., 2007] Igarashi, T., Matsuoka, S., and Tanaka, H. (2007). Teddy: a sketching interface for 3d freeform design. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, New York, NY, USA. ACM.

[Jansen and Bavoil, 2010] Jansen, J. and Bavoil, L. (2010). Fourier opacity mapping. In *I3D '10: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 165–172, New York, NY, USA. ACM.

[Ju, 2004] Ju, T. (2004). Robust repair of polygonal models. *ACM Trans. on Graphics*, 23(3):888–895.

[Kajiya and Von Herzen, 1984] Kajiya, J. T. and Von Herzen, B. P. (1984). Ray tracing volume densities. *SIGGRAPH Comput. Graph.*, 18(3):165–174.

[Kasten and Young, 1989] Kasten, F. and Young, A. (1989). Revised optical air mass tables and approximation formula. In *Applied Optics*, volume 28, pages 4735–4738.

[Keskin et al., 2003] Keskin, C., Erkan, A., and Akarun, L. (2003). Real time hand tracking and 3d gesture recognition for interactive interfaces using hmm. In *In Proc. of International Conference on Artificial Neural Networks*.

[Kraevoy et al., 2007] Kraevoy, V., Julius, D., and Sheffer, A. (2007). Model composition from interchangeable components. In *Pacific Graphics*.

[Kraevoy et al., 2008] Kraevoy, V., Sheffer, A., Cohen-Or, D., and Shamir, A. (2008). Non-homogeneous resizing of complex models. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH ASIA 2008)*, 27(5).

[Kwatra et al., 2003] Kwatra, V., Schdl, A., Essa, I., Turk, G., and Bobick, A. (2003). Graphcut textures: Image and video synthesis using graph cuts. *Proc. SIGGRAPH*.

[Laffont et al., 2010] Laffont, P.-Y., Jun, J. Y., Wolf, C., Tai, Y.-W., Idrissi, K., Drettakis, G., and Yoon, S.-e. (2010). Interactive content-aware zooming. In Mould, D. and Noel, S., editors, *Proceedings of the Graphics Interface 2010 Conference*, pages 79–87.

[Lalonde et al., 2009] Lalonde, J.-F., Efros, A. A., and Narasimhan, S. G. (2009). Webcam clip art: Appearance and illuminant transfer from time-lapse sequences. *ACM Trans. on Graphics (SIGGRAPH Asia 2009)*, 28(5).

[Laviola et al., 2004] Laviola, I. J., Laviola, J. J., Daniel, J., Keefe, F., Zeleznik, R. C., and Feliz, D. A. (2004). Case studies in building custom input devices for virtual environment. In *In VR 2004 Workshop: Beyond Glove and Wand Based Interaction*, pages 67–71.

[Laviola, 1999] Laviola, J. (1999). Flex and pinch: A case study of whole hand input design for virtual environment interaction.

[Lefebvre and Hornus, 2003] Lefebvre, S. and Hornus, S. (2003). Automatic cell-and-portal decomposition. Technical Report 4898, INRIA.

[Leigh and Johnson, 1996] Leigh, J. and Johnson, A. (1996). Calvin: an immersimedia design environment utilizing heterogeneous perspectives. *Multimedia Computing and Systems, International Conference on*, 0:0020.

[Levin et al., 2008] Levin, A., Lischinski, D., and Weiss, Y. (2008). A closed-form solution to natural image matting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(2):228–242.

[Lipp et al., 2008] Lipp, M., Wonka, P., and Wimmer, M. (2008). Interactive visual editing of grammars for procedural architecture. *ACM Trans. on Graphics (Proc. of SIGGRAPH 2008)*, 27(3).

[Liu et al., 2001] Liu, Z., Shan, Y., and Zhang, Z. (2001). Expressive expression mapping with ratio images. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 271–276, New York, NY, USA. ACM.

[Loan, 1985] Loan, C. V. (1985). On the method of weighting for equality-constrained least-squares problems. *SIAM Journal on Numerical Analysis*, 22(5):851–864.

[Loscos et al., 2000] Loscos, C., Drettakis, G., and Robert, L. (2000). Interactive virtual relighting of real scenes. *IEEE Trans. on Visualization and Computer Graphics*, 6(3).

[Lucas et al., 2005] Lucas, J. F., Kim, J.-S., and Bowman, D. A. (2005). Resizing beyond widgets: object resizing techniques for immersive virtual environments. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1601–1604. ACM.

[Malik et al., 2005] Malik, S., Ranjan, A., and Balakrishnan, R. (2005). Interacting with large displays from a distance with vision-tracked multi-finger gestural input. In *UIST '05: Proc. of the 18th annual ACM Symp. on User interface software and technology*, pages 43–52. ACM.

[Marschner and Greenberg, 1997] Marschner, S. R. and Greenberg, D. P. (1997). Inverse lighting for photography. In *Proceedings of the Fifth Color Imaging Conference, Society for Imaging Science and Technology*, pages 262–265.

[Masselus et al., 2003] Masselus, V., Peers, P., Dutré, P., and Willems, Y. D. (2003). Relighting with 4d incident light fields. *ACM Trans. Graph.*, 22(3):613–620.

[McCrae et al., 2009] McCrae, J., Mordatch, I., Glueck, M., and Khan, A. (2009). Multiscale 3d navigation. In *I3D '09: Proc. of the 2009 Symp. on I3D and games*, pages 7–14. ACM.

[Merrell, 2007] Merrell, P. (2007). Example-based model synthesis. In *Proc. of the ACM SIGGRAPH Symp. on Interactive 3D Graphics and Games*.

[Merrell and Manocha, 2008] Merrell, P. and Manocha, D. (2008). Continuous model synthesis. In *SIG - Asia*, number 27, page to appear, New York, NY, USA. ACM SIGGRAPH.

[Merrell et al., 2010] Merrell, P., Schkufza, E., and Koltun, V. (2010). Computer-generated residential building layouts. *SIGGRAPH ASIA '10: SIGGRAPH Asia 2010 papers*.

[Mine et al., 1997a] Mine, M. R., Brooks, Jr., F. P., and Sequin, C. H. (1997a). Moving objects in space: exploiting proprioception in virtual-environment interaction. In *SIGGRAPH '97: Proc. of the 24th annual conference on Computer graphics and interactive techniques*, pages 19–26. ACM Press/Addison-Wesley Publishing Co.

[Mine et al., 1997b] Mine, M. R., Brooks, Jr., F. P., and Sequin, C. H. (1997b). Moving objects in space: exploiting proprioception in virtual-environment interaction. In *SIGGRAPH '97: Proc. of the 24th annual conference on Computer graphics and interactive techniques*, pages 19–26. ACM Press/Addison-Wesley Publishing Co.

[Moscovich and Hughes, 2006] Moscovich, T. and Hughes, J. F. (2006). Multi-finger cursor techniques. In *GI '06: Proc. of Graphics Interface 2006*, pages 1–7, Toronto, Ont., Canada, Canada. Canadian Information Processing Society.

[Müller et al., 2006] Müller, P., Wonka, P., Haegler, S., Ulmer, A., and Gool, L. V. (2006). Procedural modeling of buildings. In *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH 2006)*, pages 614–623.

[Müller et al., 2007] Müller, P., Zeng, G., Wonka, P., and Gool, L. V. (2007). Image-based procedural modeling of facades. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH 2007)*, 26(3):85.

[Neubert et al., 2007] Neubert, B., Franken, T., and Deussen, O. (2007). Approximate image-based tree-modeling using particle flows. *ACM Trans. on Graphics (SIGGRAPH 2007)*, 26(3):88.

[Nicodemus, 1965] Nicodemus, F. (1965). Directional reflectance and emissivity of an opaque surface. *Applied Optics*.

[Owen et al., 2005] Owen, R., Kurtenbach, G., Fitzmaurice, G., Baudel, T., and Buxton, B. (2005). When it gets more difficult, use both hands: exploring bimanual curve manipulation. In *GI '05: Proc. of Graphics Interface 2005*, pages 17–24, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada. Canadian Human-Computer Communications Society.

[Parish and Müller, 2001] Parish, Y. I. H. and Müller, P. (2001). Procedural modeling of cities. In *Proc. SIGGRAPH '01*, pages 301–308. ACM.

[Peck et al., 2009] Peck, S. M., North, C., and Bowman, D. (2009). A multiscale interaction technique for large, high-resolution displays. In *3DUI '09: Proc. of the 2009 IEEE Symp. on 3D User Interfaces*, pages 31–38, Washington, DC, USA. IEEE Computer Society.

[Peers et al., 2007] Peers, P., Tamura, N., Matusik, W., and Debevec, P. (2007). Post-production facial performance relighting using reflectance transfer. *ACM Trans. Graph.*, 26(3):52.

[Perez-Cazorla et al., 1997] Perez-Cazorla, F., Pueyo, X., and Sillion, F. (1997). Global illumination techniques for the simulation of participating media. In *8th Eurographics Wkshp on Rendering*.

[Pharr and Humphreys, 2004] Pharr, M. and Humphreys, G. (2004). *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Pierce et al., 1999] Pierce, J. S., Stearns, B. C., and Pausch, R. (1999). Voodoo dolls: seamless interaction at multiple scales in virtual environments. In *I3D '99: Proc. of the 1999 Symp. on I3D*, pages 141–145. ACM.

[Poupyrev et al., 1996] Poupyrev, I., Billinghurst, M., Weghorst, S., and Ichikawa, T. (1996). The go-go interaction technique: non-linear mapping for direct manipulation in vr. In *UIST '96: Proc. of the 9th annual ACM Symp. on User interface software and technology*, pages 79–80. ACM.

[Preetham et al., 1999] Preetham, A. J., Shirley, P., and Smits, B. (1999). A practical analytic model for daylight. In *SIGGRAPH*.

[Prusinkiewicz and Lindenmayer, 1990] Prusinkiewicz, P. and Lindenmayer, A. (1990). *The algorithmic beauty of plants*. Springer-Verlag New York, Inc.

[Rahman et al., 1993] Rahman, H., Pinty, B., and Verstraete, M. (1993). Coupled surface-atmosphere reflectance (csar) model 2. semiempirical surface model usable with noaa advanced very high resolution radiometer data. *J. Geophys. Res.*, 98(D11).

[Reche et al., 2004] Reche, A., Martin, I., and Drettakis, G. (2004). Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Trans. on Graphics (SIGGRAPH Conference Proceedings)*, 23(3).

[Sato et al., 1999] Sato, I., Sato, Y., and Ikeuchi, K. (1999). Acquiring a radiance distribution to superimpose virtual objects onto a real scene. *IEEE Trans. on Visualization and Computer Graphics*, 5(1):1–12.

[Shamir and Sorkine, 2009] Shamir, A. and Sorkine, O. (2009). Visual media retargeting. In *ACM SIGGRAPH Asia Courses*.

[Shashua and Riklin-Raviv, 2001] Shashua, A. and Riklin-Raviv, T. (2001). The quotient image: Class-based re-rendering and recognition with varying illuminations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(2):129–139.

[Shlyakhter et al., 2001] Shlyakhter, I., Rozenoer, M., Dorsey, J., and Teller, S. J. (2001). Reconstructing 3d tree models from instrumented photographs. *IEEE Comp. Graphics and App.*, 21(3).

[Slater et al., 2009] Slater, M., Khanna, P., Mortensen, J., and Yu, I. (2009). Visual realism enhances realistic response in an immersive virtual environment. *IEEE Computer Graphics and Applications*, 29:76–84.

[Sloan et al., 2002] Sloan, P.-P., Kautz, J., and Snyder, J. (2002). Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 527–536, New York, NY, USA. ACM.

[Snavely et al., 2008] Snavely, N., Seitz, S. M., and Szeliski, R. (2008). Modeling the world from Internet photo collections. *International Journal of Computer Vision*, 80(2):189–210.

[Soler et al., 2003] Soler, C., Sillion, F. X., Blaise, F., and Dereffye, P. (2003). An efficient instantiation algorithm for simulating radiant energy transfer in plant models. *ACM Trans. Graph.*, 22(2):204–233.

[Sorkine et al., 2004] Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., and Seidel, H.-P. (2004). Laplacian surface editing. In *SGP '04: Proc. EG/SIGGRAPH Symp. on Geometry processing*, pages 175–184.

[Stoakley et al., 1995] Stoakley, R., Conway, M. J., and Pausch, R. (1995). Virtual reality on a WIM: interactive worlds in miniature. In *Proc. of the CHI'95*.

[Stoschek, 2000] Stoschek, A. (2000). Image-based re-rendering of faces for continuous pose and illumination directions. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:1582.

[Tai et al., 2008] Tai, Y.-W., Brown, M. S., Tang, C.-K., and Shum, H.-Y. (2008). Texture amendment: Reducing texture distortion in constrained parameterization. *SIG - Asia*, to appear(to appear):to appear.

[Tan et al., 2008] Tan, P., Fang, T., Xiao, J., Zhao, P., and Quan, L. (2008). Single image tree modeling. *ACM Trans. Graph. (SIGGRAPH Asia)*, 27(5):1–7.

[Teller and Séquin, 1991] Teller, S. J. and Séquin, C. H. (1991). Visibility preprocessing for interactive walkthroughs. *Computer Graphics*, 25(4):61–68.

[Toledo et al., 2003] Toledo, S., Chen, D., and Rotkin, V. (2003). Taucs: A library of sparse linear solvers. `http://www.tau.ac.il/~stoledo/taucs/`.

[Tomasi and Manduchi, 1998] Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *ICCV '98: Proc. of the Int. Conf. on Comp. Vision*, page 839.

[Tsai, 1986] Tsai, R. (1986). An efficient and accurate camera calibration technique for 3d machine vision. *Proceedings of IEEE Conference on Computer Vision*.

[Wang et al., 2008] Wang, Y.-S., Tai, C.-L., Sorkine, O., and Lee, T.-Y. (2008). Optimized scale-and-stretch for image resizing. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH ASIA 2008)*, 27(5).

[Ward, 1992] Ward, G. (1992). Measuring and modeling anisotropic reflection. *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*.

[Ware and Osborne, 1990] Ware, C. and Osborne, S. (1990). Exploration and virtual camera control in virtual three dimensional environments. In *I3D '90: Proc. of the 1990 Symp. on I3D*, pages 175–183. ACM.

[Wei and Levoy, 2000] Wei, L.-Y. and Levoy, M. (2000). Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH'01*, pages 479–488. ACM SIGGRAPH.

[Xu et al., 2002] Xu, K., Stewart, J., and Fiume, E. (2002). Constraint-based automatic placement for scene composition. In *Graphics Interface*, pages 25–34.

[Yu et al., 1999] Yu, Y., Debevec, P., Malik, J., and Hawkins, T. (1999). Inverse global illumination: recovering reflectance models of real scenes from photographs. In *SIGGRAPH '99*.

[Yu and Malik, 1998] Yu, Y. and Malik, J. (1998). Recovering photometric properties of architectural scenes from photographs. In *SIGGRAPH '98*, pages 207–217, New York, NY, USA. ACM.

[Yu et al., 2004] Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., and Shum, H.-Y. (2004). Mesh editing with poisson-based gradient field manipulation. In *Proc. SIGGRAPH*, pages 644–651.

[Zeleznik et al., 1997] Zeleznik, R. C., Forsberg, A. S., and Strauss, P. S. (1997). Two pointer input for 3d interaction. In *I3D '97: Proc. of the 1997 Symp. on I3D*, pages 115–120. ACM.

[Zeleznik et al., 1996] Zeleznik, R. C., Herndon, K. P., and Hughes, J. F. (1996). Sketch: An interface for sketching 3d scenes. In *Proc. of SIGGRAPH 96*, Computer Graphics Proc., Annual Conference Series, pages 163–170.

[Zhou et al., 2006] Zhou, K., Huang, X., Wang, X., Tong, Y., Desbrun, M., Guo, B., and Shum, H.-Y. (2006). Mesh quilting for geometric texture synthesis. In *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH 2006)*, pages 690–697.

[Zotti et al., 2007] Zotti, G., Wilkie, A., and Purgathofer, W. (2007). A critical review of the preetham skylight model. In Skala, V., editor, *Journal of WSCG 2007*, 15.